Lip-Reading AI for European Portuguese

André F. Moreira, João B. Santos, João P. Pinto
University of Coimbra
{andremoreira, jbsantos, jopepinto}@student.dei.uc.pt

Abstract

Lip-reading/Visual Speech Recognition is the process of recognizing spoken words just by the lips of the speaker. This technique allows speech recognition in noisy environments, or in scenarios with no audio at all, such as security cameras. In this paper, we build a system that learns to recognize European Portuguese (EP) words, by extracting the mouth movements of the speaker and deducing the word they are saying. To do this, we use publicly available videos in EP that have subtitles (YouTube videos) to build the dataset, since there are no available EP lip-reading datasets. To process them, we used 3 different metrics to time the words in each subtitle; then we trained 3 different models (2 different CNN models and a hybrid CNN+LSTM model) to learn to recognize the words in the videos and tested the accuracy of all the combinations. We were able to obtain a 58.3% testing accuracy just from building a dataset that consists of only two 30-minute videos with the help of a syllable metric for subdividing subtitles and the hybrid CNN+LSTM model.

1 Introduction

Lip-reading is the process of analyzing and recognizing the movements of a speaker's lips and facial expressions. It can prove useful in scenarios where the audio is not available (for example, in surveillance cameras or for deaf people) or it is not clearly recognizable (noisy environments); different accents may also make the task harder. Lip-reading is a challenging problem, as the movement of the lips may not be enough to decode speech accurately. To our knowledge, this problem has been researched several times since 1954 [1], and in different languages, for example, English [2], Arabic [3], Japanese, Nepalese, Chinese, Mongolian [4], but no research was found specifically about European Portuguese (EP).

Multiple approaches done before use Convolutional Neural Networks (CNNs) [2] [3], Recurrent Neural Networks (RNNs) [7], Long Short-Term Memory Neural Network (LSTM) [9], and, more recently, Transformers [10] for the translation. Some of the most used datasets are the Oxford-BBC' lip reading datasets LRS2 and LRW (for example, [2]), MIRACL-VC1 and TCD-TIMIT. There are many datasets for lip-reading in different languages (the previously mentioned ones are in English); however, to our best knowledge, no EP datasets were found.

In this paper, we propose a way to translate lip/facial movements specifically to EP words, by building our own dataset. Since there are a lot of available videos, for example, on YouTube, with people looking at the camera while talking, we can use the timed subtitles of those videos to estimate the start and end of each word, and build the dataset with matching video frames and the spoken word on those frames.

In the remainder of the paper, we describe the state of the art in Section 2 and the material used in Section 3. Section 4 describes the methods used and Section 5 describes the experiments we did using those methods. Section 6 presents the results of the experiments, further discussed in Section 7. The conclusions and future works are presented in Section 8 and the references used are in Section 9.

2 State of the art

Different approaches have been used in visual speech recognition in the past, each with its own benefits and drawbacks, but it is possible to reach high accuracies with the most used ones, namely feature-based and end-to-end to approaches. Some also use context to improve speech recognition.

Feature extraction-based approaches rely on hand-crafted features like mouth/lip and nose contours for speech translation. Some of examples of such approaches are [7], where a multi-grained spatio-temporal modeling of the speaking process is done in order to capture nuances between words and styles of different speakers, obtaining an accuracy of 83.34% on the LRW dataset; in [8], a word-level visual speech recognition system scored an 77.9% testing accuracy on a customized version of the MIRACL-VC1 dataset, using a customized 3D CNN architecture by extracting the spatio-temporal features and mapping the prediction probabilities of the elements.

End-to-end approaches are basically a system that directly maps the raw image of the speaker's face to text. In [9], an LSTM is used directly on the pixels, achieving accuracies of 65.9% to 84.5%.

In [10], a cross-modal language model is used to generate multi-motion-informed context to improve lip-reading, using two main modules: the visual module and the Transformer decoder. Another implementation that uses a transformer is [11], where a CNN achieves an accuracy of 80.2% and with Visual Transformer Pooling it went up to 88.2%.

However, most of these works use datasets of high-resource languages (mainly English). In [5], a visual encoder is trained with masked predictions of speech units to learn general speech knowledge and a Language-specific Memory-augmented Decoder (LMDecoder) is proposed to learn language-specific knowledge from audio-text paired data, so that general speech knowledge and language-specific knowledge are combined to address the challenge of insufficient video-text paired data of low-resource languages.

3 Materials

The datasets we used were created by ourselves, since one of the major problems that we faced was the lack of EP datasets for this problem. To create the datasets, we relied on EP videos on YouTube that had an individual face the camera. Using the subtitles of the video and the mouth of the individual, we created an algorithm which allowed us to extract the spoken words.

To create this algorithm, we used preprocessing for both the video and the subtitles. For the video we separated it in frames and cropped the lips of the person, as for the subtitles we tried to use 3 different metrics to try to approximate the exact frames a word started and ended. The metrics we tried were the following: the length of the words, the length of the words with silence on spaces and the number of syllables of the words. The exact methods for the data processing will be explained in more detail in Section 4.

The tool used to create this dataset was the Python programming language, mainly focused on the libraries pyphen, for subdividing words into syllables, CV2, for the video and image processing, and tensorflow for the neural networks.

4 Methods

In this section, we specify how we approached the problem, and explain all the techniques we used along the way.

4.1 Subtitle and video preprocessing

As mentioned in Section 3, we tried processing the subtitles to approximate the frames where the speaker started and ended a word using three different metrics. We will now explain in a more detailed manner how each one of them works.

```
5
00:00:15,400 --> 00:00:18,560
uma coisa que é um problema cá em casa
```

Figure 1. Subtitle example

It is basically impossible to get an exact mapping of the words to the exact frames they start and end just from the subtitles alone, since the format of the subtitles is as shown in Figure 1. It may be possible to get better results if audio recognition is used in addition to the metric estimation, but since we do not have free access to any accurate EP audio recognition tools, we used just the metrics.

It is also notable that most videos have auto-generated subtitles, which may have some slight inaccuracies from the original words spoken on the video. However, some videos, like the TEDx ones mentioned in [5], have human-written subtitles, which can be useful to get more accurate results.

4.1.1 Length of the words

Algorithm 1 Approximate start/end frames of words with length metric

Input: Subtitles and corresponding video
Output:

```
for each subtitle:
  words <- subtitle.text.split()
  total_word_length <- sum_lengths(words)
  total_duration <- subtitle.end_time()-subtitle.start_time()
  current_time <- subtitle.start_time()
  for each word in words:
      start_frame <- current_time*fps+1
      word_duration <- word.length/total_word_length*total_duration
      current_time <- current_time + word_duration
      end_frame <- current_time*fps
      save(word, start_frame, end_frame)
  end for
end for
```

In this algorithm, we try to approximate the division of the frames for each word, by the relative length of each word in each subtitle. That is, in the sequence of words of each subtitle, we allocate a number of frames (of the total frames of the subtitle) to each word that is determined by the proportion of letters in each word relative to the total number of letters in the subtitle.

4.1.2 Length of the words with silence on spaces

Algorithm 2 Approximate start/end frames of words with length+silence metric

Input: Subtitles and corresponding video
Output:
1:**for each** subtitle:
2: words <- subtitle.text.split()
3: total_words <- words.length
4: total_duration <- subtitle.end_time()-subtitle.start_time()
5: total_word_duration <- sum(word.length/subtitle.text.length*total_duration **for each** word **in** words)
6: silence_duration = total_duration - total_word_duration
7: silence_interval = silence_duration/(total_words - 1) **if** total_words > 1 **else** 0
8: current_time <- subtitle.start_time()
9: **for each** word **in** words:
10: start_frame <- (current_time+offset)*fps
11: word_duration <- word.length/subtitle.text.length*total_duration
12: current_time <- current_time + word_duration
13: end_frame <- current_time*fps
14: save(word, start_frame, end_frame)
15: current_time <- current_time + silence_interval
16: **end for**
17:**end for**

This approach is relatively similar to the previous one, but instead of counting just the number of total letters, we also count the spaces. Then, between each word, we add a silence interval that corresponds to the space between those words, and, in addition, a small offset. This can help to not include small parts of each word in the other one.

4.1.3 Number of syllables in each word

Algorithm 3 Approximate start/end frames of words with syllables metric

Input: Subtitles and corresponding video
Output:
1:**for each** subtitle:
2: words <- subtitle.text.split()
3: total_syllables <- count_syllables(words)
4: total_duration <- subtitle.end_time()-subtitle.start_time()
5: current_time <- subtitle.start_time()
6: **for each** word **in** words:
7: start_frame <- current_time*fps+1
8: word_duration <- count_syllables(word)/total_syllables*total_duration
9: current_time <- current_time + word_duration
10: end_frame <- current_time*fps
11: save(word, start_frame, end_frame)
12: **end for**
13:**end for**

Since, in EP, most syllables have a relatively similar duration while speaking, we thought that dividing the frames by the number of syllables of each word on each subtitle made a lot of sense. However, this approach has one requirement: there is a need for another algorithm that splits the words into syllables that works in EP. For that, we used the python library pyphen.

This algorithm, in the sequence of the words in the subtitle, allocates, for each word, a number of frames that is given by the proportion of syllables of that word relative to the total syllables of the subtitle.

4.2 Mouth Detection

To build the dataset that labels mouth shapes to words, we obviously need to be able to extract the mouth from the video's frames. To do that, we used python dlib's face detector, as well as a pre-trained lip shape predictor available with the MIRACL-VC1 dataset on [12].

That allowed us to detect, frame by frame, the location of the mouth of the speaker, as well as the frames where no one is speaking to the camera (for example, when in a video, the camera focuses on an object to further explain its uses), which was useful to not introduce false data into the dataset.

However, once again, the face detector we used is not perfect, and it is possible to obtain better results if a more accurate one is used.

4.3 Training and Testing

Since the diversity of words spoken on a big enough video is enormous, we chose to only consider words that had lots of instances, and only a select few while the dataset only consists of a couple videos.

However, there is a problem with words that had many more instances than the others in the video: the model will become biased towards that class/word if trained with too many of those. To handle that problem, we decided to balance the classes (the different target words), by setting a limit of instances of the word that will be inserted in the training set (for example, the number of instances of the word with the least instances).

Once the dataset is split into train, validation and test, a neural network (NN) architecture that would fit the problem must be chosen. We tested with 3 NNs: 3-layer 3D CNN, 4-layer 3D CNN and 3-layer 3D CNN + LSTM. Convolutional Neural Network (CNN) is a useful NN in this scenario, since we are dealing with feature recognition in images. We also tried with a Long Short-Term Memory neural network (LSTM) in addition to the CNN, since a visual spoken word is a temporal sequence of mouth images, and the LSTM could prove useful since it takes into consideration the previous instances (and next instances in the case of a BiLSTM). In the NN training, we used Early Stopping to, once again, try to avoid overfitting.

More specifically, the 3-layer 3D CNN consists of three groups of 3D Convolution and 3D Max Pooling layers, followed by a flatten layer, and two groups of dense and dropout layers and finally, a dense layer with a *softmax* activation function and number of classes as number of neurons. The 4-

layer variation follows a similar structure, but has four groups of CNN and Max Pooling layers instead of three, and three groups of dense and dropout layers instead of two. The 3D CNN + LSTM network is similar to the 3-layer 3D CNN but has a Reshape layer, an LSTM layer and a dropout layer in between the convolution layers and the flatten layer.

To test, we simply picked up the test split of the dataset, fed it into the trained NN and compared the output with the correct word.

5 Experiments

As there aren't datasets prepared exactly for our purpose, as we mentioned before, we created our own using the biggest source of videos online, YouTube. We downloaded some videos made by different EP creators and the corresponding auto-generated subtitles. With that we began building the dataset. Specifically, we used two 30 minutes long videos to build the dataset.

Those two videos allowed us to perform a design base research (DBR) methodology, where we conducted 3 rounds of experiments, to test various parameters and allow us to draw conclusions about each setup, and used those conclusions on each round do get better results in the next one. For each of the different parameters, the data processing and model training were done from zero, for the results to be the most unbiased possible.

5.1 First round of experiments

In this first round of experiments, we were looking more to test the system, and the method was as follows: we preprocessed only the first video with each of the three subtitle subdivision metrics (length, length+silence, syllables), and choose to use only the words with more than 5 instances, and to limit the number of instances of each word to 10, to avoid the existence of unbalanced classes, which can create a bias to that class, as mentioned in section 4.3.

Then, we proceeded to prepare the training, validation and testing sets, with a 60%, 20% and 20% partition, and train each of the three NN architectures presented in Section 4.3 (3-layer CNN, 4-layer CNN, 3-layer CNN + LSTM). After training each of the models with the training sets (and validating with the validation set), we proceeded to test the results, by comparing the classifications made by each model to the ground truth classifications in the testing set, and calculated the percentage of words correctly classified over the total words in the set.

5.2 Second round of experiments

In this round, we decided to narrow down the number of words used, and chose only those with at least 20 instances (20 since it was the maximum number that we could reach that still had a few words, more than five, that met the condition), and used only 20 instances of those words, even for those that had more. Another requirement we used for the words chosen was that they had to have at least 2 syllables, because we noticed that, due to the smaller words having a smaller amount of frames allocated, and the uncertainty of the approximation of the division the words in each subtitle being high, there were some of these words that were incorrectly timed (the cropped images showed the person saying part of the previous or next word).

The training, validation and testing of the models was done in a similar way to the first experiment.

5.3 Third round of experiments

In this last round of experiments, we used both videos to build the dataset, and increased the number of instances required to 40, since there were already more word instances to work with. That is, we chose only the words with at least 40 instances, and limited the number of instances to 40 for those that had more than 40.

The training, validation and testing of the models was done in a similar way to the first two experiments.

6 Results

6.1 First round of experiments

3-Layer 3D CNN

Subtitle metric	Length	Length + Silence	Syllables
Accuracy	7.7%	6.3%	7.4%

Figure 2. Accuracy (on the test set) of the 3-layer 3D CNN

4-layer 3D CNN

Subtitle metric	Length	Length + Silence	Syllables
Accuracy	3.8%	2.1%	3.4%

Figure 3. Accuracy (on the test set) of the of the 4-layer 3D CNN

3-layer 3D CNN + LSTM

Subtitle metric	Length	Length + Silence	Syllables
Accuracy	3.8%	4.2%	3.4%

Figure 4. Accuracy (on the test set) of the 3-layer 3D CNN + LSTM

As mentioned, in Section 5.1, this round of experiments was mostly done to test the system and was not so focused on the performance of the models. Consequently, the results were not very good, and we quickly understood why: we were training the models with very few instances of each word, and with a lot of different words, which would obviously not allow the model to learn to correctly distinguish the words.

As such, these experiments did not allow us to draw any conclusions among the different metrics and models, because the results were similarly low.

With all these problems in mind, we proceeded to the second round of experiments.

6.2 Second round of experiments

3-layer 3D CNN

Subtitle metric	Length	Length + Silence	Syllables
Accuracy	21.4%	7.1%	35.7%

Figure 5. Accuracy (on the test set) of the 3-layer 3D CNN

4-Layer 3D CNN

Subtitle metric	Length	Length + Silence	Syllables
Accuracy	14.3%	14.3%	14.3%

Figure 6. Accuracy (on the test set) of the 3-layer 3D CNN + LSTM

3-Layer 3D CNN + LSTM

Subtitle metric	Length	Length + Silence	Syllables
Accuracy	14.3%	14.3%	28.6%

Figure 7. Accuracy (on the test set) of the 3-layer 3D CNN + LSTM

In this round of experiments, we had more instances of each word to train the models and fewer classes (words), and, consequently, we got much better results. The 3-layer CNN proved to be the superior architecture in this case, except for the length+silence metric. The 2 other networks, being more complex and having more layers, most times overfitted the training set and were biased towards one of the classes, getting slightly worse results.

It was also in this phase that we started noticing that the syllable metric was the one that allowed us to obtain the best results.

6.3 Third and last round of experiments

3-Layer 3D CNN

Subtitle metric	Length	Length + Silence	Syllables
Accuracy	8.3%	25%	50%

Figure 8. Accuracy (on the test set) of the 3-layer 3D CNN

4-Layer 3D CNN

Subtitle metric	Length	Length + Silence	Syllables
Accuracy	16.7%	16.7%	41.7%

Figure 9. Accuracy (on the test set) of the of the 4-layer 3D CNN

3-Layer 3D CNN+LSTM

Subtitle metric	Length	Length + Silence	Syllables
Accuracy	16.7%	16.7%	58.3%

Figure 10. Accuracy (on the test set) of the 3-layer 3D CNN + LSTM

In this round, with the addition of the second video, we could set the bar higher for the minimum required number of instances of the words that were used, since we were working with more data, so we decided to only use words with at least 40 instances (and keeping the "cut" for the words with more than 40 instances at 40 still). This proved to improve the results a lot, as we could already reach more than half of the testing set correctly classified (58.3% with syllables metric and 3-layer 3D CNN + LSTM).

7 Discussion

With the results presented in the previous section, we can draw some conclusions about the different parameters we used in the three rounds of experiments.

Firstly, with the increasing amount of data, it becomes evident that the syllables metric is the best one at accurately timing the words in a subtitle, as it gave indistinguishably the best results in the last experiment. This shows the importance of preprocessing in these kinds of experiments. The choice of the neural network architecture is obviously very important, as it is also noticeable that the 3-layer CNN and CNN+LSTM neural networks gave the best results, but the difference in performance from the syllable metric preprocessing was even bigger than between the different models.

One of the most noticeable things is that the accuracy, in general, increases a lot when the data increases, as one would expect, since the models are trained with more data, thus are more capable of correctly identifying the words. The CNN+LSTM architecture, specifically, proved to get a lot better with more data, especially with the syllables metric. Since the increase in data was only from one video to two videos, we expect that if we created a bigger dataset with dozens of videos, the accuracies would get better, and the models would be able to identify more words.

Another point that must be addressed is the overfitting that happened with the more complex models (4-layer CNN especially, but also 3-layer CNN+LSTM sometimes). Since the amount of training data we used is relatively low, the 4-layer CNN often overfitted the training set, as it got more than 90% accuracy on the training set and less than 20% on the testing set. We expect that this architecture would be more adequate for larger amounts of data.

8 Conclusion and Future Work

In this project, we did an exploratory study where we tested a way to recognize visual speech in EP, by using online videos with subtitles to build a dataset, and training models with that dataset. We tested three metrics for timing the words within the subtitles, from which the syllable metric obtained the best results. We trained three different types of neural network architectures, from which a 3-layered 3D CNN + LSTM neural network proved to be the one that can get better results in the end, when paired with the syllables metric preprocessing.

However, in the experiments we did, we were limited in a lot of ways, and if these limitations are overcome, we believe much better results can be obtained.

One of the limitations is the lack of an EP audio recognition tool, so that the words from the subtitles of the videos can be correctly timed and synced with the image, which would, in addition (or could even entirely replace) to the metrics we used to subdivide the subtitles, make the images correctly represent the word said in those time frames, and make the dataset more consistent, which would allow for a better training of the models.

Another limitation of the experiments we conducted was the amount of data we were able to work with, since we only used one and two 30 minute long videos in the experiments, and just from using one to two videos, the accuracies almost doubled, so with more videos, we believe that the accuracies would go up by a lot, and the number of words that the model would be able to recognize would also increase. Perhaps an open AI solution (possibly with active learning, human-in-the-loop or reinforcement learning from human feedback), where multiple people can interact with the model and train it, could increase the range of words it can recognize and also its accuracy.

9 References

[1] W. H. Sumby, Irwin Pollack; Visual Contribution to Speech Intelligibility in Noise. J. Acoust. Soc. Am. 1 March 1954; 26 (2): 212–215. https://doi.org/10.1121/1.1907309

[2] S. NadeemHashmi, H. Gupta, D. Mittal, K. Kumar, A. Nanda and S. Gupta, "A Lip Reading Model Using CNN with Batch Normalization," 2018 Eleventh International Conference on Contemporary Computing (IC3), Noida, India, 2018, pp. 1-6, doi: 10.1109/IC3.2018.8530509.

[3] Waleed Dweik, Sundus Altorman, Safa Ashour, Read my lips: Artificial intelligence word-level arabic lipreading system, Egyptian Informatics Journal, Volume 23, Issue 4, 2022, Pages 1-12, ISSN 1110-8665, https://doi.org/10.1016/j.eij.2022.06.001.

[4] T. Saitoh, K. Morishita and R. Konishi, "Analysis of efficient lip reading method for various languages," 2008 19th International Conference on Pattern Recognition, Tampa, FL, USA, 2008, pp. 1-4, doi: 10.1109/ICPR.2008.4761049.

[5] Kim, Minsu & Yeo, Jeong & Choi, Jeongsoo & Ro, Yong. (2023). Lip Reading for Low-resource Languages by Learning and Combining General Speech Knowledge and Language-specific Knowledge.

[6] Wang, C. (2019). Multi-Grained Spatio-temporal Modeling for Lip-reading. ArXiv, abs/1908.11618.

[7] T. Makino et al., "Recurrent Neural Network Transducer for Audio-Visual Speech Recognition," 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Singapore, 2019, pp. 905-912, doi: 10.1109/ASRU46091.2019.9004036.

[8] P. Nemani, G. S. Krishna, N. Ramisetty, B. D. S. Sai and S. Kumar, "Deep Learning based Holistic Speaker Independent Visual Speech Recognition," in IEEE Transactions on Artificial Intelligence, 2022, doi: 10.1109/TAI.2022.3220190.

[9] S. Petridis, Z. Li and M. Pantic, "End-to-end visual speech recognition with LSTMS," 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 2017, pp. 2592-2596, doi: 10.1109/ICASSP.2017.7952625.

[10] X. Ai and B. Fang, "Cross-Modal Language Modeling in Multi-Motion-Informed Context for Lip Reading," in IEEE/ACM Transactions on Audio, Speech, and Language

Processing, vol. 31, pp. 2220-2232, 2023, doi: 10.1109/TASLP.2023.3282109.

[11] K R Prajwal, Triantafyllos Afouras, Andrew Zisserman; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 5162-5172

[12] MIRACL-VC1 Dataset available on Kaggle: https://www.kaggle.com/datasets/apoorvwatsky/miraclvc1/, 08/12/2023