Automatic Intelligent Translation of Videos

Shivam Bansal*

Abstract

There are a lot of educational videos online which are in English and inaccessible to 80% population of the world. This paper presents a process to translate a video into another language by creating its transcript and using TTS to produce synthesized fragments of speech. It introduces an algorithm which synthesyses intelligent, synchronized, and easily understandable audio by combining those fragments of speech. This algorithm is also compared to an algorithm from another research paper on the basis of performance.

Keywords: Automatic Video Translation, Automatic Video Dubbing

1. Introduction

Nowadays, internet is filled with videos from all around the world in different languages. There is a lot of quality educational content which should be accessible to everyone. But language barrier makes that difficult. Most of the educational content on the internet is in English, whereas less than 20% of the world's population understands English. Making that content available in other languages will allow a lot of people access to content online.

This paper presents a process to translate a video into another language by creating its transcript and using TTS to produce synthesized fragments of speech. It tackles the issue when synthesysed translation is longer than the origanal speech by introducing an algorithm which creates intelligent, synchronized, and easily understandable audio. Another algorithm was analysed from another research paper. And the results from both the algorithms were analysed and compared.

In the experiment videos from ocw.mit.edu were used as the main focus

*Corresponding author.
Email address: mail@shivamb.com (Shivam Bansal)

of the experiment is on education videos. These videos were in English and were translated to Hindi.

Plausible results were obtained. Although they didnt match those of manual dubbing, but they were amazing. It was concluded that a lot educational content can be translated in different languages.

2. Process

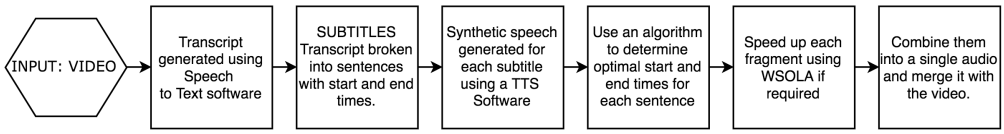

Figure 1: Outline of the procedure

2.1. Generating Transcript

Transcripts are to be genered with the following features :

- It should be seperated by sentences.
- It should also have each sentence's starting and ending time.
- It should be able to classify the sentences by the speaker.

Such transcripts can be generated using IBM Watson Speech-To-Text engine with a very high accuracy. Although the accuracy may not be very high for entertainment videos, but educational videos are usually very clear and hence will have almost 100% accuracy. These sentences are like subtitles. From now on these may be reffered to as subtitles.

2.2. Translation of Transcript

The generated transcript needs to be translated into the required language. This is an easy task and can be done using several services. Google Translator was used for this experiment. A new transcript is stored in the required language.

2.3. Generation of Synthetic Speech for each Subtitle

Synthetic speech is to be generated for each translated subtitle. For different speakers, different voices are required. The generated speech needs to be clear, loud, not slurred, and reasonably fast to get good results. There are several TTS (Text-To-Speech) services available for this step. Google TTS was used for this experiment.

2.4. Determining Scene or Topic changes

In a video, the scene or topic might change several times. And we dont want that our speech from one scene extends to the other since that would make it ugly. Hence we use technologies to detect scene and topic change using audio and video clues [1] .

2.5. Determining optimal Start and End Times

Now the generated synthetic speeches are to be combined to form a single audio file, which is to be finally merged with the video. This is the primary focus of this research paper. The issue that arises here is that the generated synthetic speech can be longer than the origanal speech. In our experiment English was used as the input language and Hindi was used as the output language, and on average sentences were 1.6 times longer. This made it hard to synchronise with the video. High speeding up factors made the speech less understandable and didn't serve the origanal purpose. Although a reasonable change in the starting and ending time of sentences could reduce the speeding up factors, large shifts could mess up the events in the video and make it ununderstable.

Hence an algorithm which determines optimal start and end times for each subtitle, keeping the speeding up factors low and shift in positions minimal, was required to produce an intelligent and understandable speech efficiently.

In this experiment two algorithms were tested and their results were compared. One of them was created while working on this experiment and the other one was introduced in a research paper by Jindrich Matousek and Jakub Vit. [2]

In each of the algorithms the resulting subtitles must have the following properties :

- a subtitle must not overlap a neighbouring subtitle
- a subtitle must not exceed a scene or topic change

- the speeding up factor has to be minimal
- the shifting in the position of the subtitle has to be minimal

2.5.1.

.

Notations Each subtitle i has the following properties :

- X_i = Starting time of origanal subtitle
- Y_i = Ending time of each subtitle
- D_i = Length of origanal subtitle $(Y_i - X_i)$
- L_i = Length of synthetic speech generated

The algorithms are as follows :

2.5.2. Spring Based Model (Brief) [2]

This model is based on physical simulation of springs. In this model each subtitle is taken to be a spring with its length = L_i. The spring's left end is denoted by X_i' and its right end is denoted by Y_i' . Its length at any point of time is denoted by D_i' which is equal to $Y_i' - X_i'$. At the beginning, $X_i'$is set equal to X_i and $Y_i'$is set equal to Y_i. After this compression is released, the spring follows the Hooke's Law with a restoring force of :

$$F_i = -K_a.(f_i' - 1)$$

where $f_i' = [\frac{l_i}{d_i'}]$. To push X_i'towards X_i and Y_i' towards Y_i, two additional springs are put at the ends of each subtitle which each have a restoring force of :

- $F_i^{(a)} = -K_b.(x_i' - x_i)$
- $F_i^{(b)} = -K_b.(y_i' - y_i)$

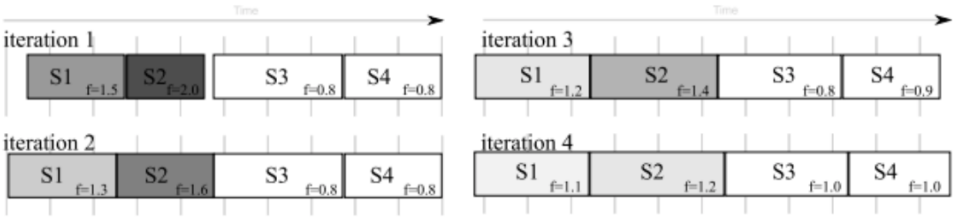

Figure 2: An illustration of the spring based algorithm. Darkness of the colour represents the high speeding up factor. Image taken from the origanal research paper. [2]

The ratio $[\frac{K_a}{K_b}]$ allows to control between preferences either towards synthetic speech with small speeding-up factors but with start and end positions somewhat receded from the original positions $K_a < K_b$ or towards synthetic speech at original positions, but more speeded up $K_a > K_b$.

The optimal timing of the subtitle and the optimal speeding up factor is obtained when equillibrium of forces $F_i, F_i^{(a)}, F_i(b)$ is reached.

This implementation of this algorithms does a physical simulation of springs which is not very efficient.

2.5.3. Linear Model

In this model we use constant K which specifies how important it is to minimize speeding-up factor than to minimize shift in position. This is a relatively simple method which surprisingly gives amazing results. The start and end positions of resulting subtitle i is denoted as X_i' and Y_i'. The length of resulting subtitle is denoted as D_i' which is equal to $Y_i' - X_i'$. In this algorithm we need to adjust the subtitles such that the following is minimised :

$$L = \sum_{i=1}^{n} |X_i - X_i'| + |Y_i - Y_i'| + K * f_i$$

where n is the total number of subtitles , $f_i = [\frac{l_i}{d_i'}]$, and K allows to control between preferences either towards synthetic speech with small speeding-up factors but with start and end positions somewhat receded from the original positions or towards synthetic speech at original positions, but more speeded up.

Implementation. For the simplicity of the implementation, we assume the following :

- All the subtitles have origanal length more than 1 second. It is practically impossible for a meaningful sentence to be less than that.
- Two neighbouring subtitles can start and end at the same second.
- We take smallest unit of time to be 1 second. Since in 1 unit of second only 2 subtitles can intersect, we do not need to take a time unit smaller than that.

The algorithm is applied seperately on the different scenes of the video, since different scenes wont share a sentence. The problem can be solved using dynamic programming where $dp[i][t]$ denotes the minimal value L for the first i subtitles such that they fit in time $<= t$ seconds from the start. This can be formulated as :

$$dp[i][t] = min(dp[i][t-1], |y_i - t| + \min_{j=1}^{t-1}(dp[i-1][j] + |x_i - j| + K * \frac{l_i}{t-j}) \ )$$

Performance Analysis. Let m be the duration of video in seconds. Then the memory complexity of this algorithm is $O(N * M)$ and the time complexity is $O(N * M^2)$. On average a subtitle is approximately 5 seconds long. Hence we can assume that $N \approx \frac{M}{5}$. Which makes the time complexity equal to $O(\frac{M^3}{5})$.

2.6. Combining the synthetic audio

Now that we have synthetic audio ready with their optimal start and end times, we combine them to make a complete audio file. Since this involves speeding up individual speeches, it can change the pitch of the speech and hence make it understandable. We use WSOLA algorithm to speed up the speeches which maintains the clarity and pitch of the voice. Then an audio file is generated and merged with the video.

3. Results

3.1. Performance

Figure 3 shows how in both the algorithms, as the no. of subtitles or the length of the video increases, the average speeding up factor decreases. This is because in short videos, there is not much of a scope to shift subtitles.

As the spring based model is based on simulation, it often takes more time on large videos. But on smaller videos it is very efficient and gives

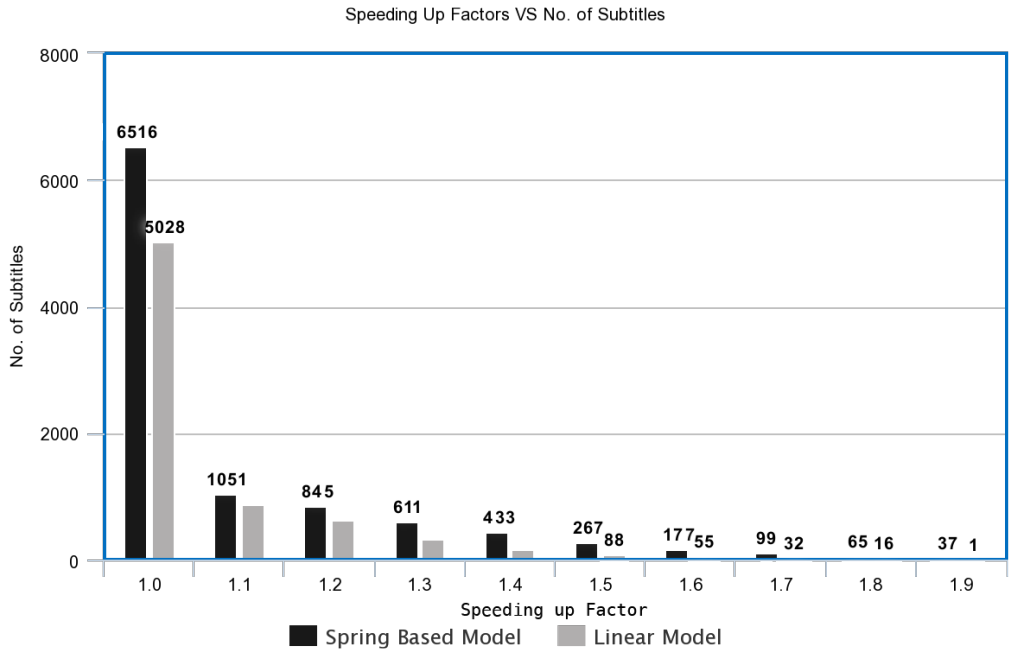

Figure 3: Average speeding up factors VS No. of subtitles

better results. On larger videos Linear Model gave better results and turned out to be more efficient.

Both the algorithms worked pretty well and gave amazing results on the videos used to test them.

3.2. Problems Faced

The following problems were faced :

1. Automatic translation gives excellent results on educational videos since they have a very smooth and medium paced speech. But these algorithms dont work well on entertainment videos since these algorithms dont have the ability to recognise emotions. Entertainment videos also require perfect synchronisation which is very hard.
2. The translation may not be very accurate. In this experiment English was used as the input language and Hindi was used as the output language. Sometimes the translation was not 100% accurate. Moreover, in some cases the Hindi translation by Google Translator was very hard

to understand. It contained words which only highly qualified Hindi Literature students would know.

3. For the experiment different services were used seperately such as STT, translation, and TTS, hence the processing was very slow. When a real application is developed, these can be speeden up.

4. Conclusions

Automatic translation of videos is a new technology and has a lot of pros and cons. Although it is very efficient with respect to manual dubbing, the results are not as good. Manual dubbing is very perfect with each sentence created of the same length, which is not possible in automatic translation.

This algorithm can be used to translate thousands of educational videos over the internet into different languages and hence spread education.

References

[1] S.-C. Chen, M.-L. Shyu, W. Liao, C. Zhang, Scene change detection by audio and video clues.

[2] J. Matousek, J. Vit, Improving automatic dubbing with subtitle timing optimisation using video cut detection.