UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Eduardo Rocha Soares

Temporal Segmentation of Video Lectures: a speech-based optimization framework

Juiz de Fora

2020

Eduardo Rocha Soares

Temporal Segmentation of Video Lectures: a speech-based optimization framework

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. D.Sc. Eduardo Barrére

Juiz de Fora

2020

Ficha catalográfica elaborada através do programa de geração automática da Biblioteca Universitária da UFJF, com os dados fornecidos pelo(a) autor(a)

Soares, Eduardo Rocha.
Temporal Segmentation of Video Lectures : a speech-based optimization framework / Eduardo Rocha Soares. – 2020.
77 f. : il.

Eduardo Barrére
Dissertação (mestrado acadêmico) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2020.

1. Video Lectures. 2. Temporal Segmentation of Video Lectures. 3. Linear Programming. 4. Natural Language Processing. 5. Speech Processing. I. Barrére, Eduardo, orient. II. Título.

Eduardo Rocha Soares

"Temporal Segmentation of Video Lectures: a speech-based optimization framework"

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 28 de fevereiro de 2020.

BANCA EXAMINADORA

Prof. Dr. Eduardo Barrére – Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Jairo Francisco de Souza – Coorientador
Universidade Federal de Juiz de Fora

Prof. Dr. Rudinei Goularte
Instituto de Ciências Matemáticas e de Computação

Dedico este trabalho à memória de minha irmã Beatriz e aos meus pais, Maria Aparecida e Marcilio.

ACKNOWLEDGEMENTS

First of all, I would like to thank my family, especially my parents, Maria Aparecida and Marcilio, for all the love and support that were essential for this moment.

I also thank my girlfriend Fernanda for all the love, understanding and companionship dedicated to me during this period. For believing in me and encouraging me to pursue that goal. Thank you very much, my love!

To my friends, my thanks for all the support and fun moments. Without you, this whole process would be much more difficult.

I am also grateful to the professors at DCC and PPGCC who contributed to my academic education through their valuable teachings and advice. I am especially grateful to my advisor and friend Eduardo Barrére for his guidance and partnership over the years, from undergraduate to master's degree.

Finally, I would like to thank all ICE employees who have always helped me and CAPES for the financial support that made this work possible.

"Whereas the beautiful is limited, the sublime is limitless, so that the mind in the presence of the sublime, attempting to imagine what it cannot, has pain in the failure but pleasure in contemplating the immensity of the attempt."

Immanuel Kant

RESUMO

As videoaulas são muito populares hoje em dia. Seguindo as novas tendências de ensino, estudantes procuram cada vez mais por vídeos educacionais na Web com os mais diferentes propósitos: aprender algo novo, revisar conteúdo para exames ou apenas por curiosidade. Infelizmente, encontrar conteúdo específico nesse tipo de vídeo não é uma tarefa fácil. Muitas videoaulas são extensas e abrangem vários tópicos, sendo que nem todos são relevantes para o usuário que encontrou o vídeo. O resultado disso é que o usuário acaba gastando muito tempo ao tentar encontrar um tópico de interesse em meio a conteúdo que é irrelevante para ele. A segmentação temporal de videoaulas em tópicos pode resolver esse problema ao permitir que os usuários naveguem de maneira não-linear entre os tópicos existentes em uma videoaula. No entanto, se trata de uma tarefa dispendiosa que precisa ser automatizada. Por esse motivo, neste trabalho, propomos um *framework* de otimização para o problema de segmentação temporal de videoaulas. Nossa proposta utiliza apenas informações da fala do professor, portanto, não depende de recursos adicionais, como slides, livros didáticos ou legendas geradas manualmente. Isso a torna versátil, pois podemos aplicá-la a uma ampla variedade de videoaulas, uma vez que requer apenas que o discurso do professor esteja presente. Para fazer isso, formulamos o problema como um modelo de programação linear, onde combinamos recursos prosódicos e semânticos da fala que podem indicar transições de tópicos. Para otimizar esse modelo, usamos um algoritmo genético elitista com busca local. Através dos experimentos, fomos capazes de avaliar diferentes aspectos de nossa abordagem, como sua sensibilidade à variação de parâmetros e comportamento de convergência. Além disso, mostramos que nosso método foi capaz de superar métodos do estado da arte, tanto em Recall quanto em F1-Score, em dois conjuntos diferentes de videoaulas. Por fim, disponibilizamos a implementação de nosso *framework* para que outros pesquisadores possam contribuir e reproduzir nossos resultados.

Palavras-chave: Videoaulas. Segmentação Temporal de Videoaulas. Programação Linear. Processamento de Linguagem Natural. Processamento de Fala.

ABSTRACT

Video lectures are very popular nowadays. Following the new teaching trends, students are increasingly seeking educational videos on the web for the most different purposes: learn something new, review content for exams or just out of curiosity. Unfortunately, finding specific content in this type of video is not an easy task. Many video lectures are extensive and cover several topics, and not all of these topics are relevant to the user who has found the video. The result is that the user spends so much time trying to find a topic of interest in the middle of content irrelevant to him. The temporal segmentation of video lectures in topics can solve this problem allowing users to navigate of a non-linear way through all topics of a video lecture. However, temporal video lecture segmentation is a time-consuming task and must be automatized. For this reason, in this work we propose an optimization framework for the temporal video lecture segmentation problem. Our proposal only uses information from the teacher's speech, therefore it does not depend on any additional resources such as slides, textbooks or manually generated subtitles. This makes our proposal versatile, as we can apply it to a wide range of different video lectures, as it only requires the teacher's speech on the video. To do this, we formulate this problem as a linear programming model where we combine prosodic and semantic features from speech that may indicate topic transitions. To optimize this model, we use a elitist genetic algorithm with local search. Through the experiments, we were able to evaluate different aspects of our approach such as sensibility to parameter variation and convergence behavior. Also, we show that our method was capable of overcoming state-of-the-art methods, both in Recall and in F1-Score, in two different datasets of video lectures. Finally, we provide the implementation of our framework so that other researchers can contribute and reproduce our results.

Key-words: Video Lectures. Temporal Segmentation of Video Lectures. Linear Programming. Natural Language Processing. Speech Processing.

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AM	Acoustic Model
ACO	Ant Colony Optimization
ASR	Automatic Speech Recognition
BoW	Bag-of-Words
DNNs	Deep Neural Networks
GA	Genetic Algorithm
GPOS	Global Pareto-optimal Set
GS	Grid Search
HMMs	Hidden Markov Models
HS	Hypothesis Search
IR	Information Retrieval
ILS	Iterated Local Search
LP	Linear Programming
LM	Language Model
ML	Machine Learning
NLP	Natural Language Processing
NP	Noun Phrases
OCR	Optical Character Recognition
PCM	Pulse Code Modulation
POS	Part-of-speech
PSO	Particle Swarm Optimization
STE	Short Term Energy
TF-IDF	Term Frequency-Inverse Document Frequency
TS	Tabu Search
TSP	Travelling Salesman Problem

VAD	Voice Activity Detection
WMD	Word Mover's Distance

CONTENTS

1 INTRODUCTION

With the continued proliferation of e-learning, video lectures have been an effective way to convey educational content. Video lectures offer several advantages for students, for instance, the possibility of the student review the content taught in regular classes or filling the gap due to absence. In addition, video lectures allow students to follow their own learning pace by pausing and moving the video forward or backward (RONCHETTI, 2010). Despite the high availability of video lectures online, students still have a lot of trouble locating the appropriated material for their studies, because too much irrelevant content is returned in their searches due to the overload of information that exists on the Web (MITRA; SRIVASTAVA, 2020).

Also, it is common for students to still have difficulty accessing the desired content, even when they finally find a video lecture that supposedly addresses it. This because it is very common for students to search for a specific topic within the main subject of the video lecture, and so they want to go directly to the point when it begins. But unfortunately, videos usually do not provide navigation that allows users to access instantly a specific topic. Thus, to find the beginning of a specific topic in an extensive video lecture, they must to watch the entire video or try to forward or rewind the video until they find what they want. That process is time-consuming and negatively contributes to the user's experience (YANG; MEINEL, 2014).

Improvements in video lecture search can be made in a repository. For instance, it is necessary to provide means for students to instantly access specific topics in video lectures and also to navigate non-linearly across topics. This type of instantaneous and non-linear navigability between the main topics of the video lecture is ideal for learning and can significantly improve the student experience in e-learning systems (PAVEL; HARTMANN; AGRAWALA, 2014). The most common way to accomplish this is to segment video lectures into topics. That is, to partition the video lecture into smaller units, where each of these units represents a specific subject addressed in the video (i.e. topics). Although human-made topic segmentation is the most accurate, it is very time-consuming and difficult to do in large existing video repositories (LIN et al., 2005a). Therefore, it is important to automate this task to make it feasible in practice. The automatic temporal segmentation of video lectures has been a challenge for the multimedia and information retrieval areas due to the nature of the problem involving both content processing and semantic understanding.

Although there are many approaches in the literature to automate this task, most of them rely on features from manually generated materials such as slides, textbooks, and subtitles. This makes those approaches very dependent on the availability of specific resources, which limits the universe of video lectures where they can be applied. For this

reason, the motivation of this work is to remove these barriers by proposing a framework able to work on circumstances where none of these resources are present using only the teacher's speech to segment the video lectures.

Our main contribution in this work is the proposal of a novel optimization framework capable of obtaining temporal segmentation of video lectures in topics, using only features automatically extracted from the audio track. Thus, our proposal can be applied to different types of video lectures, without human effort and without relying on any other resource that may or may not be available. The framework proposed combines both prosodic and semantic features into a linear programming model that we optimize to find a solution. Also, as a minor contribution, we make available the implementation of our framework as a distributed software architecture that can be easily deployed on the server-side.

1.1 PROBLEM DEFINITION

The problem of temporal segmentation of video lectures may have different definitions depending on the author. This difficulty comes mainly from the subjective concept of the topic. One of the most widespread definitions and assumed in this work is that a topic consists of a logical and semantically meaningful unit of the video lecture that is contiguous in time (GALANOPOULOS; MEZARIS, 2019; TUNA et al., 2015).

Therefore, assuming that definition, we can represent a topic by its time boundaries. That is, assuming a set of topics T of a video lecture V, we can represent each topic $t_i \in T$ as the closed interval $[init_i, end_i]$, where $init_i$ and end_i are the beginning and end time of t_i in the video lecture, respectively, also, $end_i > init_i$ and $init_{i+1} > end_i$. So, the problem this work approaches can be simplified as: given a video lecture V as input, automatically find the time boundaries $[init_i, end_i]$ of all topics t_i of the V, where $i = 1, 2, 3, 4, ..., N$ and N is the number of topics in V.

1.2 OBJECTIVES

The main objective of this work is to propose and validate through experiments an optimization framework able to segment the video lectures into topics by combining into a linear programming (LP) model the semantic and prosodic features from the teacher's speech.

In our proposal, the semantic features are extracted from automatic speech recognition (ASR) transcripts while the prosodic ones are extracted from physical aspects of speech such as pause duration, loudness and fundamental frequencies. With this, we want to evidence that our framework provides a versatile approach for the problem of temporal

segmentation of video lectures and it is capable of obtaining good solutions even without using any human-made resources to guide the segmentation.

1.3 OUTLINE

This work is organized as follows. Chapter 2 provides the theoretical fundamentals needed to understand this work. Chapter 3 presents the related work. In Chapter 4, we describe our proposal. The experiments and obtained results are presented in Chapter 5. Finally, in Chapter 6, we give our conclusions and discuss future works.

2 FUNDAMENTALS

In this chapter, we present the main fundamentals and concepts needed for a full understanding of this work.

2.1 LINEAR PROGRAMMING (LP)

LP is a method that aims to find the values of a set of variables, called decision variables, that maximize or minimize a linear objective function (VANDERBEI et al., 2015). Suppose we have a linear model with m decision variables, they can be represented by the vector $X = [x_0, x_1, x_2, x_3, ...x_m]$. Where $m \in \mathbb{N}$ and $x_0 = 1$.

In LP problems, a objective function consists in function $f : \mathbb{R}^m \mapsto \mathbb{R}$ that follows the formulation (VANDERBEI et al., 2015; MURTY, 1980):

$$f(x_0, x_1, x_2, x_3, ..., x_{m-1}) = c_0 + x_1 \cdot c_1 + x_2 \cdot c_2 + x_3 \cdot c_3 + ... + x_m \cdot c_m \tag{2.1}$$

Or in a vectorized form:

$$f(X) = X^T C \tag{2.2}$$

Where X^T is the transposed of the vector X and C is the vector of constants $C = [c_0, c_1, c_2, c_3, ..., c_m], m \in \mathbb{N}$.

Besides of the objective function, LP problems may have some constraints involving the decision variables. These constraints can be related to the domain of decision variables or can be more complex such a linear equality or inequality (VANDERBEI et al., 2015). The equation bellow shows an example of a LP constraint.

$$\sum_{i=1}^{m} x_i \cdot c_i \leq b \tag{2.3}$$

To summarize, the LP approach is often used to model a real-life problem by optimizing a linear function without compromising any constraints. For example, the classic Travelling Salesman Problem (TSP) of finding the route that includes all points without repetition, returning to the point of origin (constraints), while minimizing the total distance traveled (objective function).

2.1.1 Multi-objective Problems

A multi-objective problem is one that has more than one objective function to be optimized. That is, we want to optimize a set of functions $\overline{f} = \{f_1(X), f_2(X), ..., f_n(X)\}$, $n > 1$, instead of a single function (JONG; SPEARS, 1992).

When optimizing a multi-objective model, there are some challenges we may face. The main issue is to approximate the global Pareto optimized set (GPOS), as the search space grows dramatically with the number of objective functions and variables (SCHÜTZE et al., 2019). GPOS is the state that it is impossible to improve the solutions of any objective function without making others worse. Therefore, the task of approximating the entire GPOS can be very computationally burdensome. This is why, in some cases, heuristic approaches can be very effective in approaching GPOS without requiring so much computational effort, as opposed to exact methods.

2.1.2 Metaheuristics

A metaheuristic can be defined as an iterative method that applies high-level procedures to explore solution space by escaping local optima (GENDREAU; POTVIN et al., 2010). Generally, metaheuristics are employed to find solutions to problems for which we do not know efficient algorithms (NP-complete).

One of the main characteristics of the metaheuristics is to be independent of the problem. Therefore, it is easy to be adapted for different optimization problems. Also, a metaheuristic can be defined as a general purpose framework that guides a more specific heuristic, that is problem dependent, through the solution space in an efficient way (BLUM; ROLI, 2003). Examples of metaheuristics are: Ant Colony Optimization (ACO), Genetic Algorithm (GA), Iterated Local Search (ILS), Particle Swarm Optimization (PSO), Tabu Search (TS), among others.

2.1.3 Genetic Algorithm (GA)

GA is a bio-inspired metaheuristic presented by John Holland in 1975 (HOLLAND, 1975). GA works with multiple solutions by iteration (generation). In GA, the set of solutions found by the algorithm is called population. And, each member of the population is an individual. GA mechanisms are based on natural selection, where the most adapted individuals are more likely to survive and generate offspring, transmitting parts of their genes, while the less fit die. Also, as with natural selection, each individual has a chance of suffering mutation in its genes. The mutation is an indispensable mechanism of GA, as it is responsible for introducing variability in the population, preventing the algorithm from getting stuck prematurely in the local optima. The mutation allows genes not present in the population to appear, which makes the algorithm able to exploit other neighborhoods in the solution space (MAN; TANG; KWONG, 1996).

In GA, each individual is represented by an array that encodes a solution to the problem we are solving. This array is called “chromosome” and each position of this array is a “gene” (WHITLEY, 1994). Suppose we have an LP problem with 4 binary decision variables. We can then represent each individual of the GA as a binary chromosome array,

where each gene g_i corresponds to one variable of the problem. Figure 1 illustrates this example of representation.

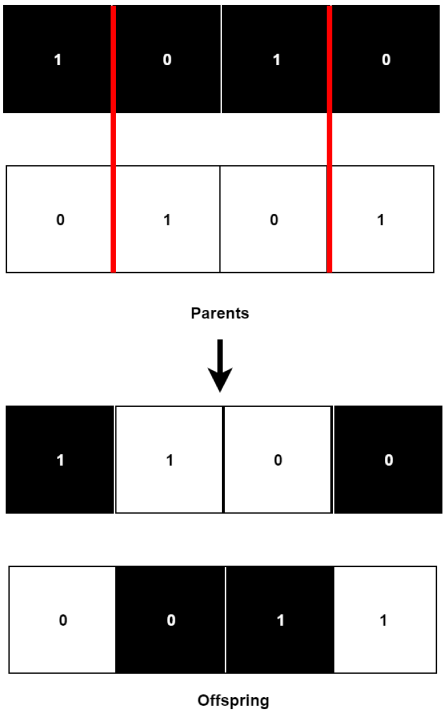
Figure 1 – Example of binary chromosome representation in GA

Source: Created by the author (2020)

Another element of GA is the fitness function $\phi : \mathbb{R}^m \mapsto \mathbb{R}$ that estimates how good a chromosome is to solve a problem. Generally, the fitness function is closely related to the objective function of the problem and to the probability of an individual surviving to the next generation (MAN; TANG; KWONG, 1996).

As in natural selection, in GA, the most adapted individuals survive and have the chance to generate offspring. The process of generating new individuals from the combination of parent chromosomes is called "crossover" (MAN; TANG; KWONG, 1996; HOLLAND, 1975). There are different ways of combining the parent chromosomes in the crossover step. The most basic way of performing it is to randomly select one or more points in parent chromosomes and using different parts of them to compose a new one. Figure 2 shows an example of two-point crossover for binary chromosomes.

Figure 2 – Example of two-point crossover for binary chromosomes



Source: Created by the author (2020)

The mutation is essential to the proper functioning of the GA, preventing its early convergence. That is because there is a tendency for individuals in the population to become very similar over the generations. This lack of genetic variability makes the algorithm not to be able to improve the solutions. The mutation step is responsible for introducing variability into the population by applying a slight perturbation to the individual's genes (MAN; TANG; KWONG, 1996; WHITLEY, 1994; HOLLAND, 1975). There are many mutation operators that can be found in the literature. One of the most common when dealing with binary chromosome representations is the bit inversion, which consists of inverting an individual's gene given a small probability.

Given the main concepts behind the GA, in the Algorithm 1, we can summarize the steps of the standard GA according to (MAN; TANG; KWONG, 1996).

Algorithm 1: Standard Genetic Algorithm

Data: population size, number of generations, crossover rate, mutation rate
Result: The individual with best fitness value

```
set the population size ||P||;
set the number of generations G;
set the crossover rate C;
set the mutation rate M;
randomly initialize the population P;
evaluate(P);
iterations := 0;
while iterations < G do
    P̄ := select_parents();
    recombine(P̄, C);
    mutate (P̄, M);
    evaluate (P̄);
    select_survivors(P̄, P)
    iterations := iterations + 1
end
return the best individual
```

2.1.4 Local Search

Before defining the concept of local search, we need to define what is a neighborhood of a solution. Suppose we have an optimization problem P, for example, the TSP. We define as an instance of P the tuple (S, f), where S is the set of all feasible solutions and f is the objective function. A neighborhood is defined by a function Φ that maps each solution $s_i \in S$ in a set N_i of solutions that are somehow close to s_i (AARTS; AARTS; LENSTRA, 2003). For example, consider a solution to a problem that can be represented

as a binary array. We could define a neighborhood function that consists of inverting the sequence of elements of a range of the array (e.g., **011**1001 $\mapsto$ **110**1001).

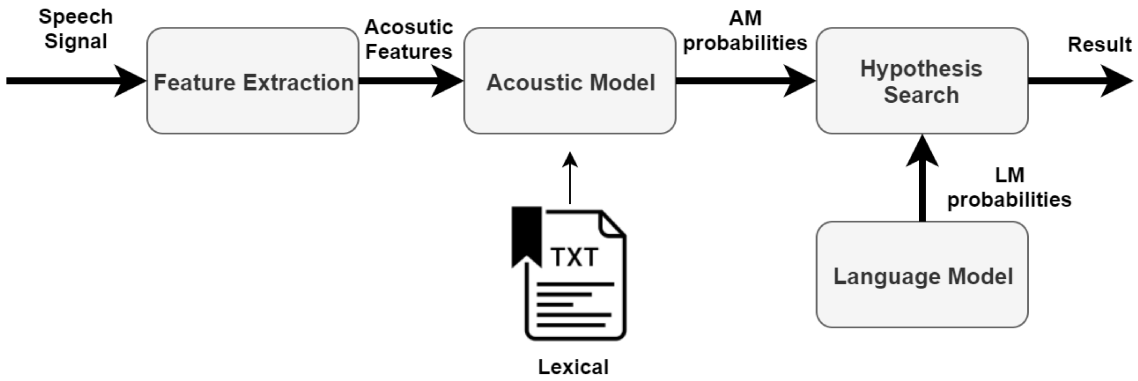
In general terms, local search is an iterative process that receives a solution s_i as input and searches for better solutions in its neighborhoods. This process continues until no improvement is found or until a stopping criterion is reached (e.g. number of iterations or execution time) (AARTS; AARTS; LENSTRA, 2003). Local search is used to improve the solutions found by a global optimizer (e.g., GA, PSO, ACO, etc.) making the convergence to optimal solutions faster (SINDHYA; DEB; MIETTINEN, 2008).

2.2 AUTOMATIC SPEECH RECOGNITION

Automatic Speech Recognition (ASR) is a technology that allows computers to recognize words and phrases from a speech input signal (JAIN; RASTOGI et al., 2019). Today, due to the advance of computational power, ASR systems are widely used in many everyday applications such as games, personal assistants, accessibility tools, translation software, automatic subtitle generation, etc (LU; LI; FUJIMOTO, 2020; YU; DENG, 2016). That is why, more than never, to build an accurate ASR system still an important and challenging subject of signal processing and natural language processing research fields (LU; LI; FUJIMOTO, 2020; YU; DENG, 2016; LEE; SOONG; PALIWAL, 2012).

Although can be variations, a common ASR architecture include some main components responsible for performing the recognition: Feature Extraction, Acoustic Model (AM), Language Model (LM) and Hypothesis Search (HS) (YU; DENG, 2016; GRUHN; MINKER; NAKAMURA, 2011). Figure 3 shows a schema of an ASR architecture.

Figure 3 – A typical ASR architecture



Source: adapted from Yu and Deng (2016)

2.2.1 Feature Extraction

The feature extraction component is an interface that receives a speech signal as input and performs a sequence of signal processing techniques such as denoising and

domain transformation to extract relevant vectors of features to be used by the acoustic model (YU; DENG, 2016; LEE; SOONG; PALIWAL, 2012).

2.2.2 Acoustic Model (AM)

AM is a centerpiece of ASR architecture. It is responsible for modeling the likelihood of a sequence of speech units (e.g., phonemes) occurring based on the acoustic features extracted from the audio signals. Also, the AM maps these sequences into vocabulary words with the help of a phonetic dictionary called the lexical model. Usually, the lexical model consists of a plain text file containing all the words of vocabulary and their respective phonetic transcription (JAIN; RASTOGI et al., 2019; GRUHN; MINKER; NAKAMURA, 2011). Mathematically, we say that an acoustic model A gives us the conditional probability of occurrence of a sequence of feature vectors O given a word sequence W. This definition is shown in Equation 2.4.

$$A = P(O|W) \tag{2.4}$$

Nowadays, two of the main algorithms used to train acoustic models are the Hidden Markov Models (HMMs) (HADIAN et al., 2018) and the Deep Neural Networks (DNNs) (HINTON et al., 2012). Although these algorithms can perform the phonetic recognition task very well, training them requires large labeled data sets of audio.

2.2.3 Language Model (LM)

Essentially, LM gives us the probability that a sequence of words W from a vocabulary will occur (i.e., $P(W)$). To do this, LM takes the previous N words in sequence to calculate the probability of the next. The value of N determines the type of LM. For example, if N equals 3, we say it is a 3-gram model (SONG; CROFT, 1999), and so on. The importance of LM for ASR architecture is to prevent non-grammatical sentences from being recognized by giving a low probability of occurrence to not usual word sequences. As a result, LM significantly improves the accuracy of ASR systems by solving acoustic ambiguities and reducing the search space (GULZAR et al., 2014). Let $W_r = \{w_1, w_2, w_3, w_4, ..., w_k\}, k \in \mathbb{N}$, be a word sequence already recognized by the ASR system and w_{k+1} a arbitrary word from the vocabulary. Considering a N-gram LM, the probability of w_{k+1} be the next word recognized in W_r is given by the Equation 2.5.

$$P(w_i) = P(w_{k+1}|w_k, w_{k-1}, w_{k-2}, ..., w_{k-N+1}) \tag{2.5}$$

2.2.4 Hypothesis Search

Finally, the Hypothesis Search (HS) unit is responsible for finding the word sequence W that is most likely to occur, given a sequence of acoustic vectors O (Equation 2.6).

$$argmax_W P(W|O) \tag{2.6}$$

Applying the Bayes Rule (VANAJAKSHI; MATHIVANAN, 2017):

$$argmax_W P(W|O) = argmax_W \frac{P(O|W) \cdot P(W)}{P(O)} \tag{2.7}$$

Note that in the Equation 2.7, after we applied the Bayes Rule, two already known terms appeared: $P(O|W)$ and $P(W)$. These terms are correspondent to the acoustic and language models, respectively. Therefore, given an acoustic vector sequence as input, the hypothesis search unit generates as output the W word sequence that maximizes the combination of the AM and LM probabilities (VANAJAKSHI; MATHIVANAN, 2017; YU; DENG, 2016).

2.3 VOICE ACTIVITY DETECTION (VAD)

Voice Activity Detection (VAD) is the problem of determining which audio segments contain speech and which do not. In other words, VAD is the task of discriminating audio segments with speech from those with background noise only (PASAD; SABU; RAO, 2017; HUGHES; MIERLE, 2013; SOHN; KIM; SUNG, 1999).

Several applications use VAD algorithms as an audio preprocessing step. For example, ASR systems tend to be very sensitive to background noise, so being able to differentiate between speech and noise segments can significantly improve the performance of those systems. Also, VAD can be used to optimize bandwidth in voice call applications (PASAD; SABU; RAO, 2017; SOHN; KIM; SUNG, 1999).

VAD approaches use features extracted from the audio signal as input to a speech/non-speech classifier. Several classification algorithms have been employed in the literature to perform the VAD, such as Gaussian Mixture Models (GMMs), HMMs, Recurrent Networks (RNNs), rule-based approaches, and others (PASAD; SABU; RAO, 2017; HUGHES; MIERLE, 2013; SOHN; KIM; SUNG, 1999).

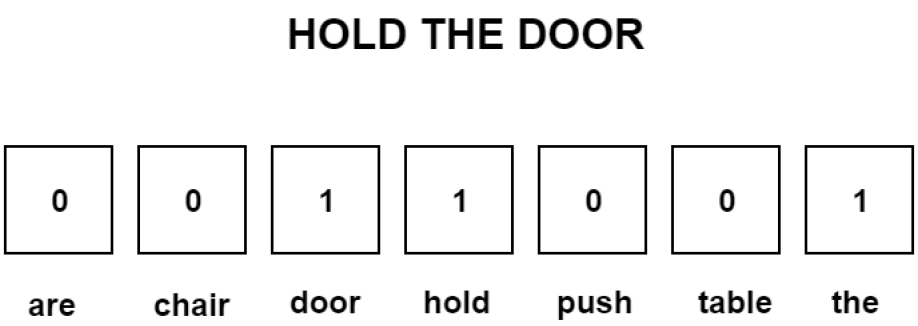
2.4 TEXT REPRESENTATION AND COMPARISON MEASURES

When dealing with Natural Language Processing (NLP) and Information Retrieval (IR), one of the main topics is how to represent text corpora in a way that facilitates their manipulation and extraction of knowledge from it. For this reason, researchers have

studied and proposed various text representation structures to optimize the search for documents and allow their syntactic and semantic comparison. In this section, we present the concepts of text representation algorithms and distance metrics that are important in this work.

2.4.1 Bag-of-Words (BoW)

Figure 4 – Example of Bag-of-Words representation of the sentence “Hold the Door”



Source: Created by the author (2020)

The BoW is a traditional method first proposed in (HARRIS, 1954) for text representation. In BoW, a sentence or document is represented by some statistical measures, such as the frequency or TF-IDF (RAMOS et al., 2003) of its words or N-grams, ignoring grammatical structure and order. The BoW representation of a document can be seen as a M dimension vector, $M \in \mathbb{N}$, where each position of this vector corresponds to a word or N-gram from a vocabulary. Due to its simplicity, the BoW model can be applied to large text data sets, making it widely used to this day. Figure 4 illustrates the BoW representation of an example sentence, considering the word frequencies as feature values.

2.4.2 Word Embeddings

Word embeddings are a class of language models that learn semantic representations of vocabulary words from their context of occurrence in sentences and paragraphs (KUSNER et al., 2015). In this way, several NP and IR applications have used Word Embeddings to improve information extraction and text comparison (NORASET et al., 2017).

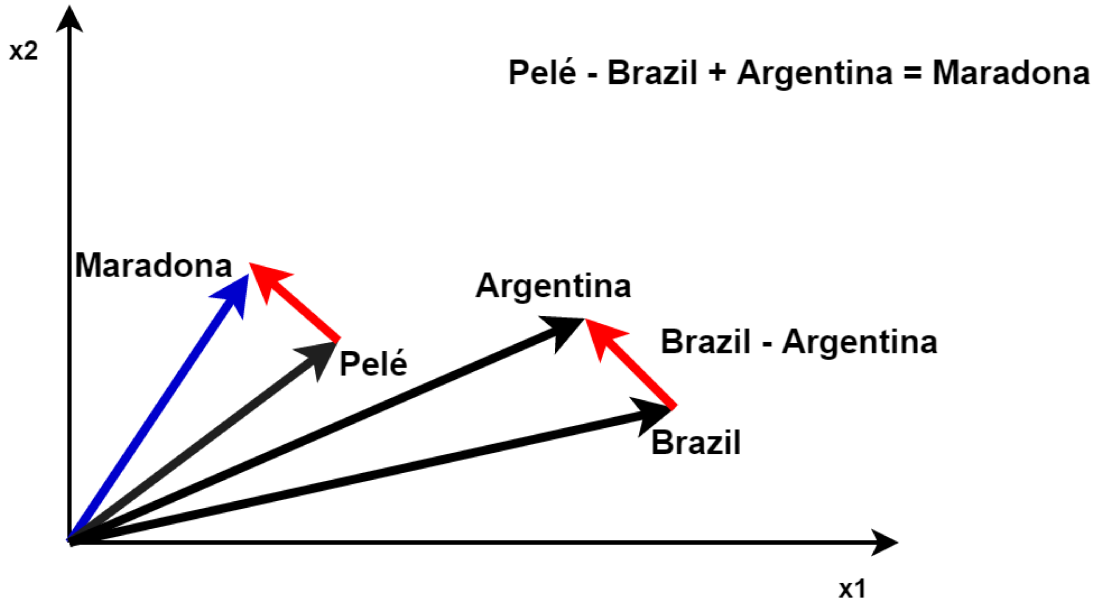
2.4.3 Word2Vec

The Word2Vec is a kind of Word Embedding technology that was first proposed in (MIKOLOV; YIH; ZWEIG, 2013) as an efficient way to represent words from large datasets as continuous multidimensional vectors. The need for this kind of representation

arose from the fact that traditional statistical models such as the BoW have certain limitations, such as the impossibility of extracting any relationship between words. On other hand, in addition to the notion of syntactical similarity, the Word2Vec representation allows extracting semantic relationships between words by performing algebraic operations with word vectors. For example, in a vector space defined by a Word2Vec model, it can be inferred that the words "Pelé" and "Maradona" are related just by performing the simple vector operation $vec(Pele) - vec(Brazil) + vec(Argentina) \approx vec(Maradona)$ (MIKOLOV et al., 2013a; MIKOLOV; YIH; ZWEIG, 2013).

The Word2Vec model is obtained by training Machine Learning (ML) algorithms, usually, neural network architectures, that learn the vector coding of vocabulary words by analyzing the context where they occur in a training set. In this way, in Word2Vec, words with similar meanings or related to similar contexts are represented by close vectors (RONG, 2014; MIKOLOV et al., 2013a). Figure 5 shows a 2-D vector space defined by a Word2Vec model from which it is possible to extract some relationships between words through some algebraic manipulations.

Figure 5 – Example of 2-D vector space defined by Word2Vec model



Source: Created by the author (2020)

2.4.4 Cosine Similarity

Comparing texts is one of the most recurring and fundamental tasks in NLP and IR. For this task, several metrics presented in the literature can be used, depending on the representation model that is adopted. An advantage of using vector representations for texts is that we can calculate distance and similarity measures that are defined in the

vector space $\mathbb{R}^n$. In this context, the cosine similarity is one of the most popular measures (TAKANO et al., 2019; BAEZA-YATES; RIBEIRO-NETO et al., 1999).

The cosine similarity between two text documents is defined as the cosine of the angle between the vectors representing these documents, where the smaller this angle, the more similar the documents are. Given two text documents represented by the vectors $\vec{d_1}$ and $\vec{d_2}$, their cosine similarity is defined by (HUANG, 2008):

$$cos(\vec{d_1}, \vec{d_2}) = \frac{\vec{d_1} \cdot \vec{d_2}}{\left\|\vec{d_1}\right\| \times \left\|\vec{d_2}\right\|} \tag{2.8}$$

where $\vec{d_1} \cdot \vec{d_2}$ is the inner product between $\vec{d_1}$ and $\vec{d_2}$ and $\left\|\vec{d_1}\right\| \times \left\|\vec{d_2}\right\|$ is the multiplication of their norms. Since each dimension of $\vec{d_1}$ and $\vec{d_2}$ contains a non-negative weight, the cosine ranges from 0 (angle of 90 degrees) to 1 (angle of 0 degrees).

As cosine similarity considers only the angle between document vectors, it is invariable for document length. For example, consider d_1 = "hold the door" and d_2 = "hold the door hold the door" as two text documents. Possible BoW vector representations for them are $\vec{d_1} = (1, 1, 1)$ and $\vec{d_2} = (2, 2, 2)$, respectively. Calculating the cosine similarity between $\vec{d_1}$ and $\vec{d_2}$ will result in 1. That is, in terms of cosine similarity they are equal (HUANG, 2008).

2.4.5 Word Mover's Distance (WMD)

As already mentioned before, Word2Vec is a way of representing semantic relations between words in a vector space. However, documents may contain several words. So from a Word2Vec perspective, a document d_i is an $M \times N$ matrix, where M is the number of dimensions of Word2Vec representation and N the number of words in d_i.

WMD is a metric proposed by Kusner et al. (2015) that uses the Word2Vec representation to get the distance between two text documents. Basically, the WMD is formulated as a Transportation Problem (LING; OKADA, 2007), where the distance between two documents d_i and d_j is given by the minimum cumulative cost to move all words (Word2Vec representation) from d_i to d_j (KUSNER et al., 2015). Unlike the cosine distance, the WMD ranges in $[0, +\infty)$, where the lower the value, the more similar the documents.

The results reported by Kusner et al. (2015) showed that WMD was able to reduce errors in classification tasks compared to other text distance metrics. Which makes WMD a promising approach for measuring the distance between text documents.

2.5 PROSODIC FEATURES

Prosodic features are commonly defined as any nonverbal speech features such as pitch, volume, pause duration, rhythm, and others (FRICK, 1985). Prosodic features are an important part of speech in many languages. For example, logical structures of speech, such as sentences, paragraphs, and topics, are often demarcated by prosodic features, such as long pauses or pitch variations (HIRSCHBERG; LITMAN; SWERTS, 2004; SHRIBERG et al., 2000; FRICK, 1985). Also, prosodic features may express emotions in speech such as anger, stress, sadness, happiness, among others (FRICK, 1985). Thus, prosodic features carry useful speech information that is invariable concerning language. The main prosodic features explored by literature researchers are:

- **Pitch**: is a prosodic feature related to the perception of fundamental frequency variations (F0) in the speaker's voice. Researchers agree that speakers tend to increase their pitch range by introducing a new subject or emphasizing parts of speech (MAYER; JASINSKAJA; KÖLSCH, 2006; ARONS, 1994; WILLIAMS, 1985). Thus, a higher pitch is a powerful tip about transitions between topic or sub-topics.

- **Loudness**: is another prosodic feature intimately related to the prominence of words and sentences. It is the perception caused by the variation in air pressure produced by the energy used in speech (CHE et al., 2016; OLIVEIRA, 2000; FRICK, 1985). Some authors like Kochanski et al. (2005) argue that loudness may play a more important role than pitch in detecting highlights on the speech.

- **Pause Duration**: when we speak, we often use pauses to delimit units of information, such as words, phrases, paragraphs, or topics. As with pitch and loudness, research in computational linguistics has concluded that the length of these pauses is longer in parts of the speech where the speaker introduces a new subject or wants to emphasize something (CHE et al., 2016; MAYER; JASINSKAJA; KÖLSCH, 2006; FRICK, 1985).

3 RELATED WORK

Literature proposals for temporal segmentation of video lectures segmentation can be classified according to some key characteristics: the domain of video lectures they are applied, the nature of features extracted from video lectures and the way these features are combined to generate the topic segmentation. In this chapter, we present the main research on the temporal segmentation of video lectures, separated by the type of feature used by the algorithm. Thus, we intend to give an overview of the existing lines of research on this topic.

3.1 TEXT-BASED APPROACHES

The research of Galanopoulos and Mezaris (2019) proposes a fully text-based method for temporal segmentation of video lectures using word embeddings to calculate the semantic similarity between text windows of the video lectures subtitles and, thus, to identify possible points of transition between topics. The authors used humanly generated subtitles to perform the experiments. Therefore, they did not evaluate the impacts of using automatically generated subtitles by ASR in their proposal.

Previous research, such as (BALAGOPALAN et al., 2012) and (TUNA et al., 2015), also have proposed methods for the segmentation of video lectures in topics that makes exclusive use of textual features. In (BALAGOPALAN et al., 2012), a Naive Bayes classifier is trained to separate key-phrases from other terms in the audio transcript of video lectures. Those key-phrases are later used to segment the video lectures into topics. In the work of Tuna et al. (2015) a investigation about the impact of the use of textual sources for video lecture segmentation was conducted. The authors' conclusion is that the use of optical character recognition (OCR) in video lecture slides provides better data for topic segmentation than Automatic Speech Recognition (ASR), in large part because the ASR has many errors involved. However, ASR outperformed OCR as the best data source for video lecture segmentation when the authors manually corrected the automatic transcripts.

One of the main shortcomings of these purely textual approaches is that not all relevant information present in the video lecture is in textual form. This means that these approaches are not able to detect more subtle cues about the transitions of topics in the video lecture. Therefore, this may compromise the quality of temporal segmentation obtained by them.

3.2 VISUAL APPROACHES

Research such as (DAVILA; ZANIBBI, 2017; LEE et al., 2017) and (SUBUDHI et al., 2017) focus on video lectures segmentation where the manuscript is shown on the screen. These papers use visual features to perform a temporal segmentation that is used later to create a video lecture summary. In (DAVILA; ZANIBBI, 2017), the key-frames of the video lecture are extracted and binarized. Then, the binarized frames are segmented to separate the background of the whiteboard from the handwritten content and obtain content regions. Next, the topic segmentation is done in order to minimize the conflict between content regions in the whiteboard. Content regions are in conflict if they occupy the same space in the whiteboard at different time intervals. Thus, they cannot be on the same topic and need to be segmented.

Similarly, Lee et al. (2017) also proposes an approach for video lecture segmentation that is based on the extraction of handwriting from the board. First, the board region is segmented through k-means and next, a connected component technique is applied to replenish the area of the board covered by the teacher's body. Then, adaptive threshold and denoising methods are employed to extract the handwriting. Finally, the extracted handwritings are used to structure the video lecture in topics.

Although the proposal of Subudhi et al. (2017) uses only visual features, it is a little bit different from those in Davila and Zanibbi (2017) and Lee et al. (2017) because its focus is on classifying video shots according to their importance. First of all, abrupt and gradual visual transitions in the video are detected through histogram calculations to obtain the shot boundaries. After that, each video shot is classified in low-content, average-content, content, high-content or non-content based on some visual features that represent the amount of handwritten content appearing on the screen. For example, a video shot where the camera is focused on the speaker's face would be classified as non-content. Otherwise, a video shot where the white/blackboard or a paper note are shown fully filled with equations would be classified as high-content. This classification is used later to obtain a summary of the video lecture, which makes possible for students to go directly to the most important parts of the video.

These purely visual approaches, in essence, are based on detecting significant visual transitions between frames in the video. One of the great gaps of these approaches is that they depend on very specific types of video lectures so that they can extract the necessary features for their segmentation. Since many video lectures are recorded in a single shot, without major visual changes.

3.3 SPEECH-BASED APPROACHES

Another research line is the exploration of different features from the teacher's speech to perform the topic segmentation task. The works of Che, Yang and Meinel (2018) and Soares and Barrére (2018), for example, spoken sentences in textual form and prosodic features are used to obtain a topic segmentation.

In the work of Che, Yang and Meinel (2018), the sentences may be obtained from subtitle or automatic speech recognition. Then, prosodic features of each sentence like pitch, loudness, pause rate, and duration are extracted to obtain highlight points in the video lecture. As each sentence has a timestamp associated, a temporal segmentation can be obtained. On other hand, Soares and Barrére (2018) proposes a data processing pipeline to obtain a temporal topic segmentation in video lectures. Three kinds of features are extracted from audio's track over this pipeline: prosodic, transcripts, and the semantic annotation of transcripts. Next, those features are used to calculate affinity matrices that are linearly combined and given as input to a clustering algorithm. The clusters formed by this algorithm are sequential parts of a video that corresponds to topics. One of the advantages of using only audio features is because most of the video lectures have an audio track where the teacher's speech predominates. Thus, these kinds of proposals can cover a greater amount of video lectures than those that depend on the presence of slides, whiteboards or textbooks to work.

Continuing in the line of research of the audio approaches, the works of (MAO et al., 2014) and (STOWELL et al., 2015) can be mentioned. In (MAO et al., 2014), the authors have presented an approach for speech emotion recognition using convolutional neural networks, while in (STOWELL et al., 2015) the state-of-the-art of methods for audio event detection is reported. Although none of the articles deals directly with the issue of temporal segmentation of video lectures, it can be seen an application of them in this problem. For example, the recognition of emotions in speech can be used to detect moments of emphasis in the teacher's speech, which may indicate the transition of topics. In the same way that audio event detection algorithms can be trained to find important moments in video lectures, such as the sound of the click of remote control to pass slides or the sound of the teacher erasing the blackboard, for example.

One of the great advantages of speech-based approaches is that, as the vast majority of video lectures contain the teacher's speech in the foreground, these methods can be applied satisfactorily in virtually all types of video lectures. On the other hand, these approaches, by not using other sources of information, are not able to take advantage of other resources present in the video lecture that could improve the quality of the temporal segmentation.

3.4 MULTIMODAL APPROACHES

Proposals that use different sources of information to perform the topic segmentation task are very common in the literature. In (SHAH et al., 2014), the authors present the ATLAS system as solution to the ACM Multimedia 2014 Grand Challenge on automatic temporal segmentation and annotation. The approach of ATLAS is based on the fusion of visual and textual transition cues. To obtain the visual transition cues, two SVM models were trained. For the extraction of textual cues, a N-gram language model was employed. In the experiment section, we can see that the fusion of the visual and textual cues perform better than use them separately. Similarly, another system for temporal segmentation of video lectures was proposed in (FURINI; MIRRI; MONTANGERO, 2018). This system uses visual transitions in the video and audio energy to obtain cut points in order to generate a topic-based playlist to improve navigability through video lecture content.

In (BISWAS; GANDHI; DESHMUKH, 2015), the authors proposed a method to automatically generate table of contents for video lectures. However, to achieve this goal, the temporal video lecture segmentation is needed. For this, the authors first calculate an importance score for each word from slides based on visual features (location, boldness, underlineness, capitalization and isolation). Next, the importance of spoken words is calculated through a graph-based model. Lastly, the temporal segmentation problem is modeled as a dynamic programming problem combining both visual and spoken features. Besides, the method proposed may work with only speech or visual features, however results showed that the combination of the two kinds of features can deliver a better segmentation.

In (MAHAPATRA; MARIAPPAN; RAJAN, 2018) the final objective is the generation of a table of contents for video lectures. Although, there is an important difference in relation to the work of (BISWAS; GANDHI; DESHMUKH, 2015). Basically, the proposal of (MAHAPATRA; MARIAPPAN; RAJAN, 2018) uses OCR technology and speech features to find the time the teacher starts talking about the bullet points that appear over slide-show. The authors proposed modeling of the problem as a knapsack tree problem, where there is a budget constraint that determines the maximum number of topics that a solution can contain. That is, if one slide has four bullet points, the algorithm may segment up to four topics from one single slide. It can be a problem, because over-segmentation can have the opposite effect in relation to improving the information retrieval in the video lecture. Another issue is in the case of an addressed topic that does not appear explicitly in the slide as a bullet point or title. In this case, it will not be segmented. This strong dependence of well-structured slides makes this method be applicable only in a reduced set of video lectures.

These multimodal approaches are capable of obtaining excellent results when taking advantage of different sources of information present in the video lectures. Despite this,

the quality of the solutions obtained by these methods can be compromised in cases where not all sources of information are available.

3.5 CONCLUDING REMARKS

This chapter presented the main research related to this work. With this, we can place our work in the area of temporal segmentation of video lecture through a survey of the main techniques used by the top-notch researches.

In the literature, there are several approaches for the task of temporal segmentation of video lectures. And just as important as the algorithm used to decide the temporal segmentation, are the features it uses. The choice of features directly affects the algorithm's ability to find topic boundaries in video lectures, as each contributes to different levels and types of information. Furthermore, some features used by the approaches may set limits on the domain of video lectures where they can be applied, depending on the feature fusion method employed. Since, This limitation of most literature methods influenced our choice by using only features extracted from the teacher's speech and thus to cover a larger number of video lectures. Another important aspect is that, unlike most approaches that only take into account the syntactic characteristics of the extracted texts, our method uses the Word2Vec model to represent the textual features and extract relationships that would not be possible through mere syntactic comparisons. The use of prosodic features is also a crucial point of our approach since not all information contained in speech is verbal.

The choice of an optimization model is because we believe this type of approach is more general when it comes to finding solutions to an instance of the problem. Although, depending on the algorithm used for optimization, there may be a lot of parameter adjustment at the beginning. In contrast, threshold-based approaches tend not to generalize well when there is too much variability between instances.

To close this chapter, we chose the main works of this literature review to compare the techniques and features used by each of them:

- **Research 1**: Our Proposal
- **Research 2**: (GALANOPOULOS; MEZARIS, 2019)
- **Research 3**: (SOARES; BARRÉRE, 2018)
- **Research 4**: (BISWAS; GANDHI; DESHMUKH, 2015)
- **Research 5**: (LEE et al., 2017)

Table 1 shows a compiled comparison between our proposal and the researches above.

Table 1 – Comparison between techniques/features used by the main literature researches in temporal segmentation of video lectures

	Research 1	Research 2	Research 3	Research 4	Research 5
OCR				x	
ASR	x			x	
Manual Subtitles		x			
Word Embeddings	x	x			
Prosodic features	x		x	x	
Visual features				x	x
Optimization approach	x			x	
Clustering approach			x		
Threshold approach		x			x

Source: The author (2020)

4 PROPOSAL

As stated before, one of the main problems in the temporal segmentation of video lectures is the dependence on the existence of additional manually generated materials (e.g. slides, textbooks, subtitles, among others) to be used as input for algorithms. Also, even those that do not depend on external material for segmentation are very specific to certain types of video lectures. For this reason, in this work, we propose a method for temporal segmentation of video lectures that uses only features obtained automatically from speech. Our proposal, in general, can be seen as a complete processing pipeline ranging from feature extraction to the segmentation algorithm itself.

In our pipeline, the video lecture audio track goes through a series of processing to extract both semantic and prosodic features from the teacher's speech. We then use these features to model the temporal segmentation of video lectures as an LP problem that we optimize to find a solution. For a better understanding of the overview of our proposal, we break it down into 4 processing stages where each one receives input and produces an output. We briefly describe each one below:

- **(i)** All the information we need is present in the teacher's speech. Therefore, in the first stage of processing, we extract the audio track from the video lecture and provide input for the next step.
- **(ii)** Although the audio track has all the useful information for our algorithm, it also has a lot of irrelevant information, such as background noise. To alleviate this problem, we have applied a process we call silence removal so that we can have audio chunks that contain as much continuous foreground speech as possible.
- **(iii)** At this stage, we extract semantic and prosodic features from the audio chunks obtained from the silence removal stage.
- **(iv)** We then use the extracted features to compose an LP model that we optimize with a metaheuristic to obtain a temporal segmentation for the video lecture.

With this processing pipeline, we can obtain a temporal segmentation of video lectures through only features of their audio track. Furthermore, all stages of this pipeline are fully automated. Figure 6 illustrates the overview of our proposal.

4.1 AUDIO EXTRACTION

The audio of video lectures is a major source of information. Since in most of them, the teacher's speech is present and carries a lot of meaning related to the video lecture content (HUSAIN; MEENA, 2019; LIN et al., 2005b).

In our proposal, we extract different features from the audio track to perform segmentation. Thus, audio extraction is the first and most fundamental step. Because of this, we must ensure that the extracted audio is compatible with the other processing steps. Therefore, we standardize the extracted audio format as single-channel audio using 16-bit uncompressed PCM encoding (GOODMAN, 1969) and 16 kHz sampling rate.

As previously stated, the decision to choose this audio format was based on technical criteria to fully meet the needs of the following steps of our proposal. Since our purpose is to capture features from the teacher's speech, we need to choose the minimum audio settings that allow us to achieve this fully. Therefore, we chose the sample rate and the number of representation bits according to the literature recommendation for wideband speech applications (CHRISTENSEN, 2019). In this way, we guarantee that we can extract all features satisfactorily while saving computational resources.

4.2 SILENCE REMOVAL

Silence removal is applied to the extracted audio track and is responsible for separating useful information for our framework (foreground teacher speech) from useless information (background noise only or silence). To do this, we use a VAD algorithm to classify audio frames as **voiced**/**unvoiced**. This VAD model receives a threshold parameter that determines the aggressiveness of the algorithm to classify an audio frame as unvoiced (we will detail parameter values in the Experimentation chapter). With a less aggressive threshold, the VAD algorithm will be more likely to classify frames that contain hesitation pauses after expressions like "Um", "Er", "Uh" and "So" as **voiced**, for example. The opposite happens with a more aggressive threshold. Figure 7 illustrates how some audio frames that were classified as voiced become classified as unvoiced if we set a more aggressive threshold.

After applying VAD, we can discriminate which audio frames have useful information for our algorithm and which do not. We then put together consecutive audio frames that were classified as voiced to form what we call audio chunks. A more formal definition of an audio chunk can be given as follows: let $q(t)$ be a sequence of audio frames in time, and suppose that each audio frame in $q(t)$ was classified as **voiced** or **unvoiced**. An arbitrary audio chunk s_i can be defined as a sub-sequence of $q(t)$ represented by the open intervals (b, e) where $q(b)$ and $q(e)$ are both audio frames classified as unvoiced. Figure 8 illustrates the concept of audio chunk.

At the end of this stage, we ensure that we have audio chunks fully composed by the teacher's speech, which is important for our feature extraction since VAD is a proven effective preprocessing for noise removal which may reduce the error rate in several applications (PASAD; SABU; RAO, 2017).

4.3 OBTAINING PROSODIC FEATURES

Not all relevant information in the teacher's speech is present in the form of words. As stated earlier, it is common for people to demarcate logical units of speech nonverbally (prosodically). Hence, prosodic features are capable of giving us additional information about speech structure, which can be very useful to achieve a more accurate temporal segmentation to video lectures. Some specific prosodic features, such as **pitch**, **pause duration**, and **loudness**, have been successfully applied over the years to perform the speech segmentation task (CHE; YANG; MEINEL, 2018; MAYER; JASINSKAJA; KÖLSCH, 2006; HIRSCHBERG; LITMAN; SWERTS, 2004; SHRIBERG et al., 2000; OLIVEIRA, 2000; KOCHANSKI et al., 2005; ARONS, 1994). Because of this, we have incorporated these prosodic features into our framework to take advantage of the valuable information they can provide to solve our problem.

Therefore, at this stage, we extract the pitch, pause duration, and loudness estimations for each audio chunk $s_i, i = 1, ..., C$, generated by the silence removal preprocessing. We represent then the prosodic features of an audio chunk s_i as a tuple $(\rho_i, \tau_i, \omega_i)$, where ρ_i, τ_i and ω_i are the pitch, pause duration and loudness estimations, respectively.

To obtain the pitch value (ρ_i) for an audio chunk s_i, we first apply the YIN method (CHEVEIGNÉ; KAWAHARA, 2002) to obtain the pitch contour of the speech in s_i. As result of this, we obtain a N dimensional array F_i that contains the fundamental frequencies of the signal. We then calculate the average of F_i values to get ρ_i:

$$\rho_i = \frac{1}{N} \sum_{j=1}^{N} F_{ij} \tag{4.1}$$

This average value can provide a good representation of the feature pitch for our model, since the larger the range of fundamental frequency values in the audio chunk, the higher it is the average.

The pause duration is another prosodic feature that may provide strong cues about topic transitions (CHE; YANG; MEINEL, 2018). In our proposal, we assume the pause duration of an audio chunk s_i as the total consecutive silence time detected before the beginning of s_i. In other words, suppose z is the index of the first audio frame of a voiced audio chunk s_i and w the index of the first audio frame of the voiced audio chunk before s_i. Suppose also that the function $d(x)$ gives the duration in seconds of arbitrary audio frame x. So, τ_i is calculated through the Equation 4.2 below:

$$\tau_i = \sum_{k=w+1}^{z-1} d(k) \tag{4.2}$$

Where the audio frames in the interval $[w + 1, z - 1]$ were classified as unvoiced by

the VAD algorithm from the previous section.

As stated in the fundamentals section, loudness is an important feature used to detect highlights in speech because it is associated with the energy contained in the speech signal. Hence, it is possible to have a good estimation of the instantaneous loudness values in speech signal by calculating the energy in short segments, such as audio frames. One of the ways to calculate this is through the Short Term Energy (STE) equation (NANDHINI; SHENBAGAVALLI, 2014), which we use to estimate the loudness for each audio chunk.

Let $\xi_{ji}(m), m = 1, ..., W_L$ be the discrete speech signal of the j^{th} frame of an audio chunk s_i, where W_L is the frame length. First, for each frame of s_i, we calculate the STE using the equation:

$$e(j) = \frac{1}{W_L} \sum_{m=1}^{W_L} |\xi_{ji}(m)|^2 \tag{4.3}$$

In Equation 4.3, $e(j)$ is the STE for a audio frame j and $\frac{1}{W_L}$ is a normalization factor inserted to remove the dependency of frame length (GIANNAKOPOULOS; PIKRAKIS, 2014).

Lastly, to obtain our loudness feature (ω_i) for an audio chunk s_i, we calculate the average of STE values of its frames:

$$\omega_i = \frac{1}{N} \sum_{j=1}^{N} e(j) \tag{4.4}$$

Where N is the number of frames in the audio chunk s_i.

4.4 OBTAINING SEMANTIC FEATURES

The words and sentences spoken by the teacher are full of meaning about the content of a video lecture. Therefore, through them, we can get important information about the structure of the video lecture and thus improve the quality of our temporal segmentation. Although it may seem, it is not trivial to obtain or represent these words to capture that meaning.

As we did for prosodic features, we also obtain semantic features for each audio chunk s_i. To do this, we send each audio chunk s_i to a processing pipeline (see Figure 9) that outputs the representation of the semantic features of s_i:

1. First, we transcribe the speech for each audio chunk using ASR. Although ASR transcripts are not accurate as manual subtitles, not all video lectures have subtitles available. So, using ASR is a good alternative to work around this problem and automatically extract the textual content of the teacher's speech.

2. Next, for each audio chunk, we perform a POS (part-of-speech) tagging in its transcripts generated by the ASR. The POS tagging consists of labeling each word or token from a text according to its morphosyntactic function in a given context (MÀRQUEZ; RODRÍGUEZ, 1998). Usual labels are: **DT** (Determiner), **IN** (Preposition), **JJ** (Adjective), **NN** (Noun, singular), **NNS** (Noun, plural), **RB** (Adverb), **RP** (Particle), **VB** (Verb, base form), **VBD** (Verb, past form) and **VBN** (verb, past participle). With this, we can understand the role of each word in sentences and treat them differently according to their importance to speech semantics.

3. Not all words or phrases in audio chunk transcripts are meaningful to understand the video lecture content. For example, prepositions and particles are important for connecting words and expressions, but they do not have information about the content of the speech itself. This kind of word is called **stopword**. For this reason, we first remove all stopwords from the transcripts. Next, we get only the words that compose Noun Phrases (NP) and remove the others. NP is composed of a noun as its core plus words that modify it or give additional information about it, such as adjectives, among others. For example, in the sentence *"Now, we will talk about* ***artificial intelligence***", the part that appears in bold is a NP. The reason we do this is that studies have shown that using NP as textual features can perform better than others (GALANOPOULOS; MEZARIS, 2019; LIN et al., 2005a).

4. Traditional methods for text representation, such as BoW models, are based on word occurrence statistics and hence only represent the syntactic nature of text documents, failing to capture the semantics behind them. For this reason, at the end of all text processing, we represent the processed transcript of each audio chunk using a Word2Vec model because of its properties already discussed. Since in Word2Vec each word is represented as a vector, the semantic features of each audio chunk is given by a matrix NxM, where N is the dimension of the word vector representation and M the number of words in the resulting text.

4.5 LP MODEL

After we extract both prosodic and semantic features from each audio chunk, we model the problem of temporal segmentation of video lectures in the function of these features. We choose to model our problem as an LP problem due to the fact that there are several efficient algorithms and solvers for this kind of problem. In our model, we map the temporal segmentation problem into the problem of selecting the audio chunks that are more likely to contain a topic boundary. Since audio chunks follow temporal order, we can map it back to the original problem.

Our LP model has originally two objectives: (1) maximize the sum of utility score of audio chunks chosen as topic boundary, and (2) minimize the number of partitions (topics). With this, we want to select the most prominent audio chunks while avoid under/over-segmentation, which would have the opposite effect of a good temporal segmentation. The utility score u_i of an audio chunk s_i is given by Equation 4.5 and is a measure of how likely an audio chunk s_i contains a topic transition, where ρ_i, τ_i, ω_i are pitch, pause duration and loudness estimates, respectively. The term Δ_i represents the difference of distances (**two distance measures will be evaluated in Experiments section**) between the semantic features of the audio chunk s_i and of its two immediate neighbors in time s_{i-1} and s_{i+1}, as shown in Equation 4.6.

$$u_i = C_0 \cdot \rho_i + C_1 \cdot \tau_i + C_2 \cdot \left(1 + \frac{i-1}{S_L - 1}\right) \cdot \omega_i + C_3 \cdot \Delta_i \tag{4.5}$$

$$\Delta_i = dist(i-1, i) - dist(i, i+1) \tag{4.6}$$

The constants C_0, C_1, C_2, and C_3 are inserted to scale the features, since their values vary on different ranges. Also, as we calculate the loudness feature using an energy measure, we incorporate the term $\left(1 + \frac{i-1}{S_L-1}\right)$ in Equation 4.5, where S_L is the total number of audio chunks, to compensate for the fact that the speaker gets tired throughout the lecture and consequently the energy level in the speech also decreases (CHE; YANG; MEINEL, 2018). Furthermore, the intuition behind the term Δ_i given by the Equation 4.6 is that we believe that, due to the temporal characteristic of a topic, the audio chunks with more chance to be a topic boundary tend have much greater semantic distances with the previous chunk $(i-1)$ than with the next one $(i+1)$.

With the utility function and its components defined, we can mathematically write our LP formulation as follow:

$$\begin{aligned}
&\text{maximize} && \sum_{i=1}^{S_L} u_i x_i && (1)\\
&\text{minimize} && \sum_{i=1}^{S_L} x_i && (2)\\
&\text{subject to} && && \\
& && \sum_{i=1}^{S_L} x_i > 0 && \\
& && x_i \in \{0, 1\}, i = 1, ..., S_L &&
\end{aligned}$$

Where the variables $x_i, i = 1, 2, 3, ..., S_L$, are binary decision variables that indicates whether an audio chunk s_i has been selected (1) or not (0) as topic boundary. The only restriction we impose on our model is that there must be at least one partition in the

solution for it to be feasible. As we state in the Fundamentals chapter, the biggest challenge in multi-objective problems is to find the optimal solutions for all objective functions simultaneously. Thus, instead of dealing with two separate objective functions, we rewrite our formulation with a single objective maximization function. We do this by subtracting the objective function (2) from (1):

$$\begin{aligned} \text{maximize} \quad & \kappa \sum_{i=1}^{S_L} u_i x_i - (1-\kappa) \sum_{i=1}^{S_L} x_i \\ \text{subject to} \quad & \\ & \sum_{i=1}^{S_L} x_i > 0 \\ & x_i \in \{0,1\}, i = 1, ..., S_L \end{aligned}$$

In this final formulation, the constant κ, $0 < \kappa \leq 1$, controls the importance of each objective to the problem. For example, if $\kappa = 0.5$ both objective functions will be equally important to the problem. If instead, the value of κ is too large ($\kappa = 0.9$), this means that the objective function (1) will be of much greater importance than (2). As a result, the best solutions to our problem will tend to have many partitions (over-segmentation). And, the opposite will happen if the value of κ is too small (under-segmentation). Figure 10 shows a geometric interpretation of the value of κ in our objective function. In this Figure, we show the trend lines of both objective functions.

4.6 OPTIMIZATION ALGORITHM

Because our model may have thousands of variables, depending on how many audio chunks we found for a video lecture, we chose to approach the problem with a heuristic method instead of an exact one. In our proposal, we use a GA to find solutions that optimize our LP model. This choice is because the GA is one of the most successfully used metaheuristics by researchers for solving linear optimization problems (CAMPOS; JIMÉNEZ-BELLO; ALZAMORA, 2020; RASHEED, 2019; LEE; YANG, 1998; DAVIS, 1985).

In our GA, we randomly build an initial population of P size and represent each individual as a binary chromosome array, where the gene in the i^{th} position corresponds to the decision variable x_i of our LP formulation. Besides, we use the modified objective function presented in the previous section as our fitness function. That is, we use the objective function of the problem itself to evaluate how good each individual is for solving it. Furthermore, each individual of the population has a probability m of suffering mutation, which we defined as a bit-flipping of a random gene in its chromosome. That is, if the gene has the value 1, it becomes 0 and vice-versa.

Unlike standard GA, we employ elitism in selecting individuals who will be parents

and survive for the next generation (SOREMEKUN et al., 2001). In other words, we select the best E individuals (elite) to generate offspring and survive along with them for the next generation while other individuals die. In the crossover stage, we select with equal probability pairs of elite individuals to generate new individuals by combining their chromosomes. These chromosomes are combined using a two-point crossover operator (MAN; TANG; KWONG, 1996) (see subsection 2.1.3). In our case, the number of performed crossovers is equal to the number of individuals who died since new individuals will replace them in the next generation. Therefore, we define as crossover rate (C), the number of performed crossovers over the population size:

$$C = \frac{P - E}{P} \tag{4.7}$$

Also, to improve the solutions found, we perform a local search on the top-10% best individuals. As this process is computationally costly, it is applied only in the most prominent individuals. To perform the local search, we use the Tabu Search (TS) algorithm (GLOVER; LAGUNA, 1998). In TS, each movement $\eta_i, i = 1, 2, 3, ..., M$, defines a neighborhood, so the algorithm starts from an initial solution Q_0 and finds the best neighbor $Q_i = Q_0 \bigoplus \eta_i$ such that the movement η_i is not in the Tabu List. After, we add η_i in the Tabu List and, if Q_i is better than Q_0, assign Q_i to Q_0. We repeat this process until a stopping criterion is reached. The Tabu List is a data structure used to avoid the algorithm to be trapped in local optima by forbidding some movements that have been applied in the last k iterations (EDELKAMP; SCHROEDL, 2011). In our case, we define k equals 2 and as stop criterion 10 iterations without improvement or 50 iterations in total. We chose these values so that they are sufficient for local search to find improvements in the neighborhood of the solutions, without using much computational effort in the attempt. Besides, we define three movements: **Merge**, **Split** and **Move Bound**. In **Merge**, two adjacent topics are merge into one. The **Split** movement is the opposite of merge: a topic is divided into two new topics. Finally, in **Move Bound** movement, a topic boundary is moved to another audio chunk. We use these local search movements to try to correct the main errors existing when defining the limits of topics in a video lecture. Figure 11 illustrates examples of these movements and the Algorithm 2 describes the TS used in this work.

Algorithm 2: Tabu Search

Data: Q_0, $f_{obj}(.)$
Result: Q^*

```
T := ∅;
iter := 0;
Q* := Q0;
while stop criterion not true do
    η* = η;
    (Qi, ηi) := best_neighbor(η*);
    while ηi ∈ T do
        η* = η* \ {ηi};
        (Qi, ηi) := best_neighbor(η*);
    end
    T := T ∪ {ηi};
    if length(T) > 2 then
        T := remove_first_element(T)
    end
    if fobj(Qi) > fobj(Q*) then
        Q* := Qi
    end
    iter := iter + 1
end
return Q*
```

To summarize, in our proposal, we use an elitist GA with local search as an optimization method to find solutions that maximize our objective function (see Algorithm 3). Each decision variable of our problem, represented in the GA by chromosome genes, is related to an audio chunk, and each audio chunk has a timestamp of when it begins in the video lecture. Thus, we can map the solutions found by GA directly to the original problem by just getting the beginning timestamps of the audio chunks selected as topic boundaries. Figure 12 shows an example of possible solution found by the GA. In this case, there are two topics delimited by the audio chunks s_i and s_{i+4}. Suppose we have a function $b(.)$ that outputs the time of beginning of any audio chunk s_i, hence we can

represent the topic j by the interval $[b(s_i), b(s_{i+4}))$ and the topic $j+1$ by $[b(s_{i+4}), b(s_n))$

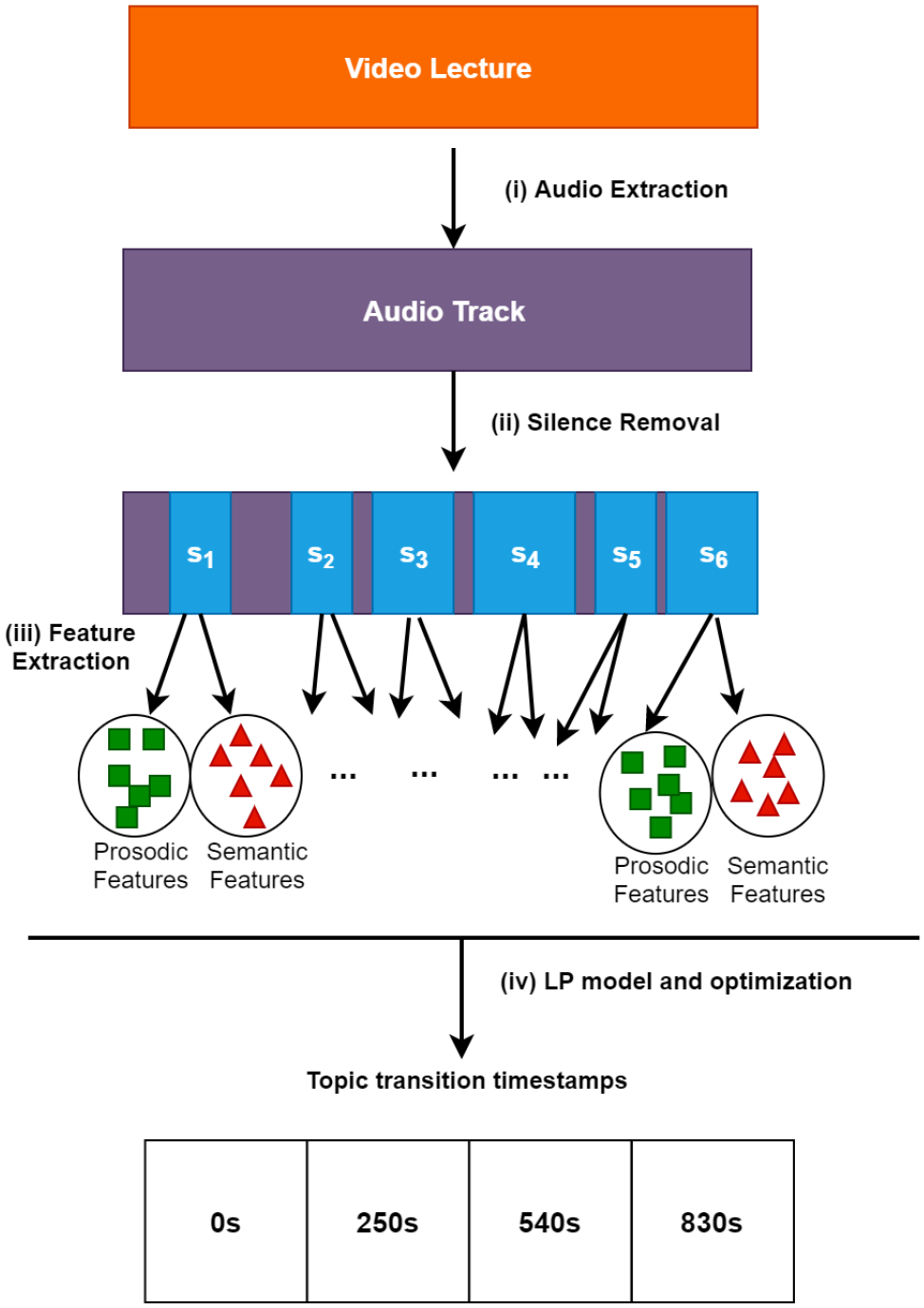
Algorithm 3: Elitist Genetic Algorithm with Local Search

Data: population size, number of generations, crossover rate, mutation rate

Result: The individual with best fitness value

```
set the population size ||P||;
set the number of generations G;
set the crossover rate C;
set the mutation rate M;
randomly initialize the population P;
evaluate(P);
iterations := 0;
while iterations < G do
    P̄ := select_parents_with_elitism(1-C);
    recombine(P̄, C);
    mutate (P̄, M);
    evaluate (P̄);
    top_ten_percent := select_top(P̄);
    local_search(top_ten_percent);
    select_survivors_with_elitism(P̄, P, C)
    iterations := iterations + 1
end
return the best individual
```

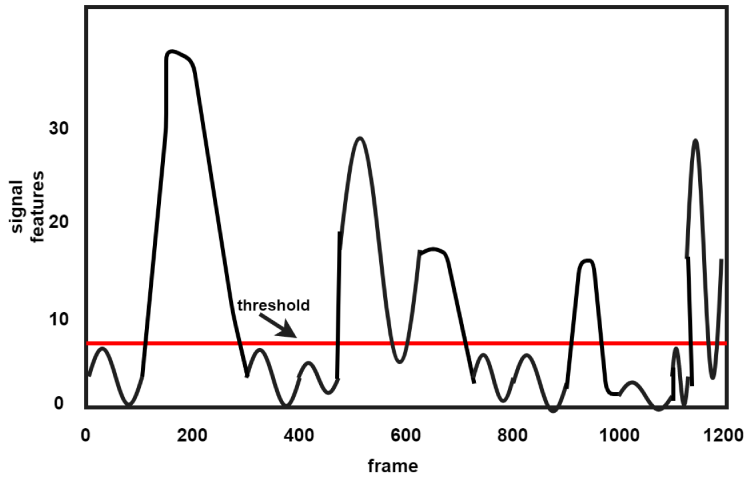
Figure 6 – Overview of the processing pipeline of our framework. (i) The audio track is extracted from the video lecture. (ii) A silence removal stage is performed. (iii) Prosodic and semantic features are extracted from the teacher's speech. (iv) Optimization of a LP model to obtain the temporal segmentation



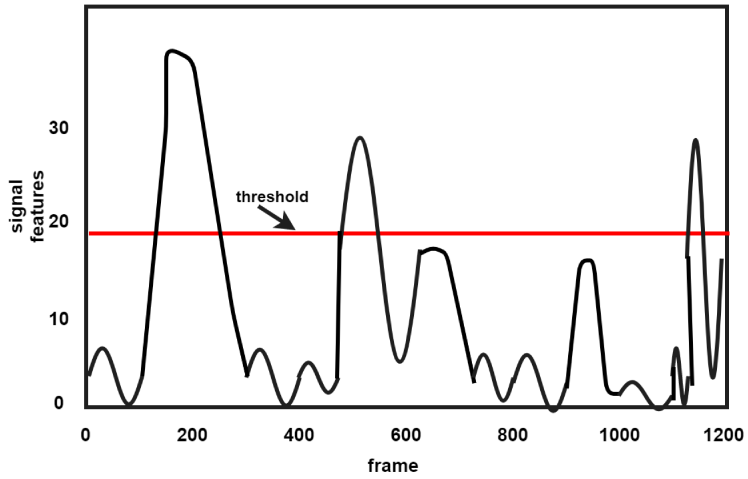
Source: Created by the author (2020)

Figure 7 – Voice activity detection applied to audio frames. (a) Less aggressive threshold. In this case, the minimum signal value to be classified as a voiced frame is low. (b) More aggressive threshold. In this case, the minimum signal value to be classified as a voiced frame is higher.

(a) Less aggressive threshold

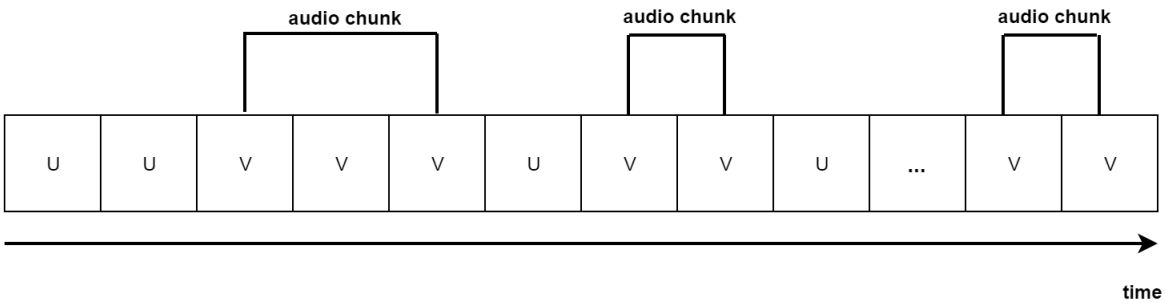


(b) More aggressive threshold



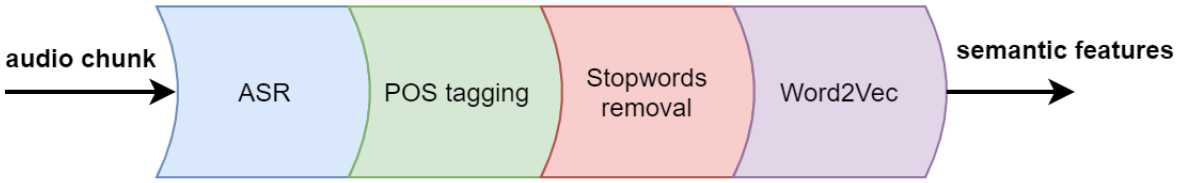
Source: Created by the author (2020)

Figure 8 – Example of audio chunks composed of sequential audio frames classified as voiced



Source: Created by the author (2020)

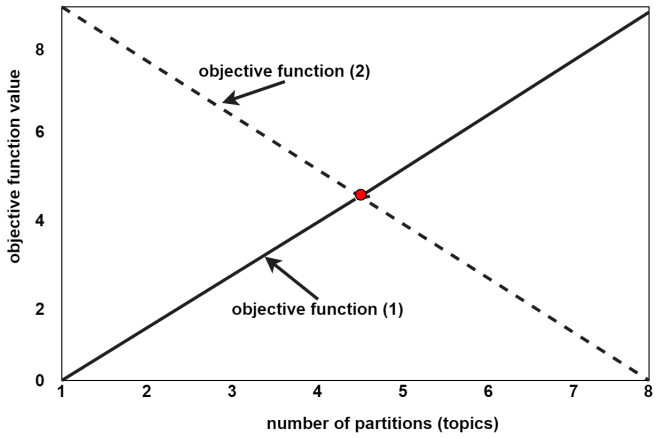
Figure 9 – Text processing pipeline. Used to obtain the semantic features for each audio chunk. First, we transcribe the speech through ASR. Next, we perform POS tagging in speech transcripts. Then, the stopwords are removed from the text. Finally, the text is represented in a vector space using a Word2Vec model.



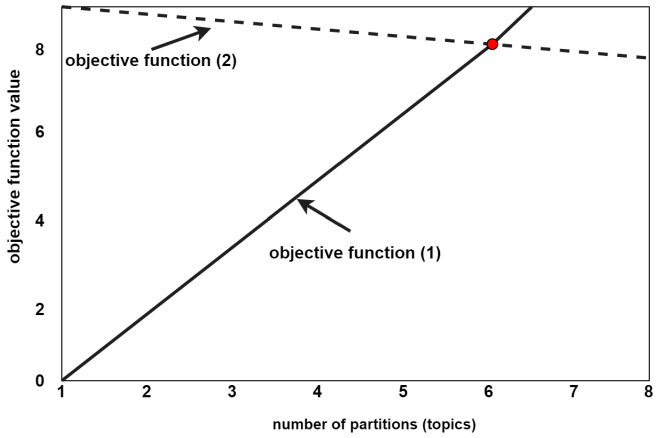
Source: Created by the author (2020)

Figure 10 – Geometric interpretation of the value of κ. The red dot represents the point where the value of the object function (1) is equal to the value of the object function (2). (a) The value of κ is 0.5. Both objective functions have very similar increase/decrease rates. (b) The value of κ is 0.9. The objective function (1) increases at a much higher rate than the rate at which (2) decreases. Therefore, the solutions found tend to have more partitions. (c) The value of κ is 0.1. The opposite of case (b) will happen.

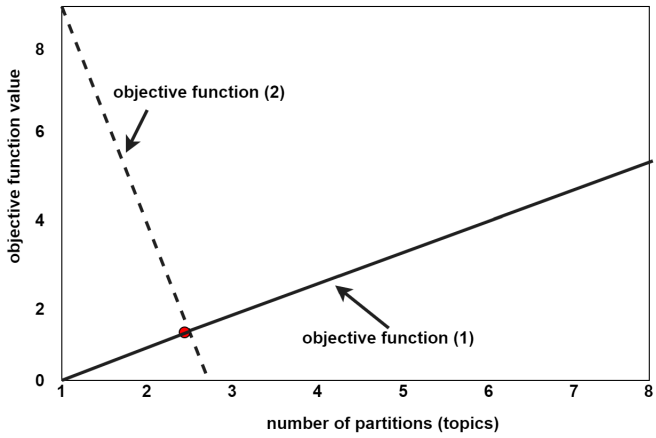
(a) $\kappa = 0.5$



(b) $\kappa = 0.9$

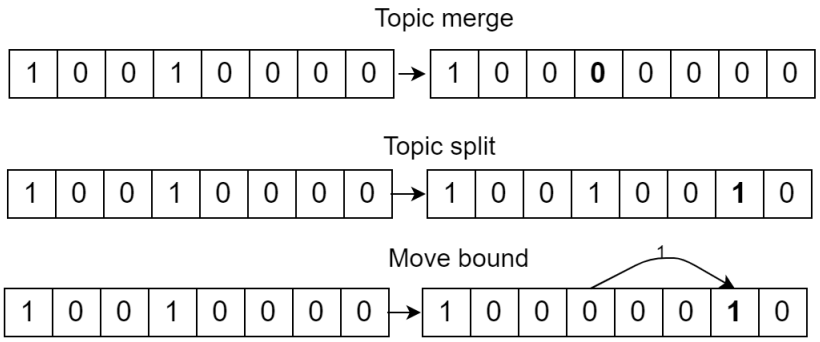


(c) $\kappa = 0.1$



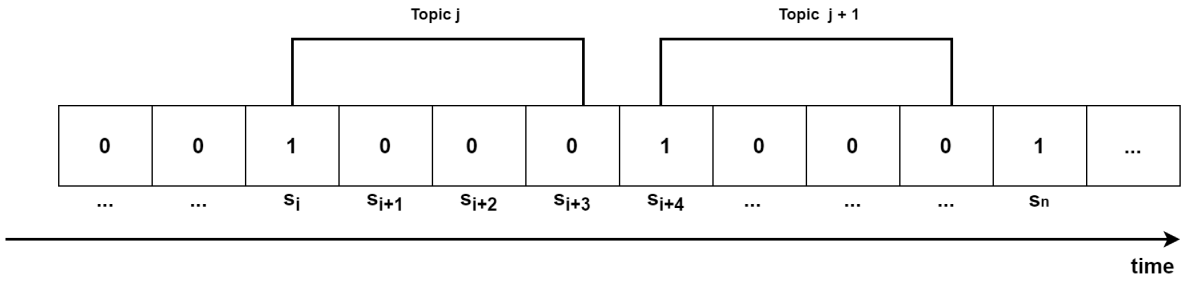
Source: Created by the author (2020)

Figure 11 – Local search movements. Each movement defines a neighborhood for an initial solution. Local search movements. Each move defines a neighborhood for an initial solution. The merge movement consists of joining two consecutive topics into one. On the other hand, split movement separates a topic into two consecutive topics. Lastly, move bound shifts the boundaries of one topic from one audio chunk to another.



Source: Created by the author (2020)

Figure 12 – This Figure illustrates how the solutions found by GA are related to the temporal segmentation of video lectures. The topic j starts in the audio chunk s_i and ends in the audio chunk s_{i+3}. In turn, topic $j+1$ starts in the audio chunk s_{i+4} and ends in the audio chunk s_n.



Source: Created by the author (2020)

5 EXPERIMENTAL RESULTS

In this chapter, we evaluate our proposal through experiments that explore different aspects of it. In the first experiment, we explored different GA parameter values for 5 versions of our objective function. By doing so, we want to show that, with the best set of parameters, the combination of prosodic and semantic features makes our method perform better than using them separately. Next, we analyze the convergence behavior of our algorithm over iterations and discuss the advantages and disadvantages of using local search in this case. Lastly, we compare the performance of our approach with two other literature proposals: a fully text-based and a multimodal algorithm that relies on features extracted from slideshows. With these comparisons, we can give evidence that our approach provides good segmentation in two distinct scenarios even when compared to specifically designed approaches.

Due to the stochastic nature of our method, each experiment was performed 10 times for each of the 5 random seeds used. Therefore, we present the results as the mean and standard deviation of the evaluation measures over these runs. For this, we used a machine with 8GB of RAM and an Intel® Core™ i5-8265U processor with 6MB of cache and a maximum frequency of 3.9 GHz to perform all experiments presented in this chapter.

5.1 IMPLEMENTATION DETAILS

We implemented our framework proposed in the previous chapter as a distributed software architecture. In this architecture, each processing stage of our framework is a fully independent module that works in a producer/consumer schema (ZHANG; ZHANG; ZHANG, 2009). Each process that performs the tasks of a module is a worker that consumes messages from a task queue and, at the end of processing, inserts the results into an output queue (that can be a task queue of another module). The main advantage of using this design pattern is that we can more easily manage processes that produce and consume data asynchronously at different speeds.

We mainly use the Python programming language to implement our architecture. For the implementation of producer/consumer queues, we use the message broker RabbitMQ[1]. We also use the databases PostgreSQL[2] and MongoDB[3] to store data. In PostgreSQL, we store processing metadata to enables us to track the steps of a segmentation request from a client and, thus, detect possible bugs. Also, these metadata can be used to keep a history of processing for further analysis. In turn, we use MongoDB's GridFS to store and retrieve video, audio and text files generated by the processing performed by

[1] https://www.rabbitmq.com/
[2] https://www.postgresql.org/
[3] https://www.mongodb.com/

workers.

Regarding architecture modules, they are all shipped in Docker[4] containers to facilitate the reproducibility and deployment of our proposal. In total, our architecture has 7 modules:

- **REST API**: The entry point of our architecture. Responsible for receiving from users the video lecture to be segmented through HTTP POST. After receiving the segmentation request, the REST API module stores the video in MongoDB and inserts a request message in the Audio Extractor module task queue.

- **Audio Extractor**: Module that extracts audio from the incoming video lecture. To perform the extraction, we use the multimedia framework FFmpeg[5]. At the end of the extraction, the Audio Extractor stores the video lecture audio in the MongoDB. Also, it inserts a request messages in the task queue of the VAD module.

- **VAD**: Performs the voice activity detection in the extracted audio file to remove silence and noise. To do this, we use a Python implementation available on GitHub[6] of the VAD algorithm of Google's WebRTC (GOOGLE, 2020). This implementation allows us to define a level of aggressiveness to be used by the algorithm to detect the absence of speech. These levels range from 0 (less aggressive) to 3 (more aggressive). In the implementation carried out in this work, we chose empirically the value 1. At the end of the processing, the VAD stores the voiced audio chunks generated in the database. Also, the VAD inserts request messages in the task queues of the ASR and Prosodic Extractor modules.

- **ASR**: Automatically transcribes the speech contained in each chunk audio generated by VAD. To build this module, we use the toolkit Kaldi (POVEY et al., 2011) with the ASpiRE Chain Model[7] for English. We chose this specific model due to its low word error rate (WER) of 15.60%. The ASR module generates and stores a text file in the database for each transcribed audio chunk. After that, it produces a request message in the Flow Aggregator task queue.

- **Prosodic Extractor**: It is responsible for extracting the prosodic features of each chunk audio. To obtain the pitch contour, we use the implementation of the YIN method from the Python library aubio[8]. To extract the other features, we use only standard Python. Like the ASR module, this module stores the result of its

[4] https://www.docker.com/
[5] https://www.ffmpeg.org/
[6] https://github.com/wiseman/py-webrtcvad
[7] https://kaldi-asr.org/models/m1
[8] https://aubio.org/

processing in the database and inserts a request message in the Flow Aggregator task queue.

- **Flow Aggregator**: As the segmentation algorithm requires both semantic and prosodic features that are processed asynchronously by their respective modules, there must be a module responsible for aggregating the two types of features extracted from the same video lecture. When a message arrives at its task queue (from the ASR or Prosodic Extractor module), the Flow Aggregator module starts a worker to wait for the missing features to finish processing and then aggregate them in a single message to dispatch it to the segmentation algorithm. The Flow Aggregator module identifies messages from the processing of a video lecture using a unique ID generated for each video lecture given as input by the user.

- **Segmentation**: Implements our segmentation algorithm (elitist GA with local search) and some other processes like the POS Tagging, stopword removal and the Word2Vec representation of audio transcripts. To perform the POS Tagging and stopword removal, we use the NLTK Toolkit[9]. Besides that, to represent the audio transcripts, we use the Google's Word2Vec model[10]. This model was trained on Google News dataset with about 100 billion words and represents each of them in a vector space of 300 dimensions using the approach from (MIKOLOV et al., 2013b). Regarding the parameters of our LP model, we choose the values of the constants C_0, C_1, C_2 and C_3 from Equation 4.5 such as 0.01, 1, 0.1 and 10, respectively. In this way, we scale all features to vary on the same scale and thus have the same importance. Also, we set the value of κ of our objective function to 0.4, since, analytically, it is a reasonable value to avoid over-segmentation, but without the opposite occurring.

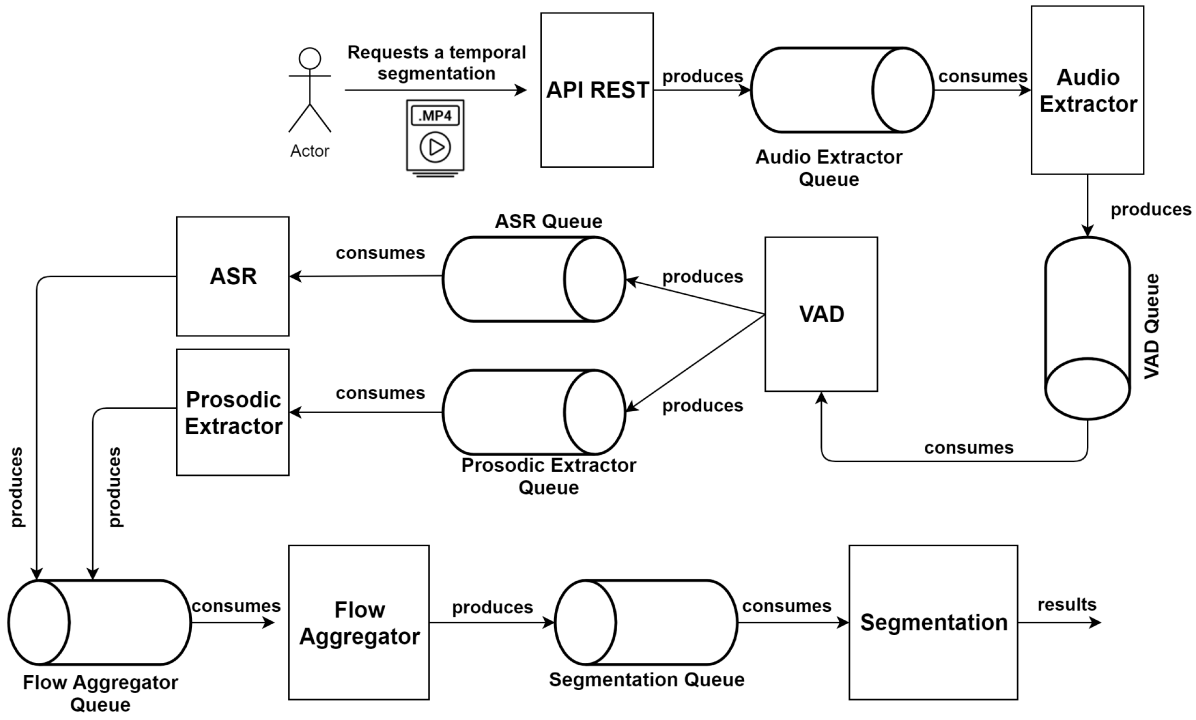
Figure 13 illustrates the queuing communication scheme between the architecture modules. Furthermore, we make all the code of our implementation available on GitHub[11] for all those interested in contributing or reproducing the results reported in this work.

5.2 DATASETS

To evaluate our proposal and provide experimental evidence on the quality of our proposal, we carried out experiments on two datasets of video lectures with different characteristics. The first dataset, which we call "**audio-based**", consists of video lectures in English about exact science subjects. In these video lectures, a teacher speaks from a stage to an audience without any visible slide show or other text on the screen. Thus, they are video lectures where the information is predominantly in the audio, more specifically in the

[9] https://www.nltk.org/
[10] https://code.google.com/archive/p/word2vec/
[11] https://github.com/eduardorochasoares/easytopic

Figure 13 – Communication schema of our architecture. This Figure shows how producer/consumer design pattern is used to implement the processing pipeline of our proposal.



Source: Created by the author (2020)

teacher's speech. We extracted these video lectures from the website **videolectures.net** where they are available for free. On the other hand, the second dataset consists of video lectures in English from the ACM's (Association for Computing Machinery) YouTube channel[12]. In these video lectures, a full-screen slideshow is displayed while a background speech explains the content, so we call it the "**slide-based**" dataset. Also, they were recorded in much more controlled and noise-free environments than those from the first dataset.

Each video lecture from these two datasets has a temporal ground-truth segmentation (see Figure 14), which we use to estimate how far the solutions found by our method is from the ideal solution. In the case of the audio-based data set, ground-truth segmentation is available at the site where video lectures are from. For the slide-based data set, we used the ground-truth segmentation available at the paper of (MAHAPATRA; MARIAPPAN; RAJAN, 2018). The two data sets have 40 video lectures each, and we make them available at the URL [13]. Figures 15 and 16 illustrate examples of video lectures from both datasets.

.

[12] https://www.youtube.com/user/TheOfficialACM
[13] https://drive.google.com/drive/u/2/folders/18wg0hh8ZJWNaDK_qlk3Vy4_quEl1pUqJ

Figure 14 – Example of JSON containing a ground-truth segmentation. The JSON keys represent the start times of the topics of the video lecture while the values represent their respective titles.

```
{
    226: 'ACM Highlights',
    274: 'Generator Power',
    311: 'Iteration',
    611: 'Iterables',
    945:  'Reduction Functions',
    1005: 'Sort x Sorted',
    1257: 'The Iterator Pattern',
    1764: 'Generator Function',
    2283: 'Bult-in Generators',
    2525: 'GENEXPS',
    2701: 'Case Study',
    3007: 'Subjects for another day...',
    3113: 'ThoughtWorks'
}
```

Source: The author (2020)

5.3 EVALUATION MEASURES

To measure the quality of the proposal, we compare the temporal segmentation obtained by our algorithm with the ground-truth segmentation provided for each video lecture. Thus, it is possible to estimate how far the results are from the ideal. We choose three measures to perform this evaluation: **Precision**, **Recall** and **F1 Score** (GOUTTE; GAUSSIER, 2005). Through them, we can analyze many aspects of the solutions for temporal segmentation problem, such as how accurate the segmentation is and how comprehensive it is according to topic numbers.

We define Precision as the number of algorithm hits over the number of hits plus the number of misses. Precision gives us a good idea of the algorithm's assertiveness when determining the existence of a topic transition at a given moment in the video lecture. Recall, instead, is the number of algorithm hits divided by the number of topics in the ground-truth segmentation. It provides a measure of how well covered the existing topics in the ground-truth were by the algorithm. Lastly, we use the F1 Score, which is defined as the harmonic mean between Precision and Recall, to get a single evaluation measure for an algorithm. In this work, based on other literature researches (MAHAPATRA; MARIAPPAN; RAJAN, 2018; BISWAS; GANDHI; DESHMUKH, 2015), we assume that a hit occurs when the start time of a topic (t_i) from the algorithm's solution differs at

Figure 15 – Example of video lecture from the audio-based dataset. A teacher speaks from a stage to an audience.

Source: http://www.videolecture.net (2019)

most 10 seconds of the start time (gt_i) of a ground-truth topic. Let T and GT be the algorithm and ground-truth solution sets for a given video lecture V, respectively, we define the evaluation measures by the Equations 5.1-5.3.

$$Hit(t_i, gt_i) = \begin{cases} 1, \ if \quad |t_i - gt_i| \leq 10 \\ 0, \ otherwise \end{cases}$$

$$Precision = \frac{\sum_{i=0}^{N} Hit(t_i, gt_i)}{|T|} \tag{5.1}$$

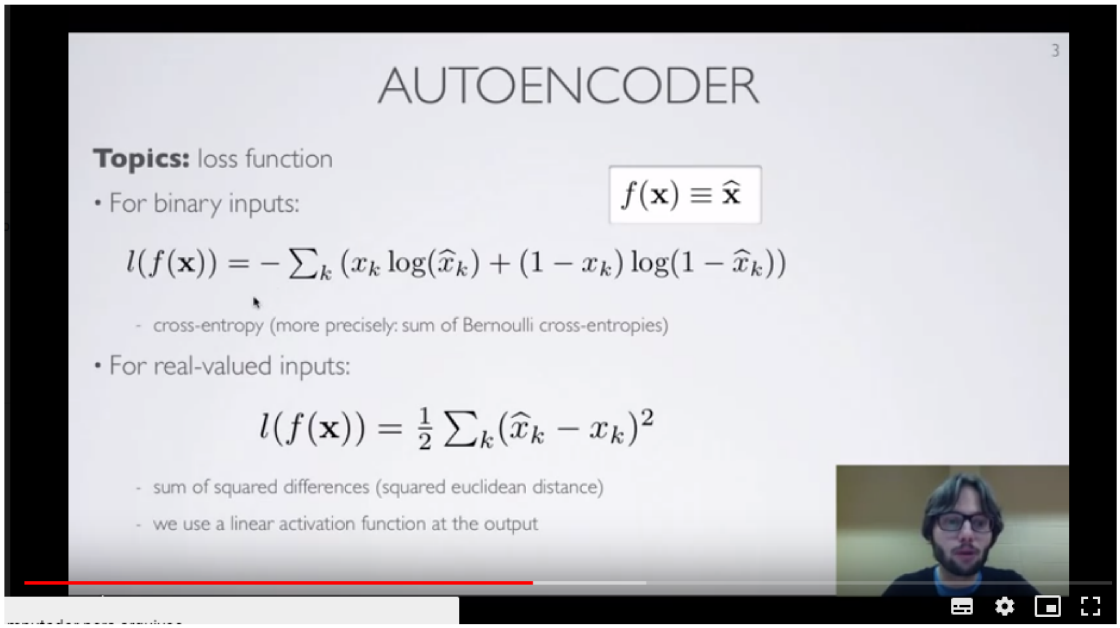
$$Recall = \frac{\sum_{i=0}^{N} Hit(t_i, gt_i)}{|GT|} \tag{5.2}$$

$$F1 \ Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5.3}$$

5.4 MODEL SELECTION

In this first experiment, we analyze and compare the performance of 5 different versions of objective functions for our model to determine which one is the best to approach our problem. These versions are:

Figure 16 – Example of video lecture from slide-based dataset. A slideshow is shown on the screen while the teacher explains the content.



Source: http://www.youtube.com (2019)

- **PROS**: An objective function that considers only the prosodic features in its formulation. We achieve this by setting the parameter C_3 of Equation 4.5 to 0.
- **COS**: An objective function that considers only the semantic features. We obtain it by setting the parameters C_0, C_1 and C_3 to 0. In this version, the textual distance metric used is the inverse of cosine similarity. For this, we need to represent the audio chunk semantic features as a single vector. We do this by calculating the average word vector (centroid) of the semantic matrix of each audio chunk.
- **WMD**: Similar to the COS version, but instead of using the cosine distance, it uses the Word Mover's distance. This distance metric works with multiple word vectors for a single document, so the centroid calculation is not needed in this case. But, differently of cosine distance, its values range in the interval of $[0, \infty)$. Therefore, we apply a min-max normalization (JAIN; RASTOGI et al., 2019), so that their values also vary from 0 to 1.
- **PROS + COS**: It considers both prosodic and semantic features using cosine distance as textual distance metric.
- **PROS + WMD**: Same as PROS + COS, but it uses Word Mover's distance instead of cosine distance.

To compare those versions, we first determine the best GA parameters for each of them on both datasets. To do this, we performed a Grid Search (GS) algorithm (LAVALLE;

BRANICKY; LINDEMANN, 2004). The GS is an exhaustive search algorithm that tests all possible combinations of parameters from a predefined subset of values. To guide the GS in the search for the best parameters, we used the metrics defined in Section 5.3. Then, we evaluated the effects of the variation in the mutation rate (m), crossover rate (c), population size (p) and number of generations (g) on the quality of the temporal segmentation of each version. The subsets of parameter values tested were:

- $m \in \{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07\}$
- $c \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
- $p \in \{50, 100, 200\}$
- $g \in \{50, 100, 200, 300, 400, 500\}$

We tested all possible combinations of parameters above to determine which one performs better in both datasets. Tables from 2 to 6 show the top-3 parameter combinations for each objective function and evaluation measure mean.

Table 2 – Results of hyperparameter tuning using Grid Search - **PROS**

	p	g	c	m	Precision	Recall	F1 Score
1	200	300	0.4	0.01	**0.51**	0.25	0.30
2	200	200	0.5	0.07	**0.51**	0.26	0.30
3	200	300	0.7	0.01	**0.51**	0.26	0.32
4	50	50	0.3	0.03	0.31	**0.57**	0.38
5	50	50	0.3	0.02	0.28	**0.55**	0.36
6	50	50	0.3	0.01	0.30	**0.53**	0.36
7	50	50	0.5	0.05	0.32	0.51	**0.38**
8	100	50	0.7	0.05	0.34	0.43	**0.38**
9	50	50	0.6	0.06	0.32	0.49	**0.38**

Source: The Author (2020)

It is possible to see in the GS results that each evaluated model has different characteristics. One point to highlight is that the versions with only semantic features (**COS** and **WMD**) obtained the most unbalanced results between Precision and Recall. While the COS version achieved the highest recall average of 86%, the WMD obtained 75% of Precision. However, these values are associate with low Precision (20%) and Recall (13%) values, respectively. Therefore, we can note that COS tends to over segmenting the video lectures and WMD does the opposite. The most plausible hypothesis to explain this is that the COS version uses the cosine distance between the average word vectors of the audio chunk transcripts, which leads to weaker semantic relations between them. Hence, the algorithm tends to segment too much the video lecture to optimize the objective

Table 3 – Results of hyperparameter tuning using Grid Search - **COS**

	p	g	c	m	Precision	Recall	F1 Score
1	50	100	0.9	0.03	**0.24**	0.72	0.34
2	50	500	0.5	0.02	**0.23**	0.72	0.34
3	50	400	0.7	0.01	**0.23**	0.69	0.32
4	200	300	0.9	0.06	0.20	**0.86**	0.32
5	200	500	0.9	0.06	0.21	**0.85**	0.32
6	200	300	0.8	0.07	0.21	**0.85**	0.32
7	100	50	0.8	0.01	0.22	0.77	**0.34**
8	50	200	0.6	0.06	0.23	0.75	**0.34**
9	50	500	0.8	0.03	0.23	0.71	**0.34**

Source: The Author (2020)

Table 4 – Results of hyperparameter tuning using Grid Search - **WMD**

	p	g	c	m	Precision	Recall	F1 Score
1	200	400	0.6	0.06	**0.75**	0.13	0.21
2	200	400	0.7	0.01	**0.67**	0.05	0.10
3	200	400	0.8	0.02	**0.67**	0.07	0.12
4	50	50	0.3	0.05	0.27	**0.39**	0.29
5	50	50	0.3	0.07	0.29	**0.39**	0.29
6	100	50	0.3	0.07	0.31	**0.39**	0.29
7	50	50	0.3	0.03	0.32	0.37	**0.31**
8	200	50	0.3	0.06	0.32	0.33	**0.30**
9	50	50	0.7	0.01	0.28	0.39	**0.30**

Source: The Author (2020)

function. And the opposite occurs with WMD, since it uses a more sophisticate algorithm to relate different word vectors.

Another point of attention is concerning the performance of objective functions that use prosodic features (**PROS**, **PROS + COS**, and **PROS + WMD**). One thing they all have in common is that they have more balanced results between Precision and Recall, which makes them able to obtain a higher F1 Score than the purely semantic versions. But, although they have a lot in common, we can note some differences. While **PROS** and **PROS + WMD** show a decrease of about 20% in maximum Precision when achieving maximum Recall, in **PROS**, this decrease is only 10%. Hence, as we can see in results, the **PROS + COS** obtains the best F1 Scores among all other objective functions by balancing the good Precision and Recall averages provided by **PROS** and **COS**, respectively.

Regarding the independent variation of the GA parameters, we show in Figure 17 how the F1 Score average and its standard deviation are affected by it. In this Figure, we can see that the versions **PROS**, **COS** and **WMD** need a small population size to obtain

Table 5 – Results of hyperparameter tuning using Grid Search - **PROS + COS**

	p	g	c	m	Precision	Recall	F1 Score
1	50	400	0.7	0.01	**0.50**	0.23	0.26
2	50	500	0.7	0.01	**0.50**	0.23	0.26
3	50	400	0.4	0.01	**0.50**	0.27	0.30
4	200	400	0.8	0.06	0.40	**0.53**	0.41
5	200	500	0.8	0.06	0.40	**0.53**	0.41
6	200	400	0.9	0.06	0.40	**0.52**	0.41
7	200	300	0.5	0.07	0.41	0.51	**0.42**
8	100	400	0.8	0.06	0.41	0.51	**0.42**
9	100	500	0.8	0.06	0.41	0.51	**0.42**

Source: The Author (2020)

Table 6 – Results of hyperparameter tuning using Grid Search - **PROS + WMD**

	p	g	c	m	Precision	Recall	F1 Score
1	50	500	0.7	0.01	**0.50**	0.23	0.27
2	50	400	0.5	0.01	**0.48**	0.23	0.28
3	50	300	0.6	0.01	**0.47**	0.23	0.29
4	100	50	0.3	0.01	0.28	**0.54**	0.34
5	50	50	0.3	0.02	0.29	**0.53**	0.35
6	100	50	0.3	0.05	0.29	**0.53**	0.36
7	100	400	0.4	0.04	0.44	0.31	**0.38**
8	100	50	0.6	0.07	0.33	0.47	**0.38**
9	50	50	0.7	0.02	0.34	0.47	**0.38**

Source: The Author (2020)

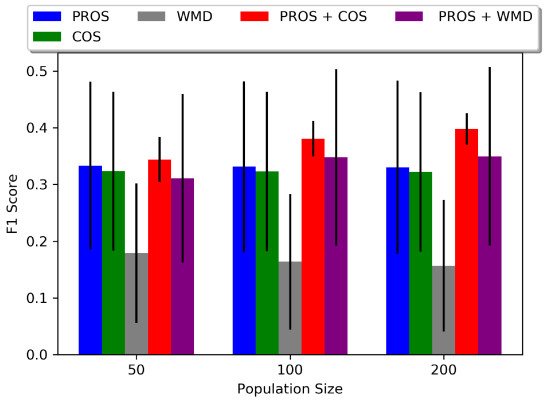
a higher F1 Score. On other hand, composite methods (**PROS + COS** and **PROS + WMD**) increase their F1 Score with larger populations. We can also notice that the F1 Score obtained by the **PROS + COS** version increase significantly along with the number of generations, while with the **PROS**, **WMD** and **PROS + WMD** versions, the inverse happens.

Crossover and mutation are two essential operators for the genetic algorithm to be able to find good solutions since they are responsible for introducing variability in the population. In Figure 17, we can see that the objective functions that obtained the best F1 Scores used a moderate crossover rate (between 0.3 and 0.6) and a high mutation rate (0.05 to 0.07). This result demonstrates that a stronger elitism (represented by moderate crossover rates) helps the algorithm to find better solutions. However, this strong elitism leads to the need for a high mutation rate to prevent individuals in the population from becoming too similar with few iterations, which would lead to an early convergence of the algorithm.

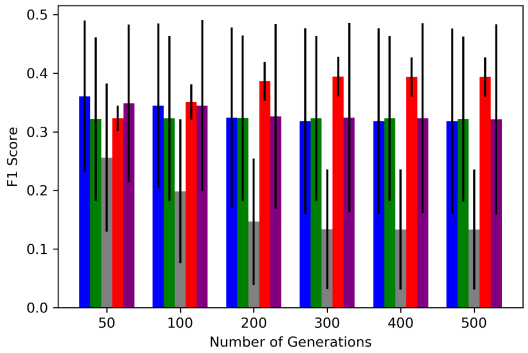
Based on all results presented in this section, we conclude that the objective function

Figure 17 – Performance of objective functions according to the variation of GA parameters. In this Figure, we can see that the objective function PROS + COS outperforms, on average, all others in almost all situations.

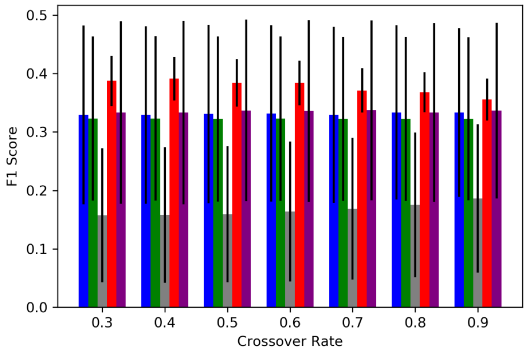
(a) Population size



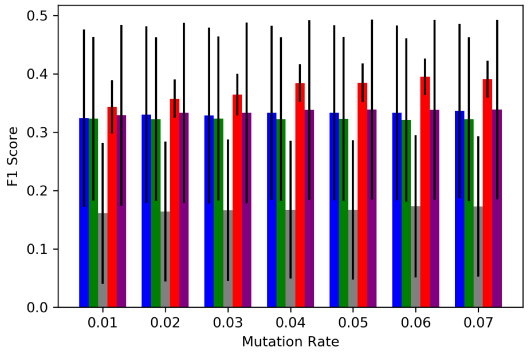
(b) Number of generations



(c) Crossover rate



(d) Mutation Rate



Source: Created by the author (2020)

PROS + COS are capable of obtaining, on average, the best temporal segmentation among all other evaluated versions. Furthermore, we can see that it is also the most stable version since it presents a low standard deviation in all situations of Figure 17 if compared to the others. For this reason, we conducted the next experiments of this chapter using the version **PROS + COS** with the best combination of GA parameters found by the Grid Search. More specifically, with the set of parameters of 7-th row of Table 5. This is because it is the combination of parameters capable of obtaining the best F1 Score with fewer generations (iterations), which results in less computational effort.

5.5 CONVERGENCE ANALYSIS

The convergence of optimization algorithms is an important subject and should be analyzed. Premature convergence can be a problem if the algorithm is too greedy and gets stuck at a local maximum with few iterations (BAKER, 1985). In the same way that

a late convergence can be problematic if the model spends too iterations to converge or if it ends up not converging, which would represent a computational expense for nothing.

The local search is responsible for preventing GA from being trapped at local maxima by doing perturbations in the best solutions found so far to explore their neighborhood and find new solutions (LOURENÇO; MARTIN; STÜTZLE, 2019). Therefore, it is necessary to analyze the impacts of this process in the GA convergence. To do this, the objective function value obtained by the GA in each generation (iteration) for each video from the two datasets were collected. The data collection contains the values for each dataset with the local search on and off.

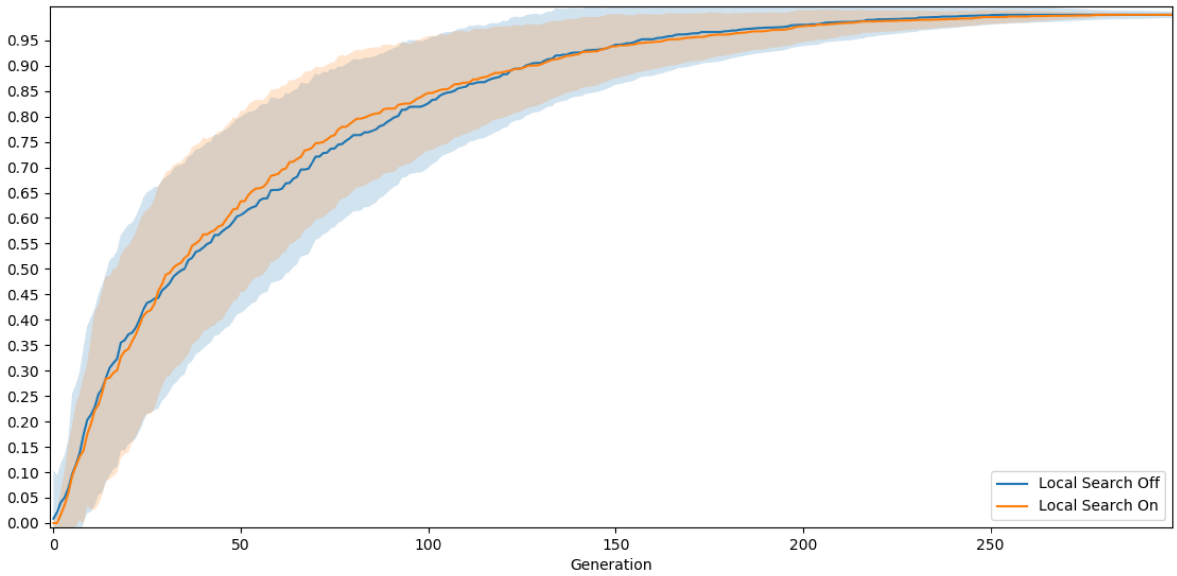
Since each data set has multiple videos, the results were grouped by generation number. Thus, the mean and standard deviation of the objective function values at each generation were calculated. For that, previously, these values were normalized between 0 and 1 for each video lecture. Thus, 0 and 1 represent the lowest and the highest value obtained, respectively, in each video lecture, according to Equation 5.4, where $F(t_i^j)$ is the fitness value of the j-th test instance in the i-th iteration (which corresponds to the i-th generation of the genetic algorithm). The GA was run for 300 iterations for each instance, as defined in the hyperparameter tuning.

$$\overline{F}(t_i^j) = \frac{F(t_i^j) - min(F(t_1^j)...F(t_{300}^j))}{max(F(t_1^j)...F(t_{300}^j)) - min(F(t_1^j)...F(t_{300}^j))} \tag{5.4}$$

Figures 18 and 19 show the convergence results for each data set. The lines represent the mean of the objective function over generations, while the shaded areas represent the standard deviation. It can be noted a slight improvement in average when using the local search in both data sets, especially when it is analyzed the area of the standard deviation. It is possible to see that the blue area (local search off) reaches smaller values than the orange one (local search on) in most of the time. Also, the local search tends to be advantageous with a smaller number of iterations because it accelerates the algorithm convergence. This behavior can also be seen in specific cases of video lectures, as shown in Figures 20 and 21.

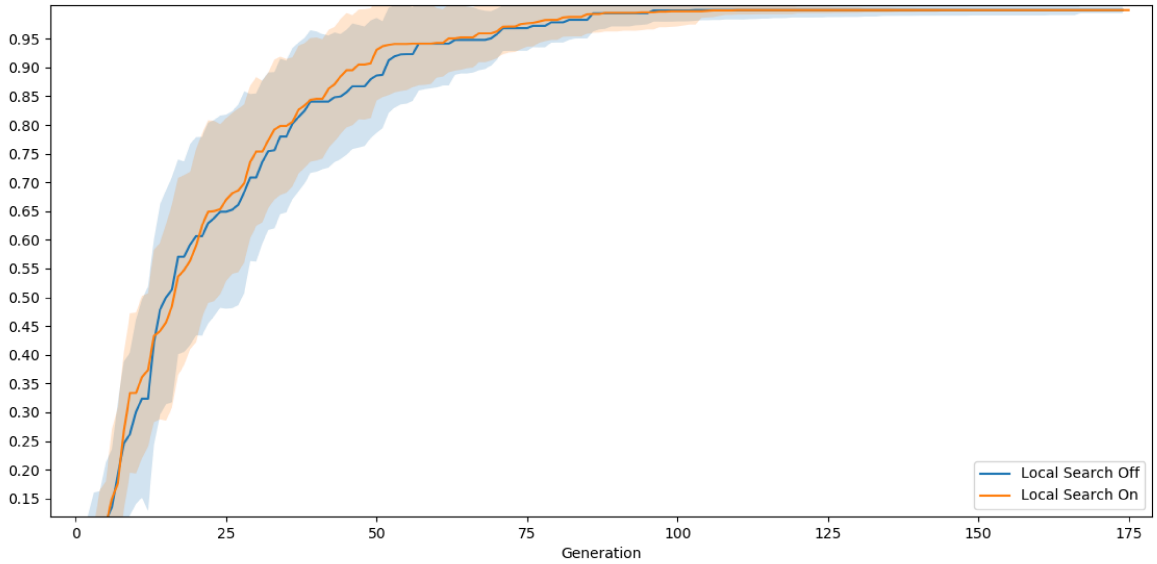
Also, in convergence charts, it is possible to see that the algorithm with local search obtains the best objective function values in most of the generations, but ends up being reached at the end by the algorithm without local search. The elitist nature of the GA used in this work can explain it. In the early generations, the solutions are still very random, so local search helps the algorithm to find, with few generations, better solutions in the neighborhood through local search movements. But as the number of generations advance, GA elitism will select the best solutions for the crossover stage, while discarding the worst ones. Therefore, as is common in any evolutionary algorithm that employs elitism, the populations of the final generations tend to be very similar. This

Figure 18 – Convergence plot in audio-based data set



Source: Created by the author (2020)

Figure 19 – Convergence plot in slide-based data set



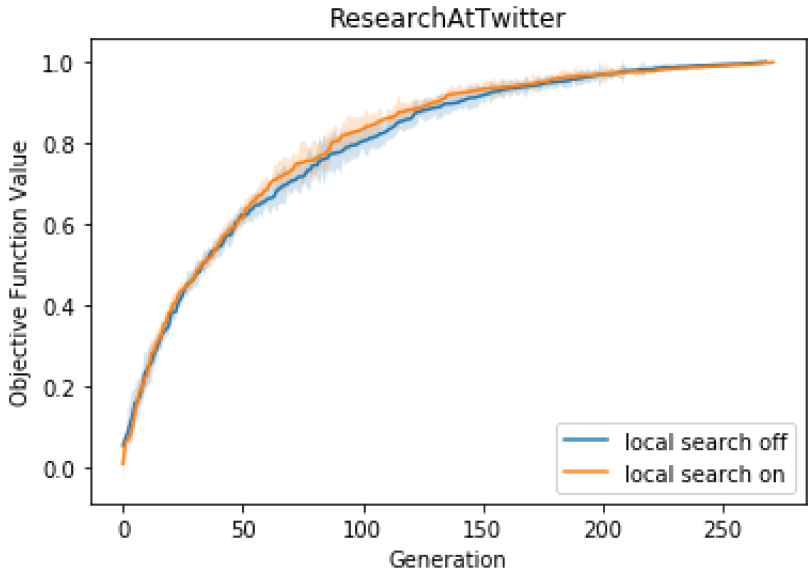
Source: Created by the author (2020)

lack of variety makes local search unable to explore different neighborhoods and find new solutions.

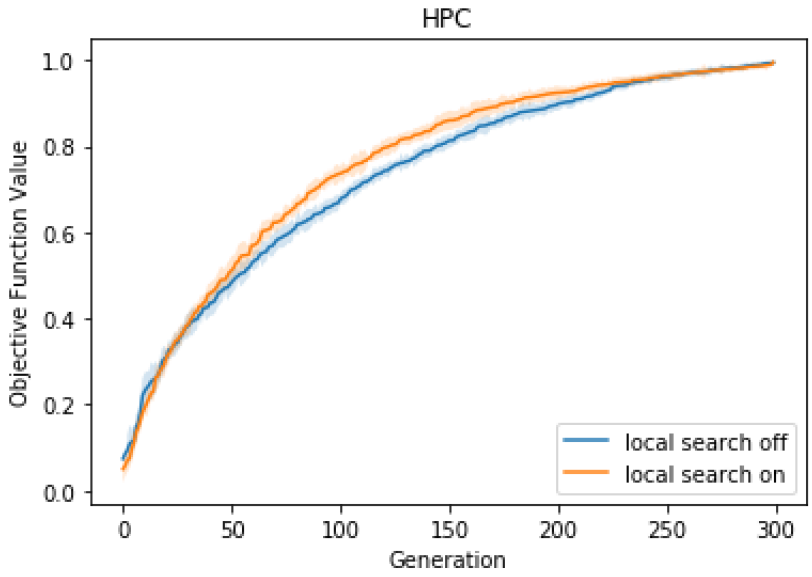
Although it seems that there is almost no difference in using local search when we look to the objective function values, it is possible to notice very interesting characteristics in its use by analyzing the quality of the temporal segmentation obtained. On average, the Precision of the solutions found using local search was greater than when not used. But, in return, the Recall values obtained when using local search tends to be smaller.

Figure 20 – (a), (b) Convergence behavior of two examples of video lectures from audio-based data set.

(a) Convergence through the generations in the video lecture “ResearchAt-Twitter”



(b) Convergence through the generations in the video lecture “HPC”



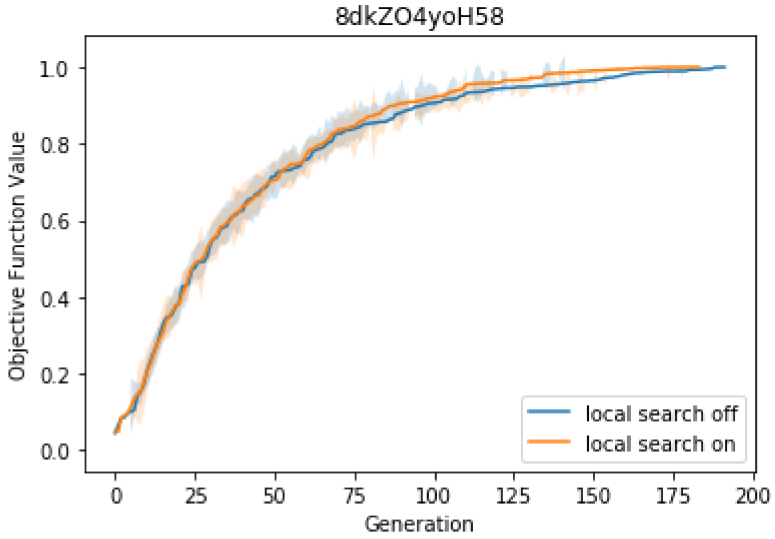
Source: Created by the author (2020)

Anyway, as we can see in the tables 7 and 8, the use of local search can find more balanced solutions between Precision and Recall than not using it. The index m represents the mean of a measure and the index s its respective standard deviation.

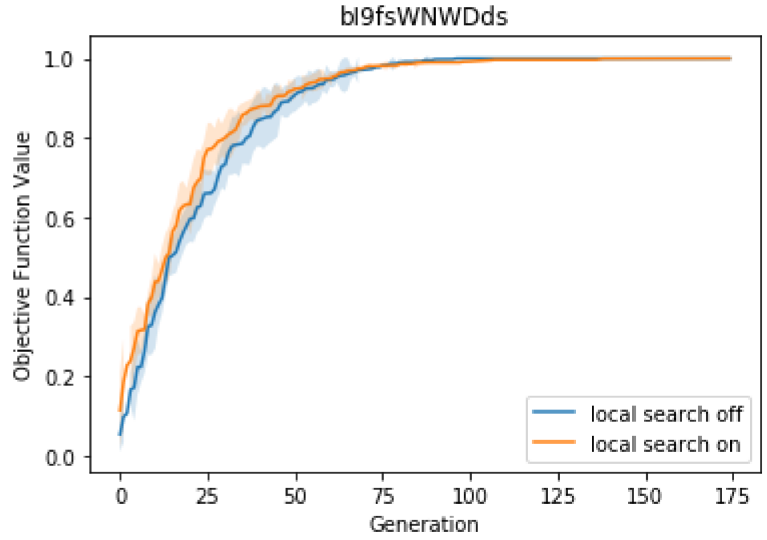
The main explanation for this is that as we build the initial solutions randomly, it is statistically consistent that each gene in the initial solutions has a 50% chance of having the value 1. That is, the initial solutions tend to have too many partitions, which increases

Figure 21 – (a), (b) Convergence behavior of two examples of video lectures from slide-based data set.

(a) Convergence through the generations in the video lecture "8dkZO4yoH58"



(b) Convergence through the generations in the video lecture "bI9fsWNWDds"



Source: Created by the author (2020)

Recall and decreases Precision. By improving the best individuals in the population with local search, we favor, in the first generations, solutions with a smaller number of partitions (as expressed in our multi-objective function where we want to maximize the sum of a utility function while minimizing the number of partitions in the video lecture). As a result, the trend is for the population to converge more quickly to solutions with a more moderate number of partitions to optimize our objective function. On the other hand, when we do not use local search, the tendency is for the characteristics of the initial

Table 7 – Comparison between temporal segmentation quality between Local Search Off and On in audio-based dataset

	$Precision_m$	$Precision_s$	$Recall_m$	$Recall_s$	$F1\ Score_m$	$F1\ Score_s$
Local Search Off	0.31	0.17	**0.69**	0.20	0.38	0.14
Local Search On	**0.49**	0.20	0.50	0.26	**0.44**	0.15

Source: The author (2020)

Table 8 – Comparison between temporal segmentation quality between Local Search Off and On in slide-based dataset

	$Precision_m$	$Precision_s$	$Recall_m$	$Recall_s$	$F1\ Score_m$	$F1\ Score_s$
Local Search Off	0.30	0.20	**0.75**	0.18	0.38	0.12
Local Search On	**0.41**	0.18	0.51	0.16	**0.42**	0.13

Source: The author (2020)

individuals to last for more generations and be passed through the crossover step. Thus, it is more likely that, at the end of the process, the best individuals have more partitions than when we use local search, which favors Recall.

Regarding the execution time of the algorithm, the Tables 9 and 10 show the difference of using or not the local search on each dataset. As expected, the use of local search requires a greater computational effort than not using. Despite this, the local search process, as well as the GA itself, are easily parallelized, which can speed up significantly the algorithm execution.

Table 9 – Execution time (in seconds) comparison between Local Search Off and On in audio-based dataset

	$Time_m$	$Time_s$
Local Search Off	49.33	31.58
Local Search On	63.50	33.92

Source: The author (2020)

Table 10 – Execution time (in seconds) comparison between Local Search Off and On in slide-based dataset

	$Time_m$	$Time_s$
Local Search Off	23.06	16.87
Local Search On	26.75	11.04

Source: The author (2020)

5.6 COMPARISON WITH OTHER APPROACHES

In this section, we compare the performance of our proposal with other two from the literature. For this, we use the version of our algorithm from the previous section with local search enabled since it obtains better F1 scores than when disabled. This comparison is divided in two parts: in the first, the performance of our proposal in the audio-based dataset was compared with the results obtained by the proposal of (GALANOPOULOS; MEZARIS, 2019); in the second part, the performance of our proposal was compared with the proposal of (BISWAS; GANDHI; DESHMUKH, 2015) on the slide-based dataset.

As mentioned in Section 3, the work of (GALANOPOULOS; MEZARIS, 2019) (VFWE) proposes a method for temporal segmentation of video lectures fully based on using a Word2vec model in the audio subtitles. Since the method of (GALANOPOULOS; MEZARIS, 2019) only uses information extracted from audio to perform its segmentation, it is appropriate to evaluate its performance on the audio-based dataset. Whereas we did not find an implementation of this method available, we coded our own version that can be accessed at the URL[14].

In turn, the method proposed in (BISWAS; GANDHI; DESHMUKH, 2015) (MMTOC) is a multimodal algorithm that fuses information from slide presentation and audio subtitles. Although it has been some years since its publication, this work has become a baseline of comparison for methods of temporal segmentation of video lectures. As it is a method designed to work in video lectures with slide presentation, its performance was evaluated on the slide-based dataset.

Tables 11 and 12 present the performance comparison among the approaches. The results are presented as mean and standard deviation of evaluation measures (subsection 5.3) on each data set.

Table 11 – Performance comparison on audio-based data set between our proposal and VFWE

Method	$Precision_m$	$Precision_s$	$Recall_m$	$Recall_s$	$F1\ Score_m$	$F1\ Score_s$
Our proposal	**0.41**	0.18	**0.51**	0.16	**0.42**	0.13
VFWE	**0.41**	0.22	0.14	0.18	0.20	0.10

Source: The author (2020)

Before drawing any comparative conclusion about the results from Tables 11 and 12, we have first calculated the statistical significance of them through two-tailed Student's t test for two independent samples (LAKENS, 2017). This test is used to determine if two independent means are different from each other, by assuming the null hypothesis that they are equal. Thus, we applied the Student's t-Test for each evaluation measure,

[14] https://github.com/eduardorochasoares/video_fragmentation

Table 12 – Performance comparison on slide-based data set between our proposal and MMTOC

Method	$Precision_m$	$Precision_s$	$Recall_m$	$Recall_s$	$F1\ Score_m$	$F1\ Score_s$
Our proposal	0.49	0.20	**0.50**	0.26	**0.44**	0.15
MMTOC	**0.83**	0.22	0.21	0.14	0.31	0.14

Source: The author (2020)

considering the results of the methods that we are comparing. Applying the t-Test, we obtain the p-values showed in Table 13.

Table 13 – Results of Student's t test for the mean of evaluation measures

	Audio-based p-value	Slide-based p-value
Precision	1.00	2.87×10^{-11}
Recall	5.10×10^{-15}	8.40×10^{-7}
F1 Score	8.53×10^{-13}	9.00×10^{-3}

Source: The author (2020)

Setting a confidence interval of 95%, we can state, by convention, that the null hypothesis is rejected if the p-value is less than 0.05 (GREENLAND et al., 2016). Therefore, according to the results in Table 13, we can say that the differences between the results obtained by the algorithms from the literature and our proposal are statistically significant. The only exception concerns the comparison between the average Precision of our proposal and that of VFWE in the audio-based dataset, from which we cannot reject the null hypothesis that they are equal.

Thus, we can say that our proposal outperformed VFWE and MMTOC in both Recall and F1-Score. Regarding Precision, we cannot conclude anything about the comparison with the VFME method. However, our proposal got beaten by MMTOC in the slide-based dataset. These results can be explained by the main characteristics of each method. In the audio-based dataset, VFWE uses a Word2Vec model to identify cut points in video lectures subtitles that can possibly be topic transitions. However, when the topic transition is not abrupt, VFWE may fail to identify it. For example, VFWE would hardly be able to identify a transition between subtopics of the same main topic, because the method depends on a drastic change of the context of words in subtitles. In (GALANOPOULOS; MEZARIS, 2019), the authors conducted experiments with VFWE on an artificially generated data set build through the concatenation of subtitles from different video lectures to represent the topics. That is, in their data set, the video lectures have abrupt topic transitions unlike the real video lectures used in our experimentation. As our method achieved a better Recall in the audio-based data set, we can say that it is able to identify topics that VFWE can not. The approach proposed in this work uses

semantic similarity information combined with prosodic features that helps the algorithm to identify topic transitions that are not perceptible by verbal means.

About the comparisons with MMTOC, some points must to be highlighted. As can be seen at Table 12, the MMTOC outperformed our proposal in relation to Precision, but it was outperformed in Recall and F1-Score like the proposal of (GALANOPOULOS; MEZARIS, 2019). That occurs because (BISWAS; GANDHI; DESHMUKH, 2015) proposes a greedy cost function where importance of words from slides and speech are combined, and then, a dynamic programming algorithm finds the solution by minimizing this cost function. In their approach, topic transitions are a subset of the slide transitions. Hence, the proposal of (BISWAS; GANDHI; DESHMUKH, 2015) is unable to identify topic transitions on the same slide. For example, if the teacher uses a slide to explain various topics through an image of a diagram, the (BISWAS; GANDHI; DESHMUKH, 2015) approach would fail to segment those topics; For that reason, in (BISWAS; GANDHI; DESHMUKH, 2015), the Precision is preferred over Recall. Unlike that, our approach balances Precision and Recall by maximizing the sum of a utility function that combines both semantic and prosodic features (that provides good cues about topic transitions) while minimizing the number of topics to avoid over-segmentation.

6 CONCLUSION AND FUTURE WORKS

In this work, we propose an optimization framework for temporal segmentation of video lectures. Our approach consists of the combination of prosodic and semantic features extracted from a teacher's speech into a linear programming model that we optimize to obtain the temporal segmentation. We based the use of prosodic features on the premise that some of the most prominent tips on topic transition in video lectures are expressed non-verbally. On the other hand, we use semantic features to detect topic transitions based on changing the meaning of words and sentences spoken by the teacher. To do this, we use a pre-trained Word2Vec model to represent the audio transcripts obtained through ASR. Furthermore, in our proposal, we use an elitist genetic algorithm with local search to optimize the linear programming model and obtain the solutions for the temporal segmentation for the video lectures. The use of only features automatically extracted from teacher's speech makes it a versatile approach that can be employed in a large universe of video lectures, as it does not depend on the availability of other sources such as slide shows, textbooks, or manually generated subtitles. This implies that our proposal can be used in real scenarios to improve search in video lecture repositories without any human effort.

As there are many processes involved in our framework, we implemented it as a distributed software architecture composed of modules responsible for performing specific tasks. Within each of these modules, there are processes, called workers, that are responsible for performing these tasks. As each module has a different processing complexity, one of the advantages of our implementation is that it allows us to scale the number of these workers to meet the processing needs of each one. Furthermore, our architecture follows a producer/consumer design pattern, in which module workers consume messages from a task queue when they are available and produce in an output queue at the end of processing. In this way, the processing of the video lecture datasets occurs asynchronously, which improves time performance. Finally, we also shipped each module of our architecture in a Docker container to facilitate tests and the deployment on production servers.

Through the experiments carried out in this work, we were able to present evidence that the combination of prosodic and semantic features provide better solutions to the problem treated in this work than to use them separately. We also show that with a given set of GA parameters, the use of the cosine distance applied to the average word vectors combined with the prosodic features obtained the better F1 Score among all objective functions tested.

Also, we analyze the effects of local search on the convergence of the algorithm and the quality of the solutions found by it. The results showed that, in terms of the objective function, the use or not of the local search leads to close results at the end of

generations. But, when we look at the quality of the found solutions, we can see that the use of the local search makes the algorithm find more balanced solutions between Precision and Recall and with a better F1 score than not using it.

Regarding the versatility of our method, the results endorsed that it is capable of obtaining competitive results in datasets of video lectures with different characteristics. By comparing our proposal with two others from the literature in datasets for which they were specifically designed, we show that our method has outperformed them both in Recall and F1 Score.

As the main weakness of our proposal, we can mention the need to adjust many parameters to obtain the best performance. We may need to adjust them according to the dataset of video lectures that we are applying the method, which can be a complicated task.

As future work, we want to improve our method by considering other prosodic features such as rhythm and syllabic duration in our LP model. Besides, we would like to experiment with other meta-heuristics like Ant Colony (DORIGO; STÜTZLE, 2019) and Particle Swarm (MICHIMURA et al., 2018) to optimize it. Finally, we would also like to carry out experiments with students to assess the quality of our proposal when applied in real e-learning environments, such as Moodle[1].

[1] https://moodle.org/

REFERENCES

AARTS, Emile; AARTS, Emile HL; LENSTRA, Jan Karel. **Local search in combinatorial optimization**. [S.l.]: Princeton University Press, 2003.

ARONS, Barry. Pitch-based emphasis detection for segmenting speech recordings. In: **Third International Conference on Spoken Language Processing**. [S.l.: s.n.], 1994.

BAEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier et al. **Modern information retrieval**. [S.l.]: ACM press New York, 1999.

BAKER, James Edward. Adaptive selection methods for genetic algorithms. In: HILLSDALE, NEW JERSEY. **Proceedings of an International Conference on Genetic Algorithms and their applications**. [S.l.], 1985. p. 101–111.

BALAGOPALAN, Arun; BALASUBRAMANIAN, Lalitha Lakshmi; BALASUBRAMANIAN, Vidhya; CHANDRASEKHARAN, Nithin; DAMODAR, Aswin. Automatic keyphrase extraction and segmentation of video lectures. In: IEEE. **2012 IEEE International Conference on Technology Enhanced Education (ICTEE)**. [S.l.], 2012. p. 1–10.

BISWAS, Arijit; GANDHI, Ankit; DESHMUKH, Om. Mmtoc: A multimodal method for table of content creation in educational videos. In: ACM. **Proceedings of the 23rd ACM international conference on Multimedia**. [S.l.], 2015. p. 621–630.

BLUM, Christian; ROLI, Andrea. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM computing surveys (CSUR)**, Acm, v. 35, n. 3, p. 268–308, 2003.

CAMPOS, JC Alonso; JIMÉNEZ-BELLO, MA; ALZAMORA, F Martínez. Real-time energy optimization of irrigation scheduling by parallel multi-objective genetic algorithms. **Agricultural Water Management**, Elsevier, v. 227, p. 105857, 2020.

CHE, Xiaoyin; LUO, Sheng; YANG, Haojin; MEINEL, Christoph. Sentence-level automatic lecture highlighting based on acoustic analysis. In: IEEE. **Computer and Information Technology (CIT), 2016 IEEE International Conference on**. [S.l.], 2016. p. 328–334.

CHE, Xiaoyin; YANG, Haojin; MEINEL, Christoph. Automatic online lecture highlighting based on multimedia analysis. **IEEE Transactions on Learning Technologies**, IEEE, v. 11, n. 1, p. 27–40, 2018.

CHEVEIGNÉ, Alain De; KAWAHARA, Hideki. Yin, a fundamental frequency estimator for speech and music. **The Journal of the Acoustical Society of America**, ASA, v. 111, n. 4, p. 1917–1930, 2002.

CHRISTENSEN, Mads G. Digital audio signals. In: **Introduction to Audio Processing**. [S.l.]: Springer, 2019. p. 31–43.

DAVILA, Kenny; ZANIBBI, Richard. Whiteboard video summarization via spatio-temporal conflict minimization. In: IEEE. **Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on**. [S.l.], 2017. v. 1, p. 355–362.

DAVIS, Lawrence. Job shop scheduling with genetic algorithms. In: **Proceedings of an international conference on genetic algorithms and their applications**. [S.l.: s.n.], 1985. v. 140.

DORIGO, Marco; STÜTZLE, Thomas. Ant colony optimization: overview and recent advances. In: **Handbook of metaheuristics**. [S.l.]: Springer, 2019. p. 311–351.

EDELKAMP, Stefan; SCHROEDL, Stefan. **Heuristic search: theory and applications**. [S.l.]: Elsevier, 2011.

FRICK, Robert W. Communicating emotion: The role of prosodic features. **Psychological Bulletin**, American Psychological Association, v. 97, n. 3, p. 412, 1985.

FURINI, Marco; MIRRI, Silvia; MONTANGERO, Manuela. Topic-based playlist to improve video lecture accessibility. In: IEEE. **2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)**. [S.l.], 2018. p. 1–5.

GALANOPOULOS, Damianos; MEZARIS, Vasileios. Temporal lecture video fragmentation using word embeddings. In: SPRINGER. **International Conference on Multimedia Modeling**. [S.l.], 2019. p. 254–265.

GENDREAU, Michel; POTVIN, Jean-Yves et al. **Handbook of metaheuristics**. [S.l.]: Springer, 2010.

GIANNAKOPOULOS, Theodoros; PIKRAKIS, Aggelos. Audio features. **Introduction to audio analysis**, Academic Press, p. 79–81, 2014.

GLOVER, Fred; LAGUNA, Manuel. Tabu search. In: **Handbook of combinatorial optimization**. [S.l.]: Springer, 1998. p. 2093–2229.

GOODMAN, David J. The application of delta modulation to analog-to-pcm encoding. **Bell System Technical Journal**, Wiley Online Library, v. 48, n. 2, p. 321–343, 1969.

GOOGLE. **WebRTC**. 2020. Accessed: 2020-01-05. Available from Internet: <<https://webrtc.org/>>.

GOUTTE, Cyril; GAUSSIER, Eric. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In: SPRINGER. **European Conference on Information Retrieval**. [S.l.], 2005. p. 345–359.

GREENLAND, Sander; SENN, Stephen J; ROTHMAN, Kenneth J; CARLIN, John B; POOLE, Charles; GOODMAN, Steven N; ALTMAN, Douglas G. Statistical tests, p values, confidence intervals, and power: a guide to misinterpretations. **European journal of epidemiology**, Springer, v. 31, n. 4, p. 337–350, 2016.

GRUHN, Rainer E; MINKER, Wolfgang; NAKAMURA, Satoshi. **Statistical pronunciation modeling for non-native speech processing**. [S.l.]: Springer Science & Business Media, 2011.

GULZAR, Taabish; SINGH, Anand; RAJORIYA, Dinesh Kumar; FAROOQ, Najma. A systematic analysis of automatic speech recognition: an overview. **Int. J. Curr. Eng. Technol**, v. 4, n. 3, p. 1664–1675, 2014.

HADIAN, Hossein; SAMETI, Hossein; POVEY, Daniel; KHUDANPUR, Sanjeev. Flat-start single-stage discriminatively trained hmm-based models for asr. **IEEE/ACM Transactions on Audio, Speech, and Language Processing**, IEEE, 2018.

HARRIS, Zellig S. Distributional structure. **Word**, Taylor & Francis, v. 10, n. 2-3, p. 146–162, 1954.

HINTON, Geoffrey; DENG, Li; YU, Dong; DAHL, George E; MOHAMED, Abdel-rahman; JAITLY, Navdeep; SENIOR, Andrew; VANHOUCKE, Vincent; NGUYEN, Patrick; SAINATH, Tara N et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. **IEEE Signal processing magazine**, IEEE, v. 29, n. 6, p. 82–97, 2012.

HIRSCHBERG, Julia; LITMAN, Diane; SWERTS, Marc. Prosodic and other cues to speech recognition failures. **Speech Communication**, Elsevier, v. 43, n. 1-2, p. 155–175, 2004.

HOLLAND, John. Adaptation in natural and artificial systems: an introductory analysis with application to biology. **Control and artificial intelligence**, University of Michigan Press, 1975.

HUANG, Anna. Similarity measures for text document clustering. In: **Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand**. [S.l.: s.n.], 2008. v. 4, p. 9–56.

HUGHES, Thad; MIERLE, Keir. Recurrent neural networks for voice activity detection. In: IEEE. **2013 IEEE International Conference on Acoustics, Speech and Signal Processing**. [S.l.], 2013. p. 7378–7382.

HUSAIN, Moula; MEENA, SM. Multimodal fusion of speech and text using semi-supervised lda for indexing lecture videos. In: IEEE. **2019 National Conference on Communications (NCC)**. [S.l.], 2019. p. 1–6.

JAIN, Neha; RASTOGI, Somya et al. Speech recognition systems–a comprehensive study of concepts and mechanism. **Acta Informatica Malaysia (AIM)**, Zibeline International Publishing, v. 3, n. 1, p. 1–3, 2019.

JONG, Kenneth A De; SPEARS, William M. A formal analysis of the role of multi-point crossover in genetic algorithms. **Annals of mathematics and Artificial intelligence**, Springer, v. 5, n. 1, p. 1–26, 1992.

KOCHANSKI, Greg; GRABE, Esther; COLEMAN, John; ROSNER, Burton. Loudness predicts prominence: Fundamental frequency lends little. **The Journal of the Acoustical Society of America**, ASA, v. 118, n. 2, p. 1038–1054, 2005.

KUSNER, Matt; SUN, Yu; KOLKIN, Nicholas; WEINBERGER, Kilian. From word embeddings to document distances. In: **International conference on machine learning**. [S.l.: s.n.], 2015. p. 957–966.

LAKENS, Daniël. Equivalence tests: a practical primer for t tests, correlations, and meta-analyses. **Social psychological and personality science**, Sage Publications Sage CA: Los Angeles, CA, v. 8, n. 4, p. 355–362, 2017.

LAVALLE, Steven M; BRANICKY, Michael S; LINDEMANN, Stephen R. On the relationship between classical grid search and probabilistic roadmaps. **The International Journal of Robotics Research**, SAGE Publications, v. 23, n. 7-8, p. 673–692, 2004.

LEE, Chin-Hui; SOONG, Frank K; PALIWAL, Kuldip K. **Automatic speech and speaker recognition: advanced topics**. [S.l.]: Springer Science & Business Media, 2012.

LEE, Greg C; YEH, Fu-Hao; CHEN, Ying-Ju; CHANG, Tao-Ku. Robust handwriting extraction and lecture video summarization. **Multimedia Tools and Applications**, Springer, v. 76, n. 5, p. 7067–7085, 2017.

LEE, Kwang Y; YANG, Frank F. Optimal reactive power planning using evolutionary algorithms: A comparative study for evolutionary programming, evolutionary strategy, genetic algorithm, and linear programming. **IEEE Transactions on power systems**, IEEE, v. 13, n. 1, p. 101–108, 1998.

LIN, Ming; CHAU, Michael; CAO, Jinwei; JR, Jay F Nunamaker. Automated video segmentation for lecture videos: A linguistics-based approach. **International Journal of Technology and Human Interaction (IJTHI)**, IGI Global, v. 1, n. 2, p. 27–45, 2005.

LIN, Ming; DILLER, Christopher BR; FORSGREN, Nicole; HUANG, Yunchu; JR, Jay F Nunamaker. Segmenting lecture videos by topic: From manual to automated methods. **AMCIS 2005 Proceedings**, p. 243, 2005.

LING, Haibin; OKADA, Kazunori. An efficient earth mover's distance algorithm for robust histogram comparison. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 29, n. 5, p. 840–853, 2007.

LOURENÇO, Helena Ramalhinho; MARTIN, Olivier C; STÜTZLE, Thomas. Iterated local search: Framework and applications. In: **Handbook of metaheuristics**. [S.l.]: Springer, 2019. p. 129–168.

LU, Xugang; LI, Sheng; FUJIMOTO, Masakiyo. Automatic speech recognition. In: **Speech-to-Speech Translation**. [S.l.]: Springer, 2020. p. 21–38.

MAHAPATRA, Debabrata; MARIAPPAN, Ragunathan; RAJAN, Vaibhav. Automatic hierarchical table of contents generation for educational videos. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. **Companion of the The Web Conference 2018 on The Web Conference 2018**. [S.l.], 2018. p. 267–274.

MAN, Kim-Fung; TANG, Kit-Sang; KWONG, Sam. Genetic algorithms: concepts and applications [in engineering design]. **IEEE transactions on Industrial Electronics**, IEEE, v. 43, n. 5, p. 519–534, 1996.

MAO, Qirong; DONG, Ming; HUANG, Zhengwei; ZHAN, Yongzhao. Learning salient features for speech emotion recognition using convolutional neural networks. **IEEE transactions on multimedia**, IEEE, v. 16, n. 8, p. 2203–2213, 2014.

MÀRQUEZ, Lluís; RODRÍGUEZ, Horacio. Part-of-speech tagging using decision trees. In: SPRINGER. **European Conference on Machine Learning**. [S.l.], 1998. p. 25–36.

MAYER, Jörg; JASINSKAJA, Ekaterina; KÖLSCH, Ulrike. Pitch range and pause duration as markers of discourse hierarchy: Perception experiments. In: **Ninth International Conference on Spoken Language Processing**. [S.l.: s.n.], 2006.

MICHIMURA, Yuta; KOMORI, Kentaro; NISHIZAWA, Atsushi; TAKEDA, Hiroki; NAGANO, Koji; ENOMOTO, Yutaro; HAYAMA, Kazuhiro; SOMIYA, Kentaro; ANDO, Masaki. Particle swarm optimization of the sensitivity of a cryogenic gravitational wave detector. **Physical Review D**, APS, v. 97, n. 12, p. 122003, 2018.

MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

MIKOLOV, Tomas; SUTSKEVER, Ilya; CHEN, Kai; CORRADO, Greg S; DEAN, Jeff. Distributed representations of words and phrases and their compositionality. In: BURGES, C. J. C.; BOTTOU, L.; WELLING, M.; GHAHRAMANI, Z.; WEINBERGER, K. Q. (Ed.). **Advances in Neural Information Processing Systems 26**. Curran Associates, Inc., 2013. p. 3111–3119. Available from Internet: <<http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>>.

MIKOLOV, Tomas; YIH, Wen-tau; ZWEIG, Geoffrey. Linguistic regularities in continuous space word representations. In: **Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**. [S.l.: s.n.], 2013. p. 746–751.

MITRA, Urvi; SRIVASTAVA, Garima. A study on agent-based web searching and information retrieval. In: **Intelligent Communication, Control and Devices**. [S.l.]: Springer, 2020. p. 569–578.

MURTY, Katta G. Computational complexity of parametric linear programming. **Mathematical programming**, Springer, v. 19, n. 1, p. 213–219, 1980.

NANDHINI, S; SHENBAGAVALLI, A. Voiced/unvoiced detection using short term processing. **International Journal of Computer Applications**, Citeseer, v. 975, p. 8887, 2014.

NORASET, Thanapon; LIANG, Chen; BIRNBAUM, Larry; DOWNEY, Doug. Definition modeling: Learning to define word embeddings in natural language. In: **Thirty-First AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2017.

OLIVEIRA, Miguel. **Prosodic features in spontaneous narratives**. Thesis (Ph.D.) — Simon Fraser University, 2000.

PASAD, Ankita; SABU, Kamini; RAO, Preeti. Voice activity detection for children's read speech recognition in noisy conditions. In: IEEE. **2017 Twenty-third National Conference on Communications (NCC)**. [S.l.], 2017. p. 1–6.

PAVEL, Amy; HARTMANN, Björn; AGRAWALA, Maneesh. Video digests: a browsable, skimmable format for informational lecture videos. In: ACM. **Proceedings of the 27th annual ACM symposium on User interface software and technology**. [S.l.], 2014. p. 573–582.

POVEY, Daniel; GHOSHAL, Arnab; BOULIANNE, Gilles; BURGET, Lukas; GLEMBEK, Ondrej; GOEL, Nagendra; HANNEMANN, Mirko; MOTLICEK, Petr; QIAN, Yanmin; SCHWARZ, Petr; SILOVSKY, Jan; STEMMER, Georg; VESELY, Karel. The kaldi speech recognition toolkit. In: **IEEE 2011 Workshop on Automatic Speech Recognition and Understanding**. [S.l.]: IEEE Signal Processing Society, 2011. IEEE Catalog No.: CFP11SRW-USB.

RAMOS, Juan et al. Using tf-idf to determine word relevance in document queries. In: PISCATAWAY, NJ. **Proceedings of the first instructional conference on machine learning**. [S.l.], 2003. v. 242, p. 133–142.

RASHEED, Mohammed S. Linear programming for solving solar cell parameters. **Insight-Electronic**, v. 1, n. 1, 2019.

RONCHETTI, Marco. Using video lectures to make teaching more interactive. **International Journal of Emerging Technologies in Learning (iJET)**, International Association of Online Engineering, v. 5, n. 2, 2010.

RONG, Xin. word2vec parameter learning explained. **arXiv preprint arXiv:1411.2738**, 2014.

SCHÜTZE, Oliver; CUATE, Oliver; MARTÍN, Adanay; PEITZ, Sebastian; DELLNITZ, Michael. Pareto explorer: a global/local exploration tool for many-objective optimization problems. **Engineering Optimization**, Taylor & Francis, p. 1–24, 2019.

SHAH, Rajiv Ratn; YU, Yi; SHAIKH, Anwar Dilawar; TANG, Suhua; ZIMMERMANN, Roger. Atlas: automatic temporal segmentation and annotation of lecture videos based on modelling transition time. In: ACM. **Proceedings of the 22nd ACM international conference on Multimedia**. [S.l.], 2014. p. 209–212.

SHRIBERG, Elizabeth; STOLCKE, Andreas; HAKKANI-TÜR, Dilek; TÜR, Gökhan. Prosody-based automatic segmentation of speech into sentences and topics. **Speech communication**, Elsevier, v. 32, n. 1-2, p. 127–154, 2000.

SINDHYA, Karthik; DEB, Kalyanmoy; MIETTINEN, Kaisa. A local search based evolutionary multi-objective optimization approach for fast and accurate convergence. In: SPRINGER. **International Conference on Parallel Problem Solving from Nature**. [S.l.], 2008. p. 815–824.

SOARES, Eduardo R; BARRÉRE, Eduardo. Automatic topic segmentation for video lectures using low and high-level audio features. In: ACM. **Proceedings of the 24th Brazilian Symposium on Multimedia and the Web**. [S.l.], 2018. p. 189–196.

SOHN, Jongseo; KIM, Nam Soo; SUNG, Wonyong. A statistical model-based voice activity detection. **IEEE signal processing letters**, IEEE, v. 6, n. 1, p. 1–3, 1999.

SONG, Fei; CROFT, W Bruce. A general language model for information retrieval. In: ACM. **Proceedings of the eighth international conference on Information and knowledge management**. [S.l.], 1999. p. 316–321.

SOREMEKUN, G; GÜRDAL, Z; HAFTKA, RT; WATSON, LT. Composite laminate design optimization by genetic algorithm with generalized elitist selection. **Computers & structures**, Elsevier, v. 79, n. 2, p. 131–143, 2001.

STOWELL, Dan; GIANNOULIS, Dimitrios; BENETOS, Emmanouil; LAGRANGE, Mathieu; PLUMBLEY, Mark D. Detection and classification of acoustic scenes and events. **IEEE Transactions on Multimedia**, IEEE, v. 17, n. 10, p. 1733–1746, 2015.

SUBUDHI, Badri Narayan; VEERAKUMAR, T; YADAV, Deepak; SURYAVANSHI, Amol P; DISHA, SN. Video skimming for lecture video sequences using histogram based low level features. In: IEEE. **2017 IEEE 7th International Advance Computing Conference (IACC)**. [S.l.], 2017. p. 684–689.

TAKANO, Yasunao; IIJIMA, Yusuke; KOBAYASHI, Kou; SAKUTA, Hiroshi; SAKAJI, Hiroki; KOHANA, Masaki; KOBAYASHI, Akio. Improving document similarity calculation using cosine-similarity graphs. In: SPRINGER. **International Conference on Advanced Information Networking and Applications**. [S.l.], 2019. p. 512–522.

TUNA, Tayfun; JOSHI, Mahima; VARGHESE, Varun; DESHPANDE, Rucha; SUBHLOK, Jaspal; VERMA, Rakesh. Topic based segmentation of classroom videos. In: IEEE. **2015 IEEE Frontiers in Education Conference (FIE)**. [S.l.], 2015. p. 1–9.

VANAJAKSHI, P; MATHIVANAN, M. A detailed survey on large vocabulary continuous speech recognition techniques. In: IEEE. **2017 International Conference on Computer Communication and Informatics (ICCCI)**. [S.l.], 2017. p. 1–7.

VANDERBEI, Robert J et al. **Linear programming**. [S.l.]: Springer, 2015.

WHITLEY, Darrell. A genetic algorithm tutorial. **Statistics and computing**, Springer, v. 4, n. 2, p. 65–85, 1994.

WILLIAMS, Briony. Pitch and duration in welsh stress perception: the implications for intonation. **Journal of Phonetics**, Elsevier, v. 13, n. 4, p. 381–406, 1985.

YANG, Haojin; MEINEL, Christoph. Content based lecture video retrieval using speech and video text information. **IEEE Transactions on Learning Technologies**, IEEE, v. 7, n. 2, p. 142–154, 2014.

YU, Dong; DENG, Li. **AUTOMATIC SPEECH RECOGNITION.** [S.l.]: Springer, 2016.

ZHANG, Yang; ZHANG, Jingjun; ZHANG, Dongwen. Implementing and testing producer-consumer problem using aspect-oriented programming. In: IEEE. **2009 Fifth International Conference on Information Assurance and Security**. [S.l.], 2009. v. 2, p. 749–752.