Arc Summarization of Tv Series

Pedro Filipe Flores Cristóvão

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. David Manuel Martins de Matos
Prof. Ricardo Daniel Santos Faro Marques Ribeiro

Examination Committee

Chairperson: Prof. Miguel Nuno Dias Alves Pupo Correia
Supervisor: Prof. David Manuel Martins de Matos
Member of the Committee: Prof. Alexandre Paulo Lourenço Francisco

May 2017

Acknowledgements

First of all I would like to thanks my supervisor David for all discussions, motivation and critics. For letting me explore this topic using crazy mathematics. Also thanks for all the enthusiasm and patience. I would like to thank my coadvisor Ricardo for all the discussions, advises and patience. Thanks to Pedro Lousada, Rui Lourenço and Alexandre Laborde for all their help and support in my years here at Técnico. Thanks to Paulo and António for their company while working at the thesis on weekends. Thanks to Paulo Figueiredo for some tips and help at starting this thesis. Thanks to Carolina Monteiro for the company and encouragement to finish the thesis. Thanks to my twin brother for patiently hearing me talking about all the mathematics and ideas for this thesis. Thanks to my parents for their patient and support and for keeping me focus. Finally I would like to thank God.

Lisboa, June 23, 2017 Pedro Filipe Flores Cristóvão

For my parents

Resumo

Nesta dissertação, procuramos criar um sistema capaz de gerar sumários de arcos de séries de televisão. Com milhares de horas de video guardadas em sites de partilha de vídeos e serviços de streaming online, surge o interesse em sumarização de video como ferramenta necessária para não perder tempo e acompanhar as nossas séries favoritas. Apresentamos neste trabalho uma maneira de resolver o problema usando apenas a informação das legendas. A solução apresentada utiliza a framework da teoria espectral de grafos para segmentar, encontrar arcos e sumarizar esses arcos.

Abstract

In this dissertation, we aim to create a system capable of generating summaries of arcs of TV series. With thousands of hours of video being uploaded and stored in video-sharing websites and online streaming services, a need for video summarization appears as a necessary tool to save time and catch up with our beloved series. We propose a way to solve this problem using just subtitles information. The presented solution uses the framework of spectral graph theory to segment, find story arcs and summarize those arcs.

Palavras Chave
Keywords

Palavras Chave

Sumarização
Séries de TV
Information Retrieval
Spectral Graph Theory

Keywords

Summarization
TV Series
Information Retrieval
Spectral Graph Theory

Index

List of Tables

1 Introduction

Every year lots of TV/Web series are made and uploaded by professionals and amateurs through video-sharing websites and online streaming services, thus need to summarize all of this information is needed. Several application can be found for video summarization of TV/Web series, (Tsoneva, Barbieri, and Weda 2007) identified some important ones such as catching up with a series, to save time, to recall series, to choose what to watch and finally to skip boring content. Instead of summarizing a series in a abstract way, it is proposed to find and summarize relevant topics/themes present in the series. These topics are called arcs in the series context.

Automatic summarization is a process of reducing an information source (text, video, multiple videos ...) to the most important parts. There are two kinds of summarization extractive and abstractive; Extractive summarization constructs a summary which is a subset of the input information source (phrases of a document, segments of video, ...) while abstractive summarization generates a new information source that communicate the same as the input source, but in a reduced way. In video summarization there are according to (Tsoneva, Barbieri, and Weda 2007) three categories in literature:

- *Final representation*: this criterion refers to the shape of the final result from the summarization. There are two fundamentally different the *still-image* and *moving-image* or *video skims*.

- *Content structure*: this criterion classifies structure of the input content. The videos can be *structured* (news, sports, talk shows) or narrative (movies, TV series).

- *Features*: this criterion covers the means used for creating summaries. Features can be *video* (color histogram, face detection, motion detection), *audio* (speaker detection, audio classification) and text (keywords, main characters identification, subtitles).

A story arc is a subset of the story constrained to a theme/topic, for instance the story of a character in the series or the story of some important event. The goal of this thesis is to find and summarize relevant topics/themes present in a series by producing a *moving-image* summaries of a *narrative* content using *text* features. In order to fulfill this goal we aim to create a system that is able to summarize arcs of a TV series. For this our system will receive as input subtitles of all the episodes of the series, then these subtitles are processed in several steps and the final summaries are made. These steps are represented in figure 1.1 and they are subtitle parsing, stop word removal, episode segmentation, segments clustering, clustered segments summarization and finally video production. All these steps are described in this thesis.

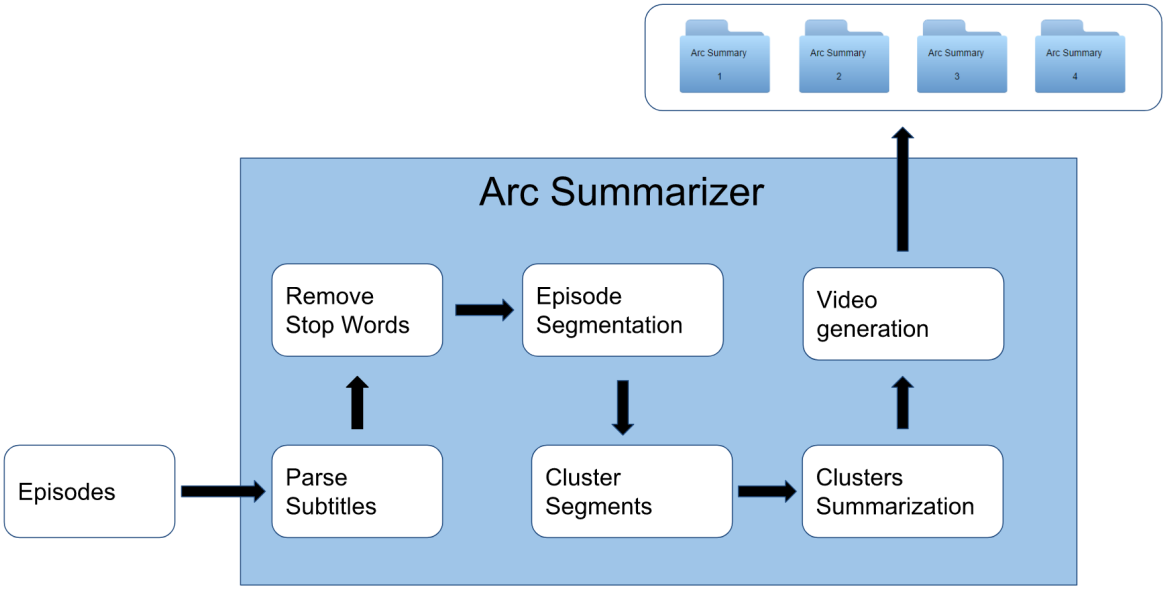


Figure 1.1: Fundamental steps of the arc summarizer described in this dissertation

1.1 Thesis Structure

In the next sections we present related work on video summarization (Section 2) and describe general document summarization algorithms (Section 2). The proposed solution is described as well as the necessary theory to understand it (Section 3). Experiments and results are presented in section 4 and finally we make conclusions and discuss future work (Section 5).

1.2 Notation

Bold variables are vectors eg. $\mathbf{x} \in \mathbb{R}^n$. Upper case letters are matrices eg. $U^T U = I$. If U is a matrix, then U_{ij} and u_{ij} are the element of U that are in the i^{th} row and j^{th} column. If $\mathbf{x} \in \mathbb{R}^n$ then x_i is the i^{th} coordinate of $\mathbf{x}$. Rectangle parenthesis operator $[\cdot]_i$ is a function that takes a vector and outputs its i^{th} coordinate, eg. $[\mathbf{x}]_i = x_i$. $\delta_{i,j}$ is the Kronecker delta function where $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise. $\mathbf{e}_i$ is a vector of the standard basis of the euclidean space, it is defined as $[\mathbf{e}_i]_j = \delta_{i,j}$. Single angle brackets represents the function $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ which is the euclidean inner product. Let $f : \mathbb{R} \mapsto \mathbb{R}$, if f is applied to a vector then $f(\mathbf{x}) = \begin{bmatrix} \dots & f(x_i) & \dots \end{bmatrix}^T$. $\mathbb{R}_+$ is the set of positive real numbers. $\mathbb{P}_n$ is the n-simplex defined as $\mathbb{P}_n = \left\{ \mathbf{x} \in \mathbb{R}^{n+1} : \sum_{i=1}^{n+1} x_i = 1 \right\}$. The entropy of the discrete probability distribution $\mathbf{p} \in \mathbb{P}_n$ is $H(\mathbf{p}) = -\langle \mathbf{p}, \log(\mathbf{p}) \rangle$. The symbol $\mathbf{1}$ is a vector will all entries filled with ones. The symbol $\mathbf{0}$ is a vector will all entries filled with zeros. If $\phi(t)$ is a time varying quantity, then $\dot{\phi}(t)$ is it time derivative. If $\psi(s)$ is a quantity that depends on s, then $\psi'(s) = \frac{\partial \psi}{\partial s}$. If L is a matrix then $\exp(Lt)$ is the matrix exponential.

2 Summarization Algorithms

2.1 General Video Summarization

In this section we present some of related work on video summarization, we will focus more on general unstructured moving-image summarization approaches, since it is close to what we want to solve. Most of the systems compute some kind of ranking on segments of video and extract those to be part of the summary. Some techniques uses simple and effective approach to solve this problem, (Furini and Ghini 2006) just skips frames without sound using various heuristics, but as it succeed in reducing the length of the video it lacks capturing the most important parts of a story.

Other methods are a little more evolved such as (Gong and Liu 2000), which uses singular value decomposition(SVD) of a matrix A, where its columns are feature vectors(they use color histograms of the frames) corresponding to samples frames of the video. SVD is used to construct a low rank approximation of A. Then they construct clusters on this lower dimensional space generated by the SVD. A frame that is close to its centroid is set to be a key-frame, then the system either return a set of the most important key-frames or returns a video summary by finding the longest video shot for each cluster found.

(Ma, Lu, Zhang, and Li 2002) integrates various features using a user attention model, a mathematical function that receive as an input video features and return a real number related to user attention. There are visual attention models (motion attention model, face attention model, etc.), audio attention models (audio saliency attention model, speech attention model, etc.) and linguistic models. Once such attention values are calculated for each frame, this values can be seen as a set of signals in time. This signals are then combined using a function of such signals. This function computes in a way a rank for each frame in time. This rank is called Attention curve. The summary is done by taking shots with high attention values. In figure 2.1 a scheme of the method is presented.

(DeMenthon, Kobla, and Doermann 1998) maps a video into a curve on a high dimensional feature space, then summarization process follows a simplification of that curve, using a recursive multidimensional curve splitting algorithm, which approximates the original curve using a few significant points. In a sense capturing the most important frames.

For a more complete video summarization survey check (Money and Agius 2008). The methods described above suffer from two main problems. First the main purpose of those algorithms were to produce a single video summary, hence failing to produce a summary of a collection of videos. Also their focus are not in the semantic part of the video and thus not capturing the main themes and parts of the story.

(Tsoneva, Barbieri, and Weda 2007) will be described in great detail because it is close to our problem, since video summarization is performed on a semantic level in narrative videos such as movies or series.

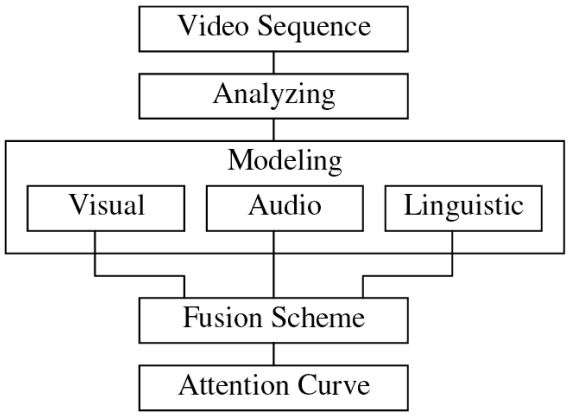


Figure 2.1: User attention model framework

Solution found in (Tsoneva, Barbieri, and Weda 2007) was to first make analysis (parsing) of subtitles and the script of a movie or TV series episode, then compute a script-subtitles alignment in order to map time with characters and scenes. With script-subtitle alignment information, we can segment video into scenes and further into sub-scenes, now a semantic index is constructed. Semantic index is a data structure that stores information about the scenes in the movie. Then features are computed from sub-scenes. These features are used to rank each sub-scene and produce a summary, a pictorial scheme of the algorithm can be seen in Fig.2.2.

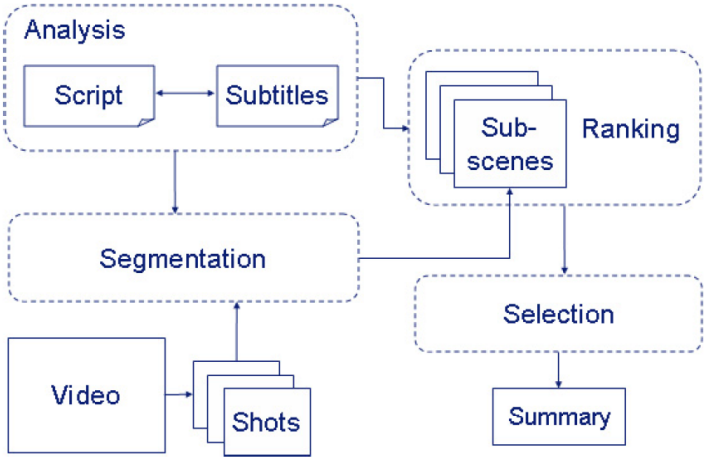


Figure 2.2: Summarization scheme

The parsing of subtitles and the screenplay/script was done in the following way; subtitles are broke into to a sequence of words and stored in a data structure that also stores the time interval that it occurred in the movie/TV episode; script information is also stored in a data structure which contains scene and character lines.

The script-subtitle alignment is done using Needleman–Wunsch algorithm(Needleman and Wunsch 1970), which is a dynamic programming algorithm. The algorithm produces a $(m+1) \times (n+1)$ similarity matrix F where m is the number of words from the sequence of words coming from subtitles and n the number of words coming from the script. If M is the subtitle sequence of words and N the sequence of words coming from the script, F_{ij} stores the

maximum alignment score using the first i words of M and the first j elements in N. Matrix F is then filled according to:

$$F_{i,0} = 0, \quad F_{0,j} = 0 \text{ for } i \in \{0, \ldots, m\}, j \in \{0, \ldots, n\}$$

$$F_{i,j} = max \begin{cases} F_{i-1,j-1} + S_{i,j} \\ F_{i,j-1} + g \\ F_{i-1,j} + g \end{cases}$$

$$S_{i,j} = \begin{cases} 2 \text{ if } M[i] = N[j] \\ -1 \text{ if } M[i] \neq N[j] \end{cases}$$

$$g = -2$$

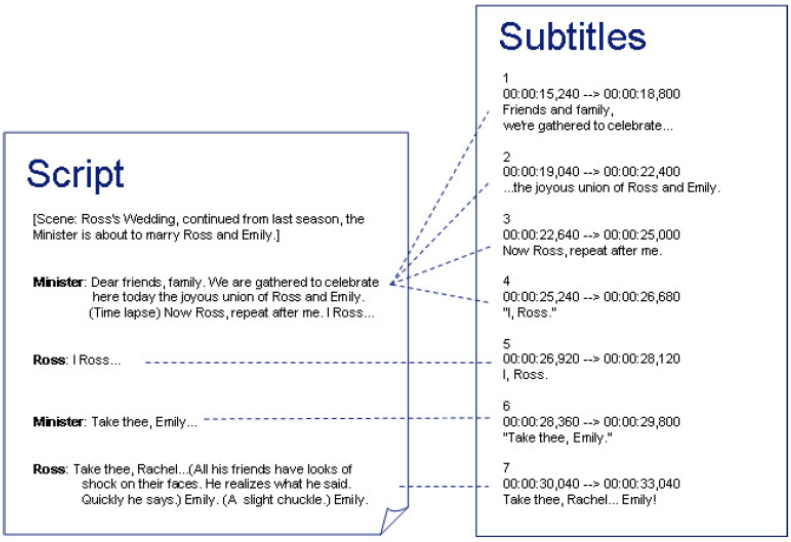


Figure 2.3: script-subtitles Alignment

With the alignment computed a map of time to scenes and characters is created. From this a video can be segmented into scenes, this can be further refined using time gap between subtitles. The gap is defined as the difference of the ending time of the previous and the starting time of the current subtitle element. Subtitle gaps are computed for a given scene, then sub-scenes are selected from that scene whenever it is between to 'big gaps', meaning gaps above some threshold. This threshold was chosen to be 2 seconds for TV series and 3 seconds for movies, these numbers were chosen after analyzing some movies and TV series episodes.

With script-subtitles alignment and sub-scene segmentation done a semantic index is constructed. In the semantic index is stored information about a scene. For instance for every scene there is a list of sub-scenes and for each sub-scene there are a list of keywords extracted from subtitles (without stop words), a list of characters extracted from the script, shots of that sub-scene, etc. Check Fig.2.4 for a visualization of the semantic index.

The extractive summarization algorithm goes as follows, a rank of each sub-scene is calculated as a linear combination of 4 different ranks, these ranks are grouped in a feature vector $\mathbf{f}_i = \{f_{i1}, \ldots, f_{ik}\}\ \forall k \in \{1, \ldots, 4\}$ for each sub-scene i (see (2.1)).

Figure 2.4: Semantic Index

$$\text{rank}_{\text{ss}_i} = \sum_{j=1}^{k} w_j f_{ij} \tag{2.1}$$

These features are *keyword rank* (KR), *character rank* (CR), *main character presence* (MCP) and *first/last appearance* (FLA).

- *Keyword rank*: The author proposes two ways of computing (KR) the first way is to compute a function of the tf-idf score vector of the words sequence of a sub-scene, the function used for sub-scene i is:

$$\text{rank}_i = \frac{\sum_j \text{tfidf}_{i,j}}{M_i}$$

  Where $\sum_j \text{tfidf}_{i,j}$ is the sum of the weights of the keywords of sub-scene i and M_i is the number of keywords in the scene. The other approach is to adopt PageRank algorithm by changing the PageRank graph to an undirected graph where vertices corresponds to a set of words (from the subtitles) in a sub-scene and the edges have a normalized value computed from the co-occurrence of words between two sub-scenes, the normalization is done such that the graph represents a Markov chain, in order to be used by the PageRank algorithm.

- *Character rank*: Uses the same adopted PageRank algorithm described for KR, but this time each vertex has a set of characters that are present in a sub-scene.

- *Main character presence*: is the amount of time main characters talk in a certain scene in terms of seconds based on the subtitles elements duration and the subtitles-script alignment.

- *First/last appearance*: is the number of main characters that appear for first or last time in the movie/episode of TV series in the particular sub-scene.

The pseudo code of the algorithm is in Fig.2.5.

Algorithm *CREATE SUMMARY*
Input: A video item $V = \{\text{ss}_1, \text{ss}_2, ..., \text{ss}_n\}$, its subscenes feature vectors $F_i = \{f_{i1}, ..., f_{ik}\}$, target summary duration D
Output: A summary S

1. $S = \{\}$
2. **for** $i \leftarrow 1$ **to** n **do**
3. $\quad I_{ss_i} \leftarrow \sum_{j=1}^{k} w_j f_{ij}$
4. SV $\leftarrow V$ sorted by I_{ss_i} in descending order
5. $i = 0$
6. **while** $l_S < D$ **do**
7. $\quad S \leftarrow S \cup \{\text{SV}_i\}$
8. $\quad i \leftarrow i + 1$
9. Sort S in chronological order
10. **return** S

Figure 2.5: Summary generation algorithm

(Tsoneva, Barbieri, and Weda 2007) approach has good results, but it lacks to summarize the content in a global way, that is it just summarizes each episode independently of the others, which makes this method not so useful to our problem since it does not find the main topics of a series. Also other aspect which is relevant is the fact that this method relies too much on the script, which is not always easy to find and it does not have a common format.

(Demirtas, Cicekli, and Cicekli 2011) uses general text summarization algorithms to segment and summarize documentaries, again it focus essentially on one document.

2.2 General Text Summarization

Since we want to produce video summaries that take into account the story of a TV series, text features such as subtitles convey much of this information, thus text summarization will be very important in our paper. Next we will describe some general summarization algorithms for text documents.

To simplify we will introduce some definitions that will be useful throughout the paper. Some of these definitions are taken from (Lebanon, Mao, and Dillon 2007). Text is a finite sequence of words with a finite vocabulary.

$$y = (y_1, \ldots, y_N) \quad , y_i \in V$$

Where V is a vocabulary, here it is assumed to be a set of integers $V = \{1, \ldots, |V|\}$. Note that we can retrieve the actual words from V by construction a function (map) $f : V \mapsto \mathbb{V}$ where $\mathbb{V}$ is the set of the actual words. N is the number of words in the text.

A simple model for text document is representing each word with a vector $\mathbf{x} \in \{0, 1\}^V$, such that $\langle \mathbf{x}, \mathbf{1} \rangle = 1$, or simply $[\mathbf{x}_j]_i = \delta_{y_j,i}$. Then a document y is represented by a $N \times |V|$ matrix

$$X = \begin{bmatrix} \cdots \\ \mathbf{x}_i^T \\ \cdots \end{bmatrix}$$

Each row of X represent a bag of word model for a single word. For a lack of word, this model will be called from now on the categorical model. Note that this models capture all the information of a document, since it is possible to reproduce the document with this representation.

Other model for text is the bag of words model, this models basically summarizes all information present in a document. Each document is represented by a vector $\mathbf{x} \in \mathbb{R}^{|V|}$ whose coordinates are given by

$$x_j = \frac{1}{N} \sum_{i=0}^{N-1} \delta_{y_i,j}$$

Note that the bag of word represents each document as a probability distribution of words that appears in the text, since $\sum_{i=1}^{|V|} x_i = 1$. Other way of explaining this model is to simply compute the average vector of the rows vectors in the categorical model. In a sense the bag of words summarizes information present in the categorical model.

Other used model is the (Salton, Wong, and Yang 1975) tf-idf model, similar to the bag of word model, but it captures information about a set of documents, ranking more words that better discriminate a document. The tf-idf is defined as :

$$\text{tfidf}_j = \text{f}_{d,j} * \log\left(\frac{|D|}{n_j}\right)$$

where tfidf_j is the tf-idf score of the word j. $\text{f}_{d,j}$ is the frequency of the word j in the document $d \in D$ and n_j is the number of document where the term j appears. For more about tf-idf see (Salton, Wong, and Yang 1975).

2.2.1 Centroid Summarization

Centroid summarization is based on the work of (Radev, Jing, and Budzikowska 2000) and it can be used for multi-document or single-document summarization. The basic principle of this approach its to model each document or segment by the bag of words model or tf-idf model. Cluster each document using some clustering algorithm. Compute a centroid for each cluster, this represents a pseudo-document that summarizes information of a cluster. Then to form a summary rank each document/segment according to its similarity with the centroid and retrieve that segment/document to become part of that summary. The rank is given by

$$rank(\mathbf{s}) = similarity(\mathbf{s}, \mathbf{s}_{centroid})$$

Where $\mathbf{s}$ is a document/segment and similarity can be given, for instance, using the cosine of the angle between two vector in Euclidean space also known as cosine similarity.

$$\cos(\theta) = \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{|\mathbf{s}_1||\mathbf{s}_2|}$$

For more details see (Radev, Jing, and Budzikowska 2000) and (Radev, Winkel, and Topper 2002).

2.2.2 Maximal Marginal Relevance

Maximal Marginal Relevance (MMR) proposed by (Carbonell and Goldstein 1998) is a query based summarization approach. It splits a document in various segments where each segment $\mathbf{s}_i$ is represent by a bag of words vector or tf-idf vector. The algorithm is an iterative algorithm. At each iteration the algorithm chooses a segment that is the most similar to a query vector and the most different against already chosen segments. This trade-off is parameterized by a variable $\lambda \in [0, 1]$. This algorithm can be described by the following expression:

$$\arg\max_{\mathbf{s}_i} = \left[\lambda\left(Sim_1(\mathbf{s}_i, \mathbf{q})\right) - (1-\lambda)\max_{\mathbf{s}_j} Sim_2(\mathbf{s}_i, \mathbf{s}_j)\right]$$

Where Sim_1 and $Sim2$ are possibly different similarity metrics; $\mathbf{s}_i$ are the unselected sentences and $\mathbf{s}_j$ are the previously selected ones; $\mathbf{q}$ is the query. The trade-off variable λ interpolates between relevance, represented by the similarity of a sentence to the query sentence; and diversity, represented by the similarity of a sentence to its most similar already selected sentence.

2.2.3 LexRank

LexRank (Erkan and Radev 2004) is a graph based algorithm which computes relevance of a sentence by using the concept of eigenvector centrality in a graph representation of sentences. This algorithm is based in the famous PageRank algorithm used at Google, see (Page, Brin, Motwani, and Winograd 1999). After construction a bag of word or tf-idf vector scores of each sentence in the text, an undirected graph of sentences is built by adding an edge every time cosine similarity between two sentences is above a certain threshold. Relevance of a sentence is then computed by iterating until convergence the following equation:

$$p_{t+1}(u) = \frac{d}{|S|} + (1-d)\sum_{v \in \text{adj}[u]} \frac{p_t(u)}{\deg(u)} \tag{2.2}$$

Where $p(u)$ is the relevance of sentence u, $\deg(u)$ is the number of adjacent nodes of u i.e. its degree, $adj[u]$ is the set of adjacent nodes of u and d is a "damping factor". The intuition for such a model is better understood in matrix form as a random walk on a graph, which is described by the following equation:

$$\mathbf{p}_{t+1} = [dU + (1-d)B]^T \mathbf{p}_t \tag{2.3}$$

Where $\mathbf{p}_t$ is a $|S| \times 1$ vector, U is $|S| \times |S|$ matrix with all elements being equal to $\frac{1}{|S|}$ and B is $|S| \times |S|$ such that $B(i,j) = \frac{A(i,j)}{\sum_k A(i,k)}$. Matrix B encodes a transition matrix of the Markov chain process. $B(i,j)$ describes the probability of a random walker to go from vertex i to vertex j. If there is the edge (i,j) in the graph then the random walker transits from i to j, with probability $B(i,j) = \frac{1}{\deg(i)}$. (2.3) computes $\mathbf{p}_t$ which the probability distribution over the sentences of where a random walker is at iteration t. A random walker described by 2.3 on vertex i goes to a random vertex with probability d or go with probability $1-d$ to an adjacent vertex j with probability $B(i,j)$. The stationary distribution $\mathbf{p}$ is obtained from the convergence of (2.3) and it can be show to be the highest eigenvector of $[dU + (1-d)B]^T$ (more details see Perron-Frobenius Theorem). The author still suggests creating a continuous LexRank where a dense graph is constructed with weight between edges given by a normalized cosine similarity between two sentences. This changes (2.2) to the following equation:

$$p_{i+1}(u) = \frac{d}{|S|} + (1-d) \sum_{v \in \text{adj}[u]} \frac{\text{similarity}(u,v)}{\sum_{z \in \text{adj}[v]} \text{similarity}(v,z)} p_i(v)$$

Once again similarity could be the cosine similarity.

2.2.4 Latent Semantic Analysis

Latent Semantic Analysis (LSA) (Dumais 2004) creates a $|V| \times |S|$ (where S is the set of sentences) matrix A where each column correspond to a sentence vector $\mathbf{s}_i = \mathbf{a}_i$ where $\mathbf{a}_i \in \mathbb{R}^{|V|}$ and corresponds to bag of words or tf-idf score vectors. LSA computes a low rank representation of matrix A corresponding to a dimensionality reduction of each $\mathbf{a}_i$ to a k dimensional space , where k is the number of topics. This dimensionality reduction is made using the singular value decomposition which decomposes matrix A in three matrix

$$A = U\Sigma V^T.$$

Where U is a $|V| \times |S|$ orthogonal matrix, V is a $|S| \times |S|$ orthogonal matrix and Σ is a $|S| \times |S|$ diagonal matrix with decreasing singular values. Then to compute the low rank version of A it is selected a $k \in \{1, \ldots, |S|\}$ number of topics. Rank k approximation is calculated by taking the first k columns of U (U_k), the $k \times k$ sub-matrix of Σ (Σ_k) and the first k rows of V^T (V_k^T). Let $\Sigma_k V_k^T = \begin{bmatrix} \ldots & \boldsymbol{\psi}_i & \ldots \end{bmatrix}$ then the column vector $\boldsymbol{\psi}_i$ corresponds to the coordinates of the sentence i in the U_k basis. In semantic terms those coordinates measures how a sentece belong to one of the latent topics. Various methods were build to compute a rank for each sentece and build the summary. (Gong and Liu 2001) chooses k sentences, each sentences is chosen to be the max value of a row of V_k^T.(Steinberger and Jezek 2004) notice two problems: k must be the number of sentences of the summary which, as the length of the summary increases, promotes the inclusion of less significant passages; and sentences with high index values in several dimensions, but never the highest, will never be chosen to be part of the summary. To solve this the author proposes a new ranking function that solves the referred issues.

$$\text{score}(i) = |\boldsymbol{\psi}_i|$$

Where score(i) is the rank of the i^{th} sentence.

3 Proposed Solution

As described in the introduction, the focus of this thesis is to build a system capable of finding and summarizing arcs of TV series. The output of this system should be a moving image summary, meaning a sequence of video segments that summarizes a certain arc. Our proposed solution will have similar scheme as the general document summarization described above. An additional step is added to the general framework, where a selection and identification of arcs is made previous to the summarization step. The general lines of the algorithms are described below.

3.1 Main Algorithm

The main algorithm is as follows: Read all subtitle files of a series. Process data in order to construct a global vocabulary of the series, this could be done by removing stop words, apply stemming, etc. Then a sequential representation of a document is computed for each document. Using this representation, segments are build. Now the system has a set of segments. A similarity graph is build from the segments, this represents a global structure of the series. Arcs are found by clustering using either the similarity graph or segments information. The number of clusters can be chosen by the user. After the arc determination process a summarization algorithm is applied to each arc in order to select the most important segments. Each segment determines a time interval in a certain episode. This time interval is then used to cut the video in order to produce moving-image summary. The output of the program is a set of folders where each folder represents an arc. Every one of these folders have a set of video segments representing the most important segment of each arc. This set of videos for each arc represents the arc summary. Some post processing of these videos can be done, such as concatenation, etc.

3.2 Data preparation

As explain above, the first step is to process the data. In this system, the only pre-processing done is removing stop words. Stop words can be defined as words that are uniform distributed across documents. It is important to remove this words from the analysis since they do not contribute to identifying what a document is all about. Two document could have lots of stop words in common but with no similarity between them, hence removing stop word will generally produce better results when comparing documents (Wilbur and Sirotkin 1992). There are simple approaches to remove stop words. There are stop words lists which can be used to remove stop words when parsing subtitles. This method is efficient but is way too general and fails to identify stop word of a particular TV show. Removing top frequent word in a set of documents is another method. Although captures specific domain stop words it also removes high frequency discriminant words, these words are important since they characterizes a document.

Other method is to remove words that have low inverse document frequency rank, which was defined in section 2.2. This method solves the issues of the previous methods but lacks a way of choosing a threshold to define which words are low rank.

We select two approaches to remove stop words. The stop words list method and another method proposed by (Lafon and Lee 2006). This technique have a nice property that there is low and upper bounds on the threshold. The method works by computing the entropy of the distribution of documents given a word. In a sense this description fits exactly the definition above. A word that is distributed uniformly across all document will have maximum entropy. Let P be a matrix where each row i defines a probability distribution $\mathbf{p}_i \in \mathbb{P}_{|D|-1}$ over documents given the word i. Where $[\mathbf{p}_i]_j = P(d = j|w = i)$ represents the probability of document $d = j$ have word $w = i$. Let $0 \leq \rho \leq 1$, then this system classifies a word as stop word if

$$H(\mathbf{p}_i) > (1 - \rho) \max\{H(\mathbf{x})\}. \tag{3.1}$$

The calculation is made by first computing the bag of words models for each document. A matrix X containing in each row a vector $\mathbf{x}_i \in \mathbb{P}_{|V|-1}$ represents the bag of words model for each document. From $[\mathbf{x}_i]_j = P(w = j|d = i)$ we can compute $P(d = i|w = j)$ using Bayes theorem

$$\begin{aligned} P(d = i|w = j) &= \frac{P(d = i)P(w = j|d = i)}{\sum_{k=1}^{|D|} P(d = k)P(w = j|d = k)} \\ \Leftrightarrow P(d = i|w = j) &= \frac{\frac{1}{|D|}P(w = j|d = i)}{\frac{1}{|D|}\sum_{k=1}^{|D|} P(w = j|d = k)} \\ \Leftrightarrow P(d = i|w = j) &= \frac{P(w = j|d = i)}{\sum_{k=1}^{|D|} P(w = j|d = k)}. \end{aligned}$$

Using $P(d = i|w = j)$ we compute its entropy and classify each word as stop if it fulfills 3.1, note that we assumed that the prior distribution $P(d = i)$ is the uniform distribution. Words with low entropy were also removed, since these words tend to appear in few document, therefore not useful when comparing documents. Thus we remove any word that satisfies

$$H(\mathbf{p}_i) > (1 - \rho) \max\{H(\mathbf{x})\} \text{ or } H(\mathbf{p}_i) < \rho \max\{H(\mathbf{x})\}. \tag{3.2}$$

Note that X is useful to calculate all kind of statistics about the series.

3.3 Scale Space Segmentation and Locally Weighted Bag of Words

In this section a fundamental part of this work is presented, the segmentation process. Segmentation is important since a summary is based on these segments, this means that the basic unit of a summary will be this segment. Mathematically a segment is a set $I = \{t \in \mathbb{S} : s_1 \leq t \leq s_2\}$ where $\mathbb{S}$ could be $\mathbb{R}$ or $\mathbb{N}$. We want to partition a document to semantic cohesive segments with low time complexity. (Tsoneva, Barbieri, and Weda 2007) segments

video using a script-subtitle alignment, but in this project we restricted our approach to use just subtitles information since they are more available. (Tsoneva, Barbieri, and Weda 2007) also used time between subtitles to further segment the document, but this approach doesn't have at least explicitly, the purpose of doing a semantic segmentation. We propose to segment the subtitle text document using a scale-space segmentation approach similar to (Lebanon, Mao, and Dillon 2007), (Yang 2012) and (Slaney and Ponceleon 2001). Since they minimizes a energy that relates in a sense to the semantics of the text. In this section will introduce scale-space segmentation and its relation to a semantic segmentation. We will see that Locally Weighted Bag of Words (Lowbow) (Lebanon, Mao, and Dillon 2007) and scale-space segmentation are equivalent. Using spectral graph theory, the scale-space representation can be compressed, thus reducing time complexity of the segmentation. Finally in the end of this section we describe the final algorithm

Scale-space methods is a technique developed by (Witkin 1984) for segmentation of 1D signals, later it was generalized to vector values 1D signals (Lyon 1987) and multivariate signals (Koenderink 1984) and (Perona and Malik 1990). Although not mentioning scale-space segmentation, (Lebanon, Mao, and Dillon 2007) used a similar approach to text segmentation using Lowbow document representation.

It is simple to understand scale-space segmentation using a problem, in this way we will explain the intuition behind the method. Imagine that someone wanted to approximate a noisy signal with a smooth one. This is a very old problem and it appears in many different areas, signal processing (Rudin, Osher, and Fatemi 1992), machine learning(Bishop 2006) and functional data analysis (Ramsay 2006) to name a few. This problem can formalized as Tikhonov regularization problem. Using a continuous notation for 1D signals it can be defined as :

$$\min_u \int_a^b (f(x) - u(x))^2 \, dx + \lambda S\left(u, u', u''\right), \ \lambda \geq 0 \tag{3.3}$$

Where S is a function that gives high values to non-smooth functions, typically $S(u, u', u'') = \int_a^b u'(x)^2 \, dx$. This cost function 3.3 tries to find a function that is smooth and close to f in a square L_2 norm way. This relates to our problem in the following sense. In a text document words referring a certain topic appear more often in a close sequence of words. Nevertheless lots of off topic words appear in this sequence. These words could be regarded as noise in the sequence. So if we consider a document as a signal we can remove its noise by computing the solution to Tikhonov regularization problem. Once the noise is removed from the signal, large scale differences become easier to spot, turning the segmentation task easier to perform. The solution to 3.3 approximates the original signal while removing local differences. The scale of the local differences are parameterized by λ. So as the scale parameter rises the signal becomes smoother and thus larger scale features will become available.

The scale-space method tries to solve Tikhonov regularization problem by starting off with the original signal and then smooth it out until it reach the defined scale. Scale-space method do not generate a single solution as 3.3, but a family of functions parameterized by t. To get a solution with this method we need to find scale t such that it solves 3.3 for a particular λ.

This approach can be formalized as a gradient descend flow of

$$\int_a^b u'(x)^2 \, dx \tag{3.4}$$

with f as the initial condition. This is a way to minimize 3.4, but instead of only getting its minimum (which are linear and constant functions) it computes family of functions parameterized by t such that as $t \to \infty$ the cost function 3.4 tends to its minimum.

This can be proved as follows: Let

$$\mathcal{L}_{u(x,t)}(t) = \int_a^b \left(\frac{\partial u(x,t)}{\partial x}\right)^2 dx \tag{3.5}$$

be the function we want minimize. So we want to find a family of functions $u(x,t)$ parameterized by t such that

$$\frac{\partial \mathcal{L}_{u(x,t)}(t)}{\partial t} \leq 0. \tag{3.6}$$

First we compute $\frac{\partial \mathcal{L}_{u(x,t)}(t)}{\partial t}$

$$\frac{\partial \mathcal{L}_{u(x,t)}(t)}{\partial t} = \int_a^b 2u'(x,t)u'_t(x,t)\ dx$$

by integration by parts

$$\frac{\partial \mathcal{L}_{u(x,t)}(t)}{\partial t} = [2u'u'_t]_a^b - \int_a^b 2u''u_t\ dx.$$

Imposing as boundary conditions that $u'_t(a,t) = u'_t(b,t) = 0$, we get

$$\frac{\partial \mathcal{L}_{u(x,t)}(t)}{\partial t} = -\int_a^b 2u''u_t\ dx.$$

Now we define a flow

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} &= \frac{\partial^2 u}{\partial x^2} \\ u(x,0) &= f(x). \end{aligned} \tag{3.7}$$

This flow satisfies 3.6, since

$$\frac{\partial \mathcal{L}_{u(x,t)}(t)}{\partial t} = -\int_a^b 2u''u_t\ dx$$

substituting u_t by 3.7,

$$\frac{\partial \mathcal{L}_{u(x,t)}(t)}{\partial t} = -\int_a^b 2(u'')^2\ dx < 0.$$

Note that the cost function is convex since 3.5 is a squared L^2 norm of u'. From this we conclude that 3.5 has only one minima, see (Boyd and Vandenberghe 2004). Therefore it is

proved that the family of functions defined by 3.7 improves 3.5 as $t \to \infty$ and it is reaching the global minimum. In literature this is called a gradient flow because

$$\frac{\partial \mathcal{L}_{u(x,t)}(t)}{\partial t} = \langle\langle \nabla \mathcal{L}, u_t \rangle\rangle .$$

Where $\langle\langle \cdot, \cdot \rangle\rangle$ is the L^2 inner product of functions spaces. For more about optimization of functions and gradient flows check (Olver) and (Peletier 2011). Note that with the 3.7 f will get smoother as the parameter t increases. In the limit the solution to 3.5 is given by the Euler-Lagrange formula to be $u'' = 0$, whose solutions are linear and constant functions. A way to solve 3.7 is by convolution of f with the heat kernel $H(x,t)$ see (Saloff-Coste). The heat kernel in the Euclidean geometry is given by

$$H(x,t) = \frac{1}{\sqrt{4\pi t}} \exp\left(\frac{-x^2}{4t}\right).$$

However most of the times in literature, the authors tend to use the Gaussian kernel

$$G(x,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-x^2}{2\sigma^2}\right)$$

, which is closely related with the heat kernel. We get the Gaussian kernel from the heat kernel by a simple change of coordinates $t = \frac{\sigma^2}{2}$.

Having the intuition behind scale-space in mind, scale-space segmentation segments a signal where the local differences are higher, formally this means cutting the segment whenever $\left(\frac{\partial u}{\partial x}\right)^2$ is maximum. Smoothing f reduces local differences while maintaining large scale differences, then the method cuts f whenever it finds a large difference locally, see (Witkin 1984) for more information.

(Yang 2012) and (Slaney and Ponceleon 2001) build a matrix X which represent a kind of categorical model, where each row represents a unit of text (a word, phrase, etc.). Each row could represent a unit of text in different ways. (Slaney and Ponceleon 2001) represented a unit of text as a Latent Semantic Index vector and (Yang 2012) tried many methods such as the simple bag of words and the Latent Dirichlet Allocation vector. The scale-space segmentation is done similar to (Witkin 1984) but here each column is a signal that is the result of a convolution with a Gaussian filter. Matrix X can be though as a vector value signal, where each row is a sample of this signal. In order to segment the text they use as topic boundaries local maxima of the signal derivative norm.

(Lebanon, Mao, and Dillon 2007) introduces the Lowbow and one of its application is text segmentation. Lowbow uses categorical model defined in section 2.2 with additive smoothing (or Laplace smoothing). A document is encoded as $N \times |V|$ matrix X where each row is defined as

$$\mathbf{x}_i = \begin{bmatrix} \dots & \frac{\delta_{y_i,j}+c}{1+|V|c} & \dots . \end{bmatrix}^T \tag{3.8}$$

Where $c \geq 0$ and it is the additive smoothing parameter. He defines $x : \{1, \dots, N\} \times |V| \mapsto [0,1]$ to be a function such that $x(i,j) = [\mathbf{x}_i]_j$. Then a continuous representation is defined to be a function $\phi : [0,1] \times |V| \mapsto [0,1]$, such that :

$$\phi(s,j) = x(\lceil Ns \rceil, j).$$

Now the Lowbow curve is defined as $\gamma : [0,1] \times |V| \mapsto [0,1]$

$$\gamma_\sigma(\mu, j) = \int_0^1 \phi(s,j) K_{\mu,\sigma}(s)\, ds. \tag{3.9}$$

We can see that γ is the convolution of the function $\phi(s,j)$ with the kernel $K_{\mu,\sigma}(s)$ which can be the Gaussian kernel. Segmentation of the Lowbow is also done by finding the maximum of

$$\left| \frac{\partial \boldsymbol{\gamma}_\sigma(\mu)}{\partial \mu} \right|_2 .$$

Lowbow and scale-space are equivalent, since they are both convolution of a text signal with a low-pass spatial filter. We will use this method to find and segment topic boundaries to obtain the basic unit in our system.

After we remove stop words in the prepossessing step, it is constructed for each document a $N \times |V|$ matrix X described above in 3.8. Now rather than define a discretization of a continuous problem like 3.5 we use a discrete energy that captures the same idea. This energy seems in a sense less artificial since in a computer everything needs is discrete and discretizations(which are approximations) are prune to errors. Plus this way we can use ideas of (Chung 1997) and (Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013), which are frameworks that already solve our problem. Using this method will produce an algorithm with lower time complexity than the former scale-space segmentation.

Every row of X, $\mathbf{x}_i$ is a point in $\mathbb{P}_{|V|-1}$. Also there is a implicit sequential structure in X since $\mathbf{x}_{i+1}$ represent a word after $\mathbf{x}_i$. The sequential aspect of the text can be given by adding a graph structure to text, where word i is connected to word $i-1$ and $i+1$. This document representation can be thought as an graph embedding in the simplex. This is just another equivalent view from the ones already described. Then in order to reduce the differences between adjacent vertices a discrete version of 3.3 or 3.5 should be defined.

Let

$$\min_{\mathbf{f}_1,\ldots,\mathbf{f}_N \in \mathbb{P}_{|V|-1}} \sum_{(i,j)\in E} |\mathbf{f}_i - \mathbf{f}_j|^2 = \sum_{k=1}^{|V|} \sum_{(i,j)\in E} ([\mathbf{f}_i]_k - [\mathbf{f}_j]_k)^2 \tag{3.10}$$

be such definition of the smoothing cost function. Note that E is the set of edges. In a way $\mathbf{f}_i - \mathbf{f}_j$ can be thought as discrete derivative of the Lowbow curve. Now we will derive a solution to this problem using the continuous problem as guidance. So we need to find gradient flow to 3.10 with X as the initial condition. It is useful to make a change of variable to 3.10 oprimization problem. This transformation is :

$$\min_{\mathbf{f}_1,\ldots,\mathbf{f}_N \in \mathbb{P}_{|V|-1}} \sum_{k=1}^{|V|} \sum_{(i,j)\in E} ([\mathbf{f}_i]_k - [\mathbf{f}_j]_k)^2 \Leftrightarrow \min_{\mathbf{g}_1,\ldots,\mathbf{g}_{|V|} \in \mathbb{R}^N} \sum_{k=1}^{|V|} \sum_{(i,j)\in E} ([\mathbf{g}_k]_i - [\mathbf{g}_k]_j)^2.$$

Let A be the $N \times N$ adjacency matrix of the graph defined above. Where $A_{ij} = 1$ if there is an edge between i and j and 0 otherwise, then

$$\sum_{k=1}^{|V|} \sum_{(i,j)\in E} ([\mathbf{g}_k]_i - [\mathbf{g}_k]_j)^2 = \sum_{k=1}^{|V|} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij} ([\mathbf{g}_k]_i - [\mathbf{g}_k]_j)^2$$

$$= \sum_{k=1}^{|V|} \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij}[\mathbf{g}_k]_i^2 - 2A_{ij}[\mathbf{g}_k]_i[\mathbf{g}_k]_j + A_{ij}[\mathbf{g}_k]_j^2$$

$$= \sum_{k=1}^{|V|} \sum_{i=1}^{N} \deg(i)[\mathbf{g}_k]_i^2 - 2\sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij}[\mathbf{g}_k]_i[\mathbf{g}_k]_j + \sum_{j=1}^{N} \deg(j)[\mathbf{g}_k]_j^2 \tag{3.11}$$

Note that $\deg(i) = \sum_{j=1}^{N} A_{ij}$ and that since the graph is undirected we have that $\sum_{j=1}^{N} A_{ij} = \sum_{i=1}^{N} A_{ij}$, thus 3.11 is equal to :

$$= \sum_{k=1}^{|V|} 2 \left(\sum_{i=1}^{N} \deg(i)[\mathbf{g}_k]_i^2 - \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij}[\mathbf{g}_k]_i[\mathbf{g}_k]_j \right)$$

$$= 2\sum_{k=1}^{|V|} \mathbf{g}_k^T D\mathbf{g}_k - \mathbf{g}_k^T A\mathbf{g}_k$$

Where D is a diagonal matrix with $D_{ii} = \deg(i)$ and zeros elsewhere. Finally we get that our cost function is equal to:

$$2\sum_{k=1}^{|V|} \mathbf{g}_k^T (D - A)\mathbf{g}_k = 2\sum_{k=1}^{|V|} \mathbf{g}_k^T L\mathbf{g}_k. \tag{3.12}$$

Matrix L in literature is called the graph laplacian matrix. Now with 3.12 a gradient flow can be found in a simpler manner. Let $\mathcal{L}$ be discrete smoothing cost function, then :

$$\mathcal{L}(\mathbf{g}_1, \ldots, \mathbf{g}_{|V|}) = \frac{1}{2} \sum_{k=1}^{|V|} \mathbf{g}_k^T L\mathbf{g}_k$$

Note that the factor $\frac{1}{2}$ does not alter the solution of our problem (3.12), it is introduced just to simplify calculations. Then the gradient descend is given by

$$\frac{\partial \mathbf{g}_k}{\partial t} = -\nabla_{\mathbf{g}_k} \mathcal{L}$$

$$-\nabla_{\mathbf{g}_k} \mathcal{L} = -L\mathbf{g}_k \tag{3.13}$$

$$\mathbf{g}_k(0) = [\ldots, [\mathbf{x}_i]_k, \ldots]^T.$$

It is important to note for full understanding of this method that if $G(t)$ is a $N \times |V|$ matrix where each columns is given by $\mathbf{g}_k(t)$, then $G(0) = X$. Now it can be understood the change of variables done in the beginning. $\mathbf{g}_k(t)$ are the columns of X after smoothing t seconds. We are

simply doing a spatial filter to the columns of X. We can aggregate 3.13 into a single equation using $G(t)$ as follow:

$$\dot{G} = -LG, \;\; G(0) = X. \tag{3.14}$$

The solution of 3.14 is,

$$G(t) = \exp(-Lt)X. \tag{3.15}$$

It can be shown that L is a semi-positive definite matrix with positive eigenvalues $\lambda_1 = 0 \leq \lambda_2 \cdots \leq \lambda_N$ and orthonormal eigenvectors U (Von Luxburg 2007). Hence by making a change of variables $\hat{G} = U^T G$, 3.14 becomes:

$$U^T \dot{\hat{G}} = -LU^T \hat{G}$$

$$\dot{\hat{G}} = -ULU^T \hat{G}$$

By the spectral theorem $L = USU^T$, therefore:

$$\dot{\hat{G}} = -S\hat{G}. \tag{3.16}$$

Solution of 3.16 is just :

$$\hat{G}(t) = \exp(-St)\hat{G}(0) \tag{3.17}$$

Converting equation 3.17 back to the original variables we get :

$$G(t) = U \exp(-St)U^T X$$

$$G(t) = U \exp(-St)\hat{X}. \tag{3.18}$$

Here S is a diagonal matrix with the eigenvalues of L, $\exp(-St)$ is a diagonal matrix with entries $\exp(-St)_{ii} = e^{-\lambda_i t}$ and $\hat{X} = U^T X$ is a graph spectral representation of X (Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013), which is a kind of generalization of a Fourier transform of a signal in a graph. Note also that our graph for sequential representation of text is very simple, hence eigenvalues and eigenvectors of the graph laplacian have a simple closed form solution. This provides efficient calculation of eigenvectors in $O(N^2)$ time, see (Bindel 2011) and (Burkardt 2013). We can still improve performance by noting that :

$$t \to \infty \, , e^{-\lambda_i t} \to 0, \;\; \forall i > 1 \tag{3.19}$$

and

$$e^{-\lambda_{k+1} t} < e^{-\lambda_k t}, \;\; k \in \{1, \ldots, N\}. \tag{3.20}$$

So we can approximate $G(t) = X(t)$ by using just the first k eigenvalues and eigenvectors,

$$X(t) \simeq U_k \exp(-St)_k \hat{X}_k(0). \tag{3.21}$$

Where U_k is the first k columns of U, $\exp(-St)_k$ is a $k \times k$ sub matrix of $\exp(-St)$ and $\hat{X}_k(0)$ are the first k rows of $\hat{X}(0)$. Using 3.19 and 3.20 a good approximation of $X(t)$ can be given by finding a k such that it is the maximum value that fulfills $e^{-\lambda_k t} > \varepsilon, \;\; \varepsilon > 0$. Where ε is close to zero. In practice our system works the way around, the system ask for k and it computes the t parameter. t is calculated as follow :

$$t(k) = -\frac{\log(\varepsilon)}{\lambda_{k+1}}.$$

More about how we select a good k for our summaries will be discussed later in the text.

To perform segmentation we will use the same principle of (Lebanon, Mao, and Dillon 2007) and segment text where there is a local maximum of the derivative norm at a scale t. Let derivative of the text signal be defined as a function $\mathbf{d}_X : \{1, \ldots, N-1\} \mapsto \mathbb{R}^{|V|}$ equal to :

$$\mathbf{d}_X(k) = \mathbf{x}_{k+1} - \mathbf{x}_k.$$

This can be written in a succinct way in matrix form:

$$\mathcal{D}_N X = \begin{bmatrix} \mathbf{d}_X(1) \\ \ldots \\ \mathbf{d}_X(N-1) \end{bmatrix}$$

Where $\mathcal{D}_N$ is the tridiagonal matrix defined as $(\mathcal{D}_N)_{i,i} = -1$, $(\mathcal{D}_N)_{i,i+1} = 1$ and 0 otherwise, for all $i \in \{1, \ldots, N-1\}$. Then we compute

$$|\mathbf{d}_{X(t)}(k)|^2 = \left\langle \mathbf{d}_{X(t)}(k), \mathbf{d}_{X(t)}(k) \right\rangle \tag{3.22}$$

which is the norm of the derivative of the smoothed text signal. Now segmentation of the text is done by segment text in the local maxima of $|\mathbf{d}_{X(t)}(k)|^2$. In summary the algorithm can be described using the following pseudo-code :

- 1. Compute $\mathbf{d}_{X(t)} = \mathcal{D}_N X(t)$
- 2. Compute $|\mathbf{d}_{X(t)}(k)|^2$
- 3. find and segment local maxima from $|\mathbf{d}_{X(t)}(k)|^2$

The time complexity of segmenting a document given k eigenvectors and eigenvalues(which just takes $O(Nk)$) is $O(k|V| + 2Nk|V| + (N-1)|V|)$. Where $O(k|V|)$ came from $\exp(-St)_k \hat{X}$ (note that $\exp(-St)$ is a diagonal matrix), $O(2Nk|V|)$ from $U_k(\exp(-St)_k \hat{X})$ and $U_k^T X$, and finally $(N-1)|V|$ from $\mathcal{D}_N X(t)$ (note that $\mathcal{D}$ is a tridiagonal matrix). It is important to check that the operation that takes more time is the smoothing procedure, since $O(k|V| + 2Nk|V| + (N-1)|V|) = O(Nk|V|)$. One could argument that there is no speed improvement when comparing with the convolution of text signal with the heat/gaussian kernel,

after all it takes only $O(Nb|V|)$, where b is the bandwidth of the kernel. But there is something missing which is that b and k are both functions of the scale t and as $t \to \infty$, $b(t) \to N$ and $k(t) \to 1$. Usually t will not be small hence our method becomes faster than the convolution. Its is worth noticing that $U_k \exp(-St) U_k^T$ is the discrete heat kernel (Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013) and there are lots of benefits of using discrete objects instead of discretizations of continuous objects, see (Bobenko, Sullivan, Schröder, and Ziegler 2008).

One final note on this text representation, as (Lebanon, Mao, and Dillon 2007) proved when $t \to \infty$,

$$X(t) \to \mathbf{1} \left(\frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_i(0) \right)^T \tag{3.23}$$

which represents the bag of words model in each row of $X(t)$. We know from (Burkardt 2013) and (Von Luxburg 2007) that the first eigenvector of the laplacian matrix is a constant vector $\frac{1}{\sqrt{n}}\mathbf{1}$ and the first eigenvalue is 0. Thus we can also conclude from our representation the same as (Lebanon, Mao, and Dillon 2007). When $t \to \infty$, $e^{-\lambda_k t} \to 0$ for $k > 1$, then just the first eigenvalue is needed to compute $X(t)$ as in 3.21. Therefore as $t \to \infty$ we get :

$$\lim_{t\to\infty} X(t) = U_1 U_1^T X(0) = \begin{bmatrix} \frac{1}{\sqrt{n}} \\ \dots \\ \frac{1}{\sqrt{n}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{n}} & \cdots & \frac{1}{\sqrt{n}} \end{bmatrix} X(0).$$

Which is the same result as 3.23 formula. Since less rows of $\hat{X}$ are needed to represent $X(t)$ as $t \to \infty$, this shows that the spectral representation is a good way of compressing Lowbow curves.

3.4 Clustering for arc/topic finding

After the text segmentation process we get a set of text segments per episode. These segments are the basic unit of this summarization system. These segments were by construction made to capture topic boundaries within each episode. Now using these basic units we will find the relations between them by clustering these segments. The partition formed by the clustering process will be the arcs of the series. These segments are represented by a bag of word model of the words of a given segment. Mathematically this translates in a operation that receives a document X (smoothed categorical model described above) and a segment $I = [i, j]$ as input and produces a segment given by:

$$\mathbf{s}_I = \frac{1}{j-i} \sum_{k=i}^{j} \mathbf{x}_k.$$

From now on the algorithm only works with segments $\mathbf{s}_i, \quad i \in S$, where S is the set of all segments of a series. To connect segments, a similarity graph is constructed. This graph is a K-nearest neighbor graph, where each vertex i is connected to vertex j if j is among the the k nearest neighbors of i. Nearest in this context is defined with a metric. Three metrics are used in this project:

- *Euclidean distance*: $d(\mathbf{x}, \mathbf{y}) = |\mathbf{x} - \mathbf{y}|$
- *Cosine distance*: $d(\mathbf{x}, \mathbf{y}) = 1 - \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{|\mathbf{x}||\mathbf{y}|}$
- *Simplex distance*: defined in (Lebanon et al. 2005) to be the geodesics distance between to points in $\mathbb{P}_{N-1}$ using the Fisher information metric.

$$d(\mathbf{x}, \mathbf{y}) = 2 \arccos \left(\sum_{i=1}^{N} \sqrt{x_i y_i} \right)$$

By the definition above this graph is directed since the relation of neighbor is not symmetric, but since an undirected graph is needed for the clustering and summarization process then every time an edge is added between vertex i and j another edge is added between j and i. There are many types of similarity graphs, but as (Von Luxburg 2007) pointed out, K-nearest neighbor graph tends to create large connected components and it is "less sensitive to unsuitable choice of parameters". In practice we always choose low value of k, (Von Luxburg 2007) recommend to use k on the order of $\log(|S|)$. We define an arc of a series to be all segments related to a certain topic. The way we found those arcs was to cluster segments into T topics. We explore three methods to cluster segments. Two of them use the similarity graph as the basis of the cluster method while the third algorithm doesn't. The first two algorithms are based on spectral graph theory as segmentation process described above. The third algorithm uses a standard topic finding algorithm in natural language processing which is the Latent Dirichlet Allocation(LDA) (Blei, Ng, and Jordan 2003).

3.4.1 Spectral Clustering

Having a similarity graph of segments and K number of topics, we want to partition the graph such similar segments stay in the same cluster and dissimilar ones on a different clusters. One framework of graph clustering algorithms is the spectral clustering approach. The intuition behind the method is clear with the following problem. We will use $K = 2$ so that is simpler to explain. Let $G = (S, E, W)$ be a graph with vertex S(each vertex is a segment), edges E and similarity matrix W where similarity between i and j is computed as:

$$w_{ij} = e^{-\frac{d_{ij}^2}{2\sigma^2}}. \tag{3.24}$$

Where d_{ij} is the distance between segments i and j using the metrics above. Let $\mathbf{f} \in \{1, -1\}^{|S|}$ be a indicator function on each vertex. If $f_i = 1$ then i belongs to the group 1 otherwise belongs to group 2. To partition this graph in two, dissimilar vertex must be in different groups, so a cost function is designed for that purpose. We want to assign a group to each vertex such that the sum of the similarity values between different groups is minimum. This is formalized as :

$$\min_{\mathbf{f} \in \{-1,1\}^{|S|}} \frac{1}{2} \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} w_{ij} (f_i - f_j)^2 \tag{3.25}$$

Note that when $f_i = f_j$ then $(f_i - f_j)^2 = 0$ and so w_{ij} isn't taken in account. From the cost function above we can deduce that a minimum will select similar values for f_i and f_j when the

similarity is high and different values when the similarity is low. The only issue with 3.25 is that it is a integer optimization problem due to the constraint of $\mathbf{f} \in \{-1, 1\}^{|S|}$. This problem is hard in fact it can be showed to be NP-Hard ((Wagner and Wagner 1993) and (Von Luxburg 2007)). The way this problem is usually solved is by means of relaxation, that is relaxing the constraint $\mathbf{f} \in \{-1, 1\}^{|S|}$ to be $\mathbf{f} \in \mathbb{R}^{|S|} : |\mathbf{f}|^2 = 1$. Other constraint we need to add is the constraint on the size of each group, this can be made by noticing that

$$\sum_{i=1}^{|S|} f_i = n_1 - n_2 \mid \mathbf{f} \in \{-1, 1\}^{|S|}$$

Where n_1 and n_2 are the number of segments in groups 1 and 2 respectively. A easy way to reformulate 3.25 is by noticing the similarity with 3.10 function. Using similar steps we can derive a matrix formulation of 3.25.

$$\min_{\mathbf{f} \in \mathbb{R}^{|S|}} \mathbf{f}^T (D - W)\mathbf{f}, \text{ s.t } \langle \mathbf{f}, \mathbf{1} \rangle = n_1 - n_2 \text{ and } |\mathbf{f}|^2 = 1 \tag{3.26}$$

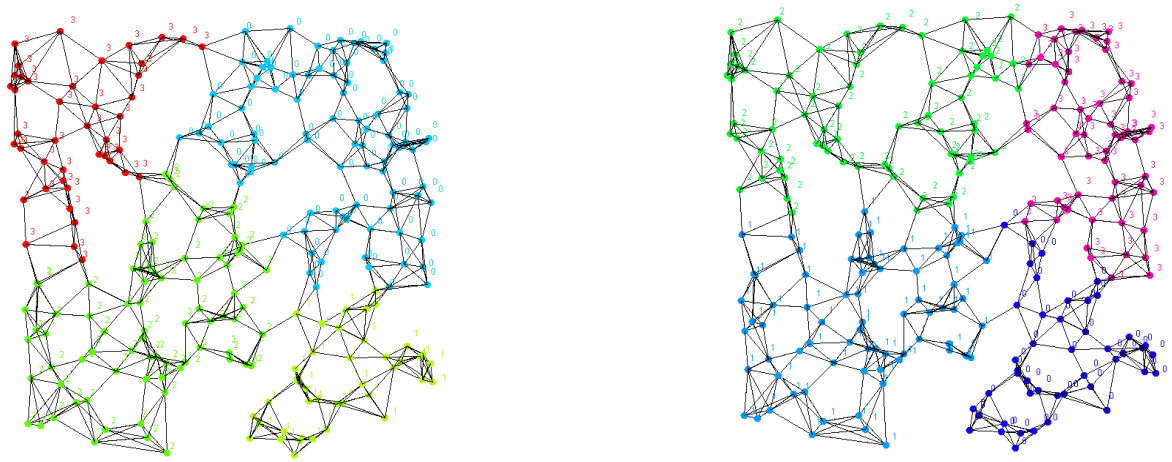
Where D is a diagonal matrix where each entry is given by $D_{ii} = \sum_{j=1}^{|S|} w_{ij}$. $L = (D - W)$ is graph laplacian matrix as we see on 3.12 but here W is used instead of the adjacency matrix. As show in (Newman 2010), 3.26 is solved by the second lower eigenvector of L. With the relaxed version of 3.25, $\mathbf{f}$ is not a well defined indicator function then what is usally done it to quantize this function using the K-Means algorithm (Lloyd 1982) with $K = 2$. The generalization to K partitions, is well explained in (Von Luxburg 2007) where there are K functions $\mathbf{f}_i \in \mathbb{R}^{|S|}, \quad i \in \{1, \ldots, K\}$ that represent indicator functions of a vertex belonging to a group. If vertex i belongs to class j then $[\mathbf{f}_j]_i = 1$ and -1 otherwise. Using the same idea as before we can formulate a optimization problem that partition the graph into K components :

$$\min_{\mathbf{f}_1, \ldots, \mathbf{f}_K} \sum_{k=1}^{K} \mathbf{f}_k^T L \mathbf{f}_k = \text{tr}(F^T L F), \text{ s.t } F^T F = I \tag{3.27}$$

where $F = \begin{bmatrix} \mathbf{f}_1 & \ldots & \mathbf{f}_k \end{bmatrix}$. The solution to 3.27 are the first K eigenvectors of L (see (Kokiopoulou, Chen, and Saad 2011) and (Von Luxburg 2007)). Then a quantization is made to the K eigenvectors of L, matrix U_K. This is done by using the K-means algorithm, where each row of U_K is seen as a data point in the K-means algorithm. With a few changes to this algorithm it is possible not only to maximize the dissimilarities between clusters (the one explained is performing just that) but also maximize the similarities within clusters((Von Luxburg 2007)). This changes the cost function to :

$$\min_F \text{tr}(F^T L F), \text{ s.t } F^T D F = I \tag{3.28}$$

Where the solution of the problem is given by the K lowest generalized eigenvectors $L\mathbf{v}_k = \lambda D \mathbf{v}_k$. Typically this is the cost function used for most of the problems. In order to solve 3.28 usually a change of variables is made, let $H = D^{\frac{1}{2}} F$ (note that $D^{\frac{1}{2}}$ is easy to compute since D is diagonal) then 3.28 becomes :



(a) (b)

Figure 3.1: Clustering of a random graph using Spectral Clustering algorithms, (a) Not normalized Spectral clustering; (b) Normalized Spectral Clustering. In this case there isn't so much differences in these methods.

$$\min_{H} \operatorname{tr}(H^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} H), \text{ s.t } H^T H = I. \tag{3.29}$$

Using this transformation the solution becomes easier to compute since optimization of 3.29 turns to a eigenvectors problem. $D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ is called the normalized laplacian, and has nicer properties than the unnormalized version (see (Von Luxburg 2007) for discussion). After H is calculated F is discovered by $F = D^{-\frac{1}{2}} H$.

The last interpretation of Spectral Clustering algorithms is very similar to Diffusion Distance clustering. In this interpretation Spectral clustering is viewed as doing K-means on a graph embedding, such that the coordinates of this embedding are close to each other if the vertex are similar to each other. This embedding problem can be formalized as 3.28 see (Belkin and Niyogi 2003). There is also the (Ng, Jordan, Weiss, et al. 2002) approach to spectral clustering, which is very similar to the normalized spectral clustering explained above, that instead of computing F they use a normalized version of H.

3.4.2 Diffusion Distance Clustering

Based on the diffusion distance of (Coifman and Lafon 2006), we perform diffusion distance clustering algorithm to find the arcs of a series. (Lafon and Lee 2006) already used this framework to compute topics in a collection of documents with good results. Diffusion distance framework basically creates an embedding of a graph using similar theory as (Belkin and Niyogi 2003), but where laplacian eigenmaps fail to give a explicit metric in its embedding, diffusion distance defines a proper family of embeddings with a metric distances. There are many similarities with spectral cluster algorithm, although this one has a clear justification of K-means step.

The algorithm work with the same graph of segments that the spectral cluster algorithm works. This algorithm basically creates a graph embedding such that the metric on this space is the diffusion distance. Diffusion distance uses the distributions of the random walks starting in x and y as a proxy to compute the distance between x and y. This distance takes in account the graph connectivity at different scales t. It is a distance that consider all paths from x to y and not just one like the geodesic distance. Let $P = D^{-1}W$ be the Markov transition matrix on the graph. Let

$$p_t(x, y) = [P^t \mathbf{e}_x]_y$$

be the probability of x walk into y in t steps. Then diffusion distance is

$$\mathcal{D}_t^2(x, y) = \sum_{y=1}^{|S|} \frac{1}{\phi_0} \left(p_t(x, y) - p_t(z, y)\right)^2$$

$$\phi_0 = \frac{\mathbf{tr}(D)}{D_{yy}}$$

Where $\phi_0(y) = p_\infty(x, y)$ when the graph is connected. It was noticed by (De Goes, Goldenstein, and Velho 2008) that in the limit case of $|S| \to \infty$ then $p_t(\cdot, \cdot)$ converges to the heat kernel $\mathcal{H}_t(\cdot, \cdot)$. Hence (De Goes, Goldenstein, and Velho 2008) defined the following diffusion distance, in continuous theory language :

$$\mathcal{D}_t^2(x, z) = \int_{\mathcal{M}} \left(\mathcal{H}_t(x, y) - \mathcal{H}_t(z, y)\right)^2 dz.$$

Where $\mathcal{H}_t(x, y)$ is heat kernel on a manifold $\mathcal{M}$. Using this approach we can use the framework described in (Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013) to define this diffusion distance. The idea is to find a discrete heat kernel on the graph. In section 3.3 we derived such discrete heat kernel. In summary we defined a smoothing energy on the space of function on graphs ($\mathbf{f} \in \mathbb{R}^{|S|}$) similar to the cost function 3.25.

$$\mathcal{L}(\mathbf{f}) = \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} w_{ij}(f_i - f_j)^2 = \mathbf{f}^T L \mathbf{f}. \tag{3.30}$$

Then a gradient descend flow was used to minimize such energy, this flow is the discrete heat equation as seen in equation 3.13.

$$\dot{\mathbf{f}} = -L\mathbf{f}$$

whose solution is

$$\mathbf{f}(t) = U \exp(-St) U^T \mathbf{f}(0) = H(t)\mathbf{f}(0).$$

Where $H(t)$ is the discrete heat kernel (a $|S|$ by $|S|$ matrix). Now the diffusion distance can be written

$$\mathcal{D}_t^2(x, z) = \sum_{i \in S} \left(H_{i,x}(t) - H_{i,z}(t)\right)^2 = |H(t)\mathbf{e}_x - H(t)\mathbf{e}_z|^2 \tag{3.31}$$

$$\Leftrightarrow |H(t)(\mathbf{e}_x - \mathbf{e}_z)|^2 = |U \exp(-St) U^T (\mathbf{e}_x - \mathbf{e}_z)|^2 \tag{3.32}$$

$$\Leftrightarrow \left\langle U\exp(-St)U^T(\mathbf{e}_x-\mathbf{e}_z), U\exp(-St)U^T(\mathbf{e}_x-\mathbf{e}_z)\right\rangle = (\mathbf{e}_x-\mathbf{e}_z)^T U\exp(-St)\exp(-St)U^T(\mathbf{e}_x-\mathbf{e}_z) \tag{3.33}$$

$$\Leftrightarrow |H(t)(\mathbf{e}_x-\mathbf{e}_z)|^2 = |\exp(-St)U^T(\mathbf{e}_x-\mathbf{e}_z)|^2 \tag{3.34}$$

Since $e^{-\lambda_k t}$ decay very fast an approximation of this distance can be made using just k eigenvectors as demonstrated in equation 3.21,

$$\mathcal{D}_t^2(x,z) \approx |\exp(-St)_k U_k^T\mathbf{e}_x - \exp(-St)_k U_k^T\mathbf{e}_z|^2. \tag{3.35}$$

By consider

$$\mathbf{\Psi}_t(x) = \exp(-St)_k U_k^T\mathbf{e}_x = \begin{bmatrix} e^{-\lambda_1 t}[\boldsymbol{\phi}_1]_x \\ \dots \\ e^{-\lambda_k t}[\boldsymbol{\phi}_k]_x \end{bmatrix} \tag{3.36}$$

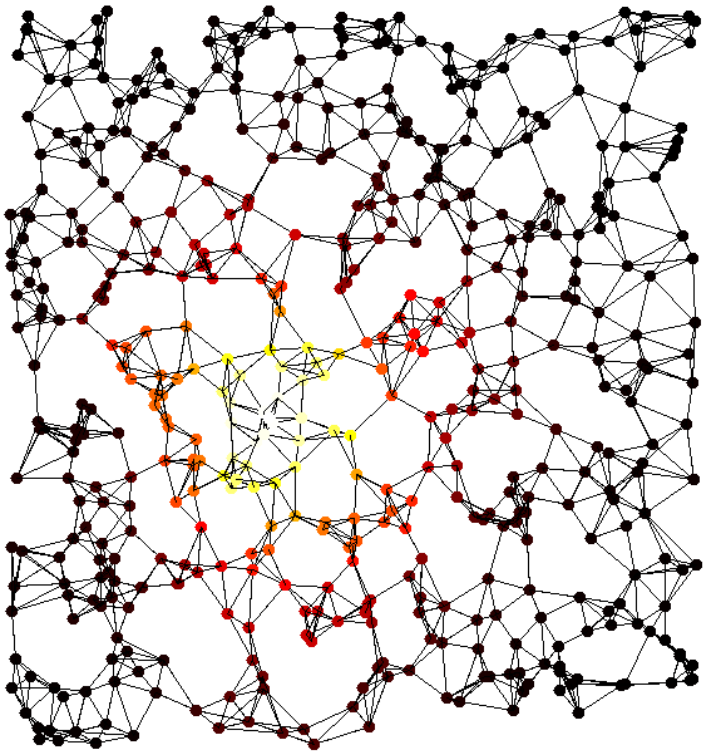


Figure 3.2: Visualization of the heat kernel on the graph. Light color nodes represent higher values of the heat kernel, while darker nodes represent lower values.

as the coordinates of the embedding of vertex x, where $\boldsymbol{\phi}_i$ is the ith column of U, the matrix of eigenvectors of L. From 3.36 and 3.35 it is concluded that the euclidean distance of the embedding is the diffusion distance of the graph. Then by applying the K-means algorithm on this embedding we get a partition of the graph. This algorithm offer a low dimensional

representation of the graph (check 3.36). When using this algorithm there is a need to choose a good parameter t, since low values of t will capture only information about its closest neighbors while large values t will collapse the embedding to just one point turning all segments equal by the diffusion distance. Similar issue appeared in section 3.3. Such selection of t will be further discussed later in the text. One final detail is needed to conclude this section which is how k is selected from t, using a similar approach as in scale-space segmentation section 3.3. We select k such that it is the maximum value that fulfills $e^{-\lambda_k t} > \varepsilon, \quad \varepsilon > 0$.

In summary this method find a partition of a graph using similar technique as spectral clustering(using an embedding based on the eigenvectors of the laplacian matrix), while correcting some unjustified steps of the spectral clustering. Note that this graph embedding minimizes 3.30 which is also the spectral clustering optimization problem. It has a parameter t that controls the scale where the algorithm operates. The main idea of the method is to diffuse heat for some time t from a vertex x, this defines a function on the graph called the heat kernel at x. Then we use the heat kernel at different segments to compute a distance between them. This defines a distances that takes into account the graph connectivity. Also there is the random walk interpretation described at the beginning of this section, for more about random walks and its relation with diffusion check (Lawler 2010). This relation of the heat equation and computation of distances was also explored by (Crane, Weischedel, and Wardetzky 2013) who also uses the heat kernel to compute geodesic distances of surfaces.

3.4.3 Latent Dirichlet Allocation Clustering

While the last approaches treated the segments in a abstract way as a similarity graph. This method removes this structure but considers each segment as document generated by a probabilistic generative model, the latent Dirichlet allocation (LDA) (Blei, Ng, and Jordan 2003). This method represents each document as a mixture of k topics/arcs. Such topics are represented as a distribution over words ($\mathbb{P}_{|V|-1}$). This method can be explained by the following generative process:

- 1. Selects k topics. For $i = 1 \dots k$:
  - (a) $\boldsymbol{\beta}_i \sim$ Dirichlet$(\boldsymbol{\eta}, |V|)$
- 2. For each segment $s \in S$:
  - (a) $\boldsymbol{\theta}_s \sim$ Dirichlet$(\boldsymbol{\alpha}, k)$
  - (b) For each word $w_j \in s$:
    - i. $z_j \sim$ Categorical$(\boldsymbol{\theta}_s)$
    - ii. $w_j \sim$ Categorical$(\boldsymbol{\beta}_{z_j})$

Where Dirichlet$(\boldsymbol{\eta}, M)$ is the Dirichlet distribution with probability density function given by

$$\mathbf{x} \in \mathbb{P}_{M-1}, \quad \text{Dirichlet}(\mathbf{x}; \boldsymbol{\alpha}, M) = \frac{\Gamma\left(\sum_{i=1}^{M} \alpha_i\right)}{\prod_{i=1}^{M} \Gamma(\alpha_i)} \prod_{i=1}^{M} x_i^{\alpha_i - 1}$$

where $\Gamma(x)$ is the gamma function. The LDA algorithm infers $\boldsymbol{\theta}_s \in \mathbb{P}_{k-1}$ for each segment. There are various ways in literature to infer the parameters of the LDA model from data, there are the variational methods (Blei, Ng, and Jordan 2003) and Markov chain Monte Carlo algorithms (Griffiths and Steyvers 2004) and (Darling 2011). $\boldsymbol{\theta}_s$ provides a probability distribution over k topics that measures the probability of a word in the segment s to belong to topic k. Using $\boldsymbol{\theta}_i$ as a low dimensional representation of $\mathbf{s}_i$ that capture how much a segment belongs to a certain topic we devise a clustering step. Since $\boldsymbol{\theta}_i \in \mathbb{P}_{k-1}$ there is no simple way to cluster these points, K-means is only appropriate to data in the euclidean space, so we decided to cluster each segment i based on the mode of $\boldsymbol{\theta}_i$, check figure 3.3 to see the segmentation. The justification of this clustering procedure is to assign segment s its most probable topic which is the mode of $\boldsymbol{\theta}_s$. To perform Markov chain Monte Carlo version of LDA, (Nguyen 2015) library was used. This library implements (Griffiths and Steyvers 2004) method, the collapsed Gibbs sampling and uses 3 hyper-parameters, which are the prior's $\boldsymbol{\eta}$ and $\boldsymbol{\alpha}$, and the number of iterations/samples of the Markov chain Monte Carlo.

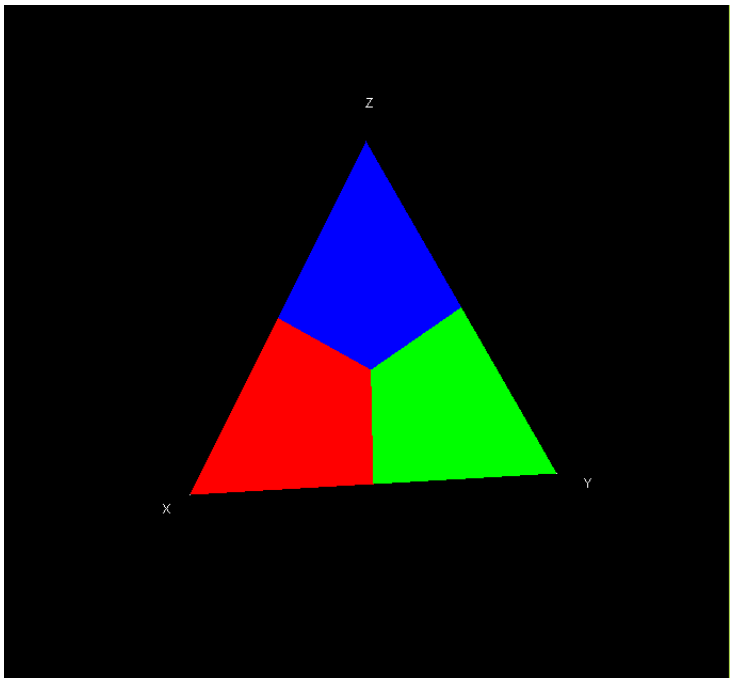


Figure 3.3: Partition of the $\mathbb{P}_{k-1}$ based on the mode of each point

3.5 Summarization and Video Formation

After finding the arcs from clustering of segments, we need to select for each arc a set of segments that summarizes this arc. This summarization process have a time constraint which limits the time of the summary. In the limit case when there is no time constraint, all of the segments of an arc will be part of the summary. In order to solve this problem it was used the LexRank approach described in 2.2.3 using the clustered similarity graph as input of the method. Other approaches could be used such as (Zhu, Goldberg, Van Gael, and Andrzejewski 2007) that also improves diversity in the summary. This could be done in future work. The segmentation of video was done by mapping each word, represented by the rows of X, to the

time interval that this word appeared in subtitle text file. This mapping is a function $h(k) : \mathbb{N} \mapsto I$ where I is the set of positive intervals $\mathbb{R}^2_+$. Having this map and a segment $\mathbf{s}_i$, that defines a interval in the rows of X, we can compute a cut in the video. As an example suppose $\mathbf{s}_I$ is a segment that defines the interval $[i, j], \quad i, j \in \mathbb{N}$, then using the map h, we compute correspondent video interval by:

$$\text{SegmentInterval}(\{i, \ldots, j\}) = \bigcup_{k=i}^{j} h(k).$$

After computing this time interval FFMPEG (Developers 2016) is used to segment the video file. We present the summary of each arc by retrieving a set of segments corresponding to that arc, or a single video is produced for each arc/topic where this video is the result of concatenation of all segments in a cluster.

3.6 Practical considerations

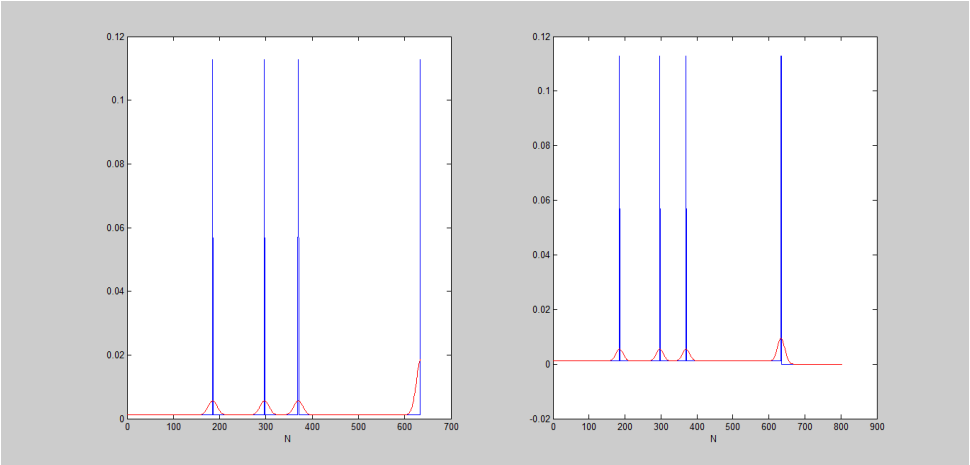


Figure 3.4: Smoothing process of a column of X using the clamped laplacian matrix(left) and the larger laplacian matrix(right). The error (L2 norm) between the smoothing signals is 0.0262 .

It is important to note that in the segmentation step each document has a different number of words N, when implementing this algorithm we set N as the maximum number of words in the document collection, but each document d has a distinct number of words n_d. The eigenvalues and eigenvectors of L are calculated once and used for every document. This can be justified by adding $\mathbf{0}$ to the rows X (a $n_d \times |V|$ matrix) such that X becomes a $N \times |V|$ matrix unifying all documents length. From experiments this operation doesn't degrade results of smoothing, check figure 3.4, also when doing segmentation these zeros are ignored.

4 Results

In this section we will discuss results of our system. First we discuss an empiric parameter selection. Next segmentation results are presented. Topic/arc finding results are shown and finally summarization of arcs is discussed.

4.1 Parameters selection

An important topic is the tuning of the various parameters of the summarizing model. This system has the following parameters:

- ρ : which is the parameter responsible for identifying stop words as described in section 3.2.
- $t(k)$: which is the time $X(t)$ is diffused/smoothed. We saw that this parameters is dependent of the number of eigenvectors k used in the spectral representation (see section 3.3).
- $d : \mathbb{P}_{|V|-1} \times \mathbb{P}_{|V|-1} \to \mathbb{R}$: is the distance function that measures distance between histograms, this function is used to construct the similarity graph (see 3.4).
- κ : number of neighbors that is select when construction the similarity graph in 3.4.
- σ : parameter responsible for computing similarity between nodes, in the κ-nearest-neighbor graph (check equation 3.24).
- T : number of topics/arcs of the series.
- Diffusion Clustering parameters:
  - $\tau(k)$: similar to $t(k)$, τ is used to diffused the heat kernel in a sense measure the range of neighbors the method takes into account. $\tau(k)$ is also dependent of the number of eigenvectors k used in the spectral representation of the graph(check 3.4.2).
- LDA parameters:
  - $\boldsymbol{\eta}$: the prior parameter of the $\boldsymbol{\beta}_i \sim \text{Dirichlet}(\boldsymbol{\eta}, |V|)$.
  - $\boldsymbol{\alpha}$: the prior parameter of the $\boldsymbol{\theta}_s \sim \text{Dirichlet}(\boldsymbol{\alpha}, T)$.
  - # Samples : number of samples of Markov chain Monte Carlo algorithm.

Starting with ρ, we devised two simple experiments that gives an intuition about how to choose this parameter. The first thing we did was to change ρ from zero to one and compute the average ratio between number of stop words and the size of the vocabulary. From 3.2 we know that for $\rho \geq 0.5$ this ratio is one. From figure 4.1 we see that when $r(\rho) = \frac{\#\text{StopWords}}{|V|} > 0.1$ its growth rate increases very fast. Based just on this experiment we could say that ρ should be about $0 < \rho < 0.3$, since for $\rho > 0.3$ more than 40% of the words are stop words. The other experiment was to compute the average Jaccard Index of the set of stop words from different series as ρ is varying. The result can be visualized in 4.2, it is clear that most of the common stop words happens when $\rho < 0.1$. Based on this simple experiments we used values of $\rho < 0.1$.

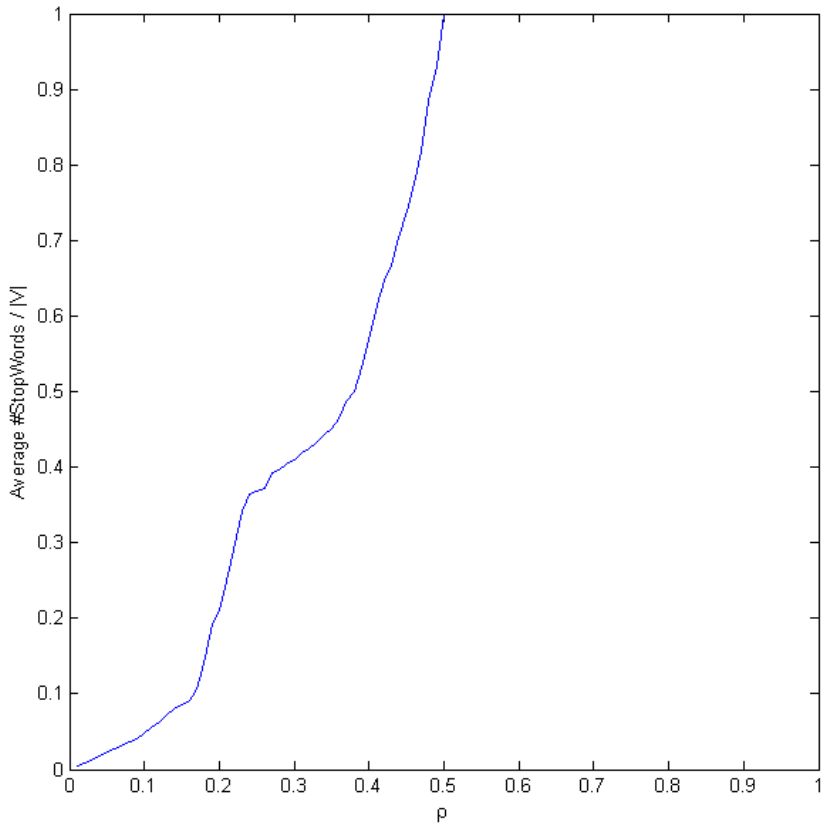


Figure 4.1: $\frac{\#\text{StopWords}}{|V|}$ by ρ of four different series.

The parameters κ and σ were chosen according to (Von Luxburg 2007), where $\kappa = \log(|S|)$ and σ is mean distance of a point to its k-th nearest neighbor.

The number of topics T will be a user defined parameter. The LDA parameters are the default according to (Nguyen 2015) with twice the samples of the default.

The distance function used by default will be the simplex distance (Lebanon et al. 2005)(see 3.4). We use this distance function as default because it is a metric (unlike the cosine distance) made for histogram data (unlike the euclidean metric).

The selection of parameters $\tau(k)$ and $t(k)$ will be discussed in the next sections.

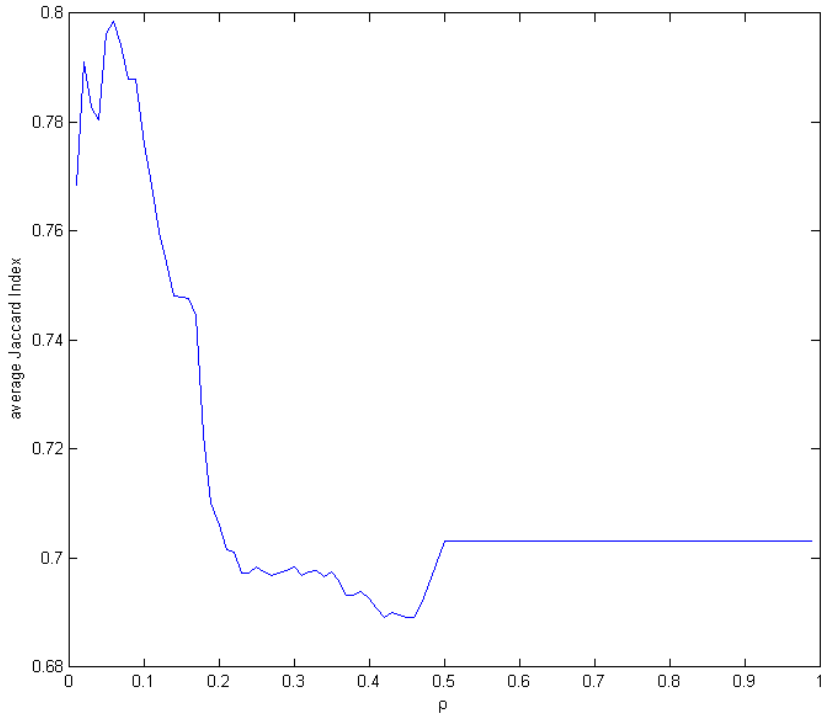


Figure 4.2: Average Jaccard Index of stop words sets of different series by ρ.

4.2 Segmentation Evaluation

To evaluate our segmentation method we will first study some properties of this method in comparison with a trivial method. This preliminary study will build us an intuition for the value k in $t(k)$. Then we use our method in a benchmark segmentation data set and compared with the results of (Alemi and Ginsparg 2015).

In order to understand our method and choose a good values of k we device two basic experiments. As described above $k \in \{1, \ldots, N\}$, in our experiments we sample 50 values of k for 8 episodes in 4 series(32 episodes). We want segments that have a good topic cohesion, that is neighbor segments should have high distance otherwise should be in the same segment. Other aspect is the number of words in each segment, a segment shouldn't have all words in a document or just one word. In the first experiment the number of segments $|S|$ was computed, from this we can check percentage of words in each segment. Assuming that each segment has the same number of words (this assumption is in general false, but is used to simplify the analysis) we compute the percentage of words in a segment (number of words in segment divided by N). This percentage of words is easy to compute from our assumption, which is that $|S|\#\text{wordsInSegment} = N$, from simple algebra the percentage of words in a segment is $\frac{1}{|S|}$. The second experiment we compute distances between adjacent segments for a number of samples of k. With this we found a distribution of distances as k is varying. From 4.3a and 4.3b we verify a trade of between the difference between the neighbor segments and the percentage of words in each segment. As k increases the difference between neighbor segments

increases rapidly where the percentage of words in each segment decreases in a fast rate. We want some kind of equilibrium between the two, good amount of words per segment and good difference between neighbor segment, but what good means ? From some experiments the value $\frac{k}{N} = 0.04$, gave good visual results. This value validates in a way our results, since we have 10% of words per segment and a average cosine distance between neighbor segments of 0.2 with some variance when $k/N = 0.04$. For comparison we made the same experiment with a simple segmentation function. This function segments a document with equal sizes parameterized by p, where the size of the segment is equal to $(1-p)N$. From 4.4a and 4.4b we notice that the results are worst than the scale space methods since at 10% words per segment the average cosine distance between neighbor segments is less than 0.1 with less variance.

To demonstrate the validity of the segmentation process we used a benchmark segmentation data set C99 presented in (Choi 2000) and the segmentation evaluation metric P_k defined in (Beeferman, Berger, and Lafferty 1999). To avoid misunderstandings we will use P_q as P_k. We compared results with the paper (Alemi and Ginsparg 2015). The P_q metric is simply described in (Alemi and Ginsparg 2015) as : "The P_q metric, captures the probability for a probe composed of a pair of nearby elements (at constant distance positions $(i, i+q)$) to be placed in the same segment by both reference and hypothesized segmentation. In particular, the P_q metric counts the number of disagreements on the probe elements". Mathematically it its computed as :

$$P_q = \frac{1}{N-q} \sum_{i=1}^{N-q} \left[\delta_{\text{ref}}(i, i+q) \neq \delta_{\text{hyp}}(i, i+q)\right]$$

Where $\delta_{\text{ref}}(i, i+q) = 1$ if word in i and $i+q$ belong to the same segment and 0 otherwise. Also $[\text{true}] = 1$ and 0 otherwise. Note that smaller P_q value indicate better segmentation. In order to compare our results with (Alemi and Ginsparg 2015) we decided to use the same value of q as in (Alemi and Ginsparg 2015), which is :

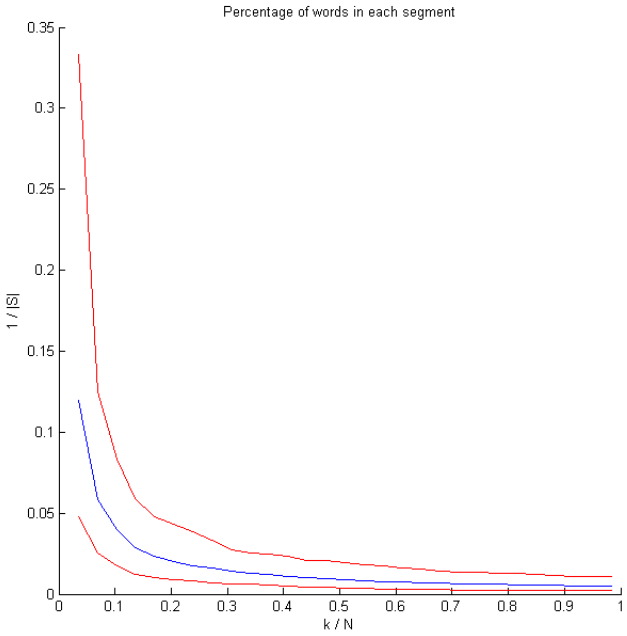
$$q = \frac{1}{2}\frac{\#\text{elements}}{\#\text{segments}} - 1 \tag{4.1}$$

After selecting a value for q we computed P_q using the system segmentation method described in section 3.3 using various values of k/N (stop words were removed using a list of stop words). We computed the average value of P_q for the whole C99 data set for each k/N. Since we already know from the first experiments that for high values of k/N the segmentation performs poorly, we just sample values from 0 to 0.1, as shown in figure 4.5.

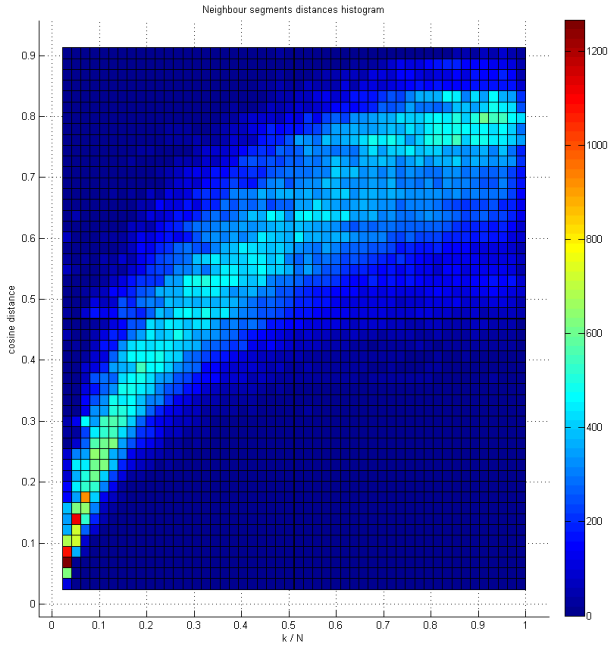
By comparing the best average P_q value of 0.1486 (with k/N = 0.032) with the average P_q value of 0.1339 of the OC99 method in (Alemi and Ginsparg 2015), we see that even though our method performs worst than the OC99 its value is very close (with relative error $\frac{|0.1339-0.1486|}{0.1339} \simeq 0.11$). It is worth noticing that OC99 method is a supervised technique since it uses word embeddings trained from (JeffreyPennington and Manning) and our method is unsupervised.

4.3 Cluster Evaluation

In order to compare the different ways of clustering the segments into arcs/topics, few experiments were made. Those experiments include measuring intra and inter cluster distance distribution and the number of segments per cluster. The experiments were made for each

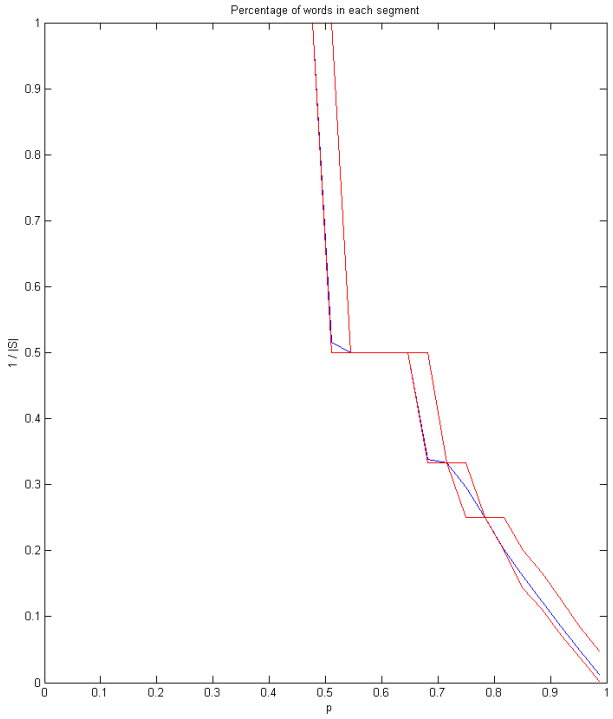


(a)

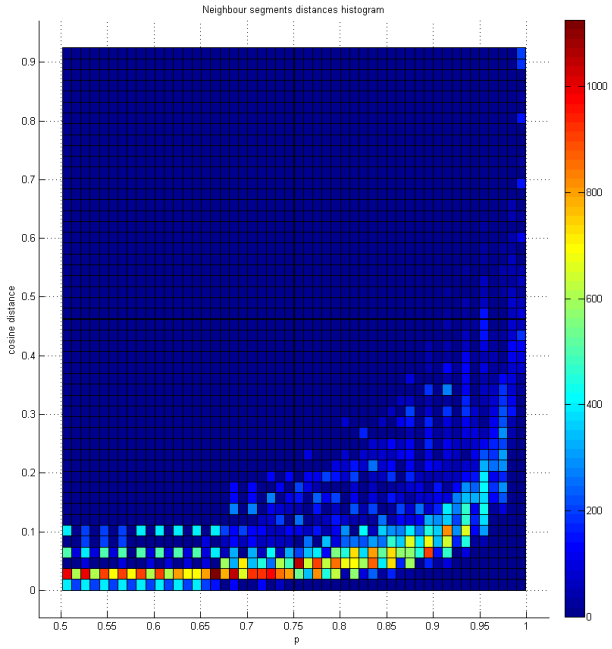


(b)

Figure 4.3: (a)Percentage of words in each segment as function o k/N, blue curve is the average value while the red curves represent the min and max values. (b) Neighbor segment distance distribution using the scale space method as a function of k/N.



(a)



(b)

Figure 4.4: (a)Percentage of words in each segment as function o p, blue curve is the average value while the red curves represent the min and max values. (b) Neighbor segment distance distribution using the equal size segmentation method as a function of p.

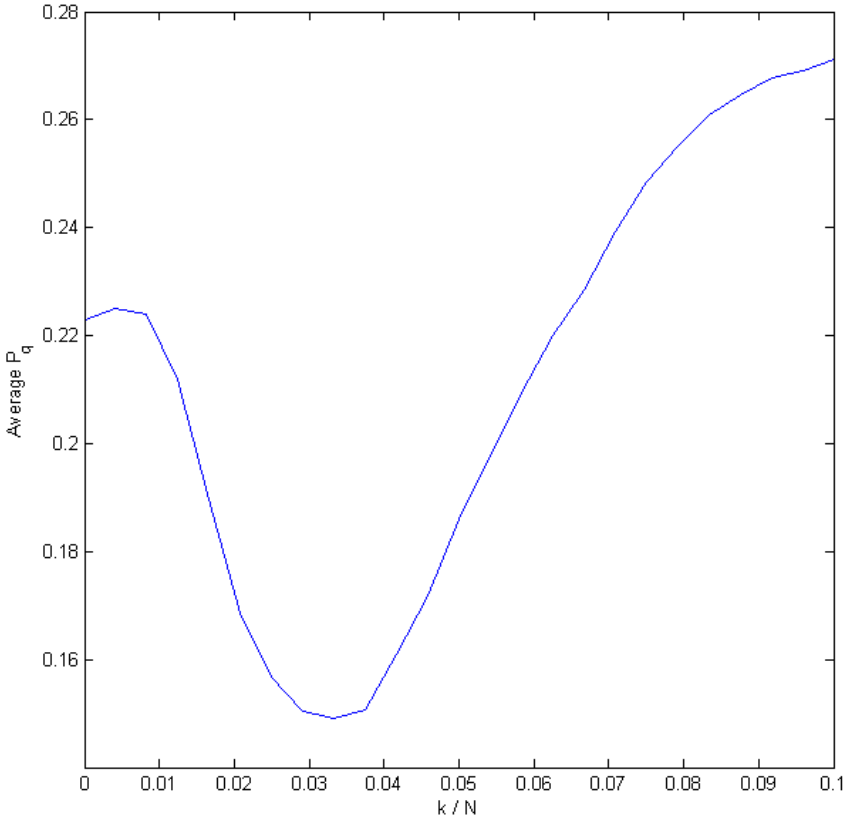


Figure 4.5: Average P_q of C99 data set for various values of k/N.

clustering method in section 3.4 with four series. The number of episodes for each series are presented in table 4.1.

Table 4.1: Series data set

Series	Number of episodes	Average episode length (min)
Over The Garden Wall	10	11
Mr Robot	10	45
Breaking Bad	62	45
Battle Star Galactica	73	42

We choose these experiments because in a sense good arcs/topics have similar segments in the same cluster and dissimilar segments between different clusters. Therefore a good partition of segment should have low intra-cluster distance and high inter-cluster distance. It is important to note that these variables are dependent of the number of segments in each cluster. Consider the following scenario where a clustering algorithm creates two cluster, one of them has just one segment while the other has the remainder of segments, then the intra/inter-cluster distance would be biased in a sense there isn't enough data to compute inter-cluster distance. A good clustering algorithm in this setting should be able to distribute segments between clusters/topics such that number of segments per topic isn't "too uneven". There are

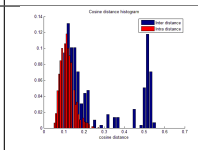
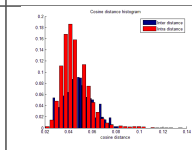
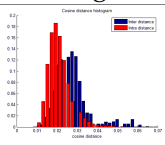
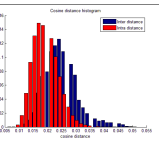
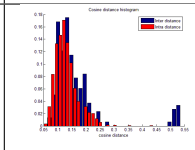
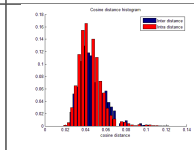
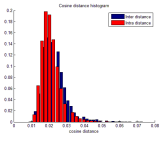
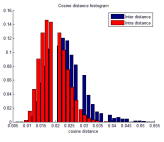
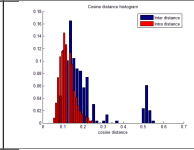
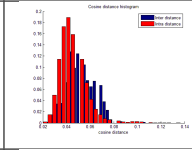
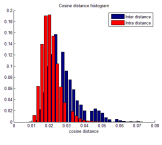
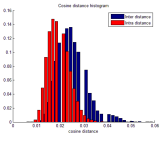
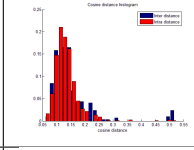
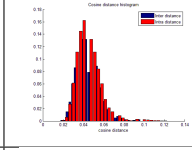
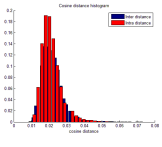
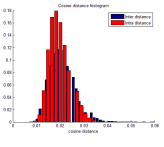
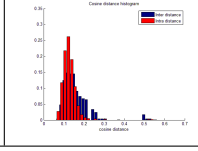
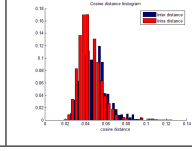
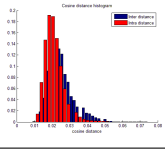
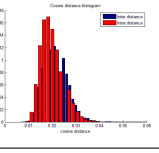
a lot of ways one could measure if the number of segments per topic is uniform, one of these ways is to compute the probability of a segment to be part of topic/arc i, let it be called $p_i = \frac{n_i}{|S|}$, where n_i is the number of segments in topic i. Then we propose the entropy of the p_i as the measure of how much uniform is the distribution of segments between topics, note that the uniform distribution has maximum entropy.

To study the clustering process cluster intra and inter distance distributions were calculated in the following way : All the $\sum_{i=1}^{T} \frac{n_i(n_i-1)}{2}$ intra-clusters distances were sampled, while only $3\sum_{i=1}^{T} n_i$ inter-cluster distances samples were taken, due to a more complex sampling problem and higher number of samples(there are $\sum_{i=1}^{T-1} n_i \left(\sum_{j=i+1}^{T} nj\right)$ inter cluster distances).

The parameters of the first experiment were the default with $t = -\frac{\log(0.1)}{\lambda_{\lfloor 0.04N \rfloor}}$, $\tau = -\frac{\log(0.1)}{\lambda_{\lfloor 0.04|S| \rfloor}}$ (note that the first lambda corresponds to eigenvalues mentioned in section 3.3, while the second lambda corresponds to the eigenvalues of section 3.4) and a stop words list were used to remove stop words instead of the entropy algorithm (3.2). The number of topics $T = 6$.

In table 4.2 histograms of the first experiments are presented, it can be noticed that the first three methods (Diffusion, Spectral Norm and Spectral NNorm) have higher discrepancy between intra-cluster distance histogram and the inter-cluster distance histogram than the remainder methods. A small series like OverTheGardenWall has even bimodal distribution in various methods. To evaluate this in an approximate way we compute the mean(expected value) of the inter and intra cluster distance distribution.

Table 4.2: Histograms of the first experiment

Method/Series	OverTheGardenWall	MrRobot	BreakingBad	BattleStarGalactica
Diffusion				
Spectral Norm				
Spectral NNorm				
Spectral Ng et al				
LDA				

From images 4.6a and 4.7b we see that spectral non-normalized and diffusion have high

inter-cluster distance but have low entropy. Spectral norm method have great inter-cluster distance with high average entropy.

In the second experiment we removed stop words using algorithm in 3.2 with $\rho = 0.05$. Figures 4.7a and 4.7b shows very similar results to the the first experiment with a slight improvement in the Lda method.

We noticed that the κnn-graphs build for the spectral and diffusion clustering methods with the default parameters were creating highly connected graphs. In order to build graphs with more structure we decided to set $\kappa = 1$ and set the other parameters to the same as the first experiment. The Lda method didn't change with these new parameters, as expected. Diffusion, spectral non-normalized and (Ng, Jordan, Weiss, et al. 2002) have very close distance distributions means with (Ng, Jordan, Weiss, et al. 2002) still having highest average entropy although there was a great improvement in the average entropy of the diffusion method (see 4.8a and 4.8b).

We also tried the previous experiment with $\rho = 0.05$. Average entropy of every method have increased with diffusion and Lda methods having the highest inter-cluster mean distance(check 4.9a and 4.9b).

We notice that segment in the summary were very large so we decreased t to $t = -\frac{\log(0.1)}{\lambda_{\lfloor 0.05N \rfloor}}$. Also τ was increased to $\tau = -\frac{\log(0.1)}{\lambda_{\lfloor 0.033|S| \rfloor}}$. In the fifth experiment Lda method performed better when looking to both average segment distribution entropy and mean cosine inter-cluster distance. All the other methods improve on their mean inter-cluster distance but degrade in their average entropy (see 4.9a and 4.9b).

4.4 Qualitative Results

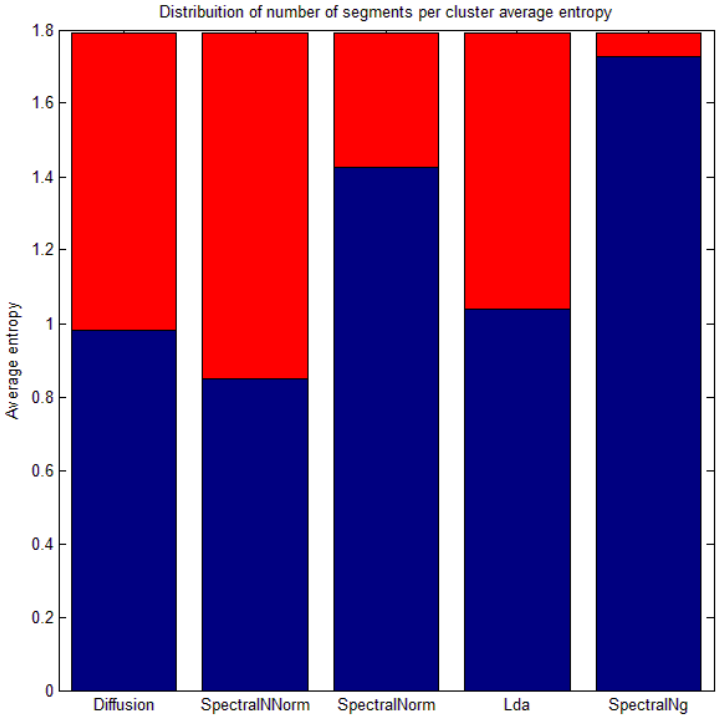
In order to evaluate the quality of the summaries generated by the program, we decided to take a small series like "Over The Garden Wall" and study how the summaries relate to the TV series. A good plot summary of this series is found in (Wikia 2017) and its content is on Appendix A, we compare this summary with the generated arc summaries. One thing to note is that a small number of segments in an arc normally produces worst summary since it is more difficult understand the underlying topic, also there are less relevant segments to chose. By analyzing the results form the first experiment we see that (Ng, Jordan, Weiss, et al. 2002) gave best results, capturing concepts of the Adelaide journey arc, or Beatrice plus going home topic and Sara's and Jason arc. Methods like normalize and non-normalized spectral clustering could isolate episodes which aren't so related to the other like episode 8 where Wirt saves Lorna or episode 3 where Gregory saves the school from being closed. More or less most of the methods capture The Beast, Sara's and Jason arcs, also they could cluster segments from the same episode together. Something we notice in all the methods of the first experience is that there is some consistence in the clusters formed specifically they produce similar clusters (cluster with more or less the same segments). Only a few clusters have this similarity. This pattern happened with cluster with significantly number of segments. There are some failures, namely some segments that appear in the summary of an arc that doesn't seem to have nothing in common with that arc.

When removing stop words using the entropy method(3.2), there were more segments that didn't belong to the cluster or seem unrelated to the rest of the segments. Still segments form

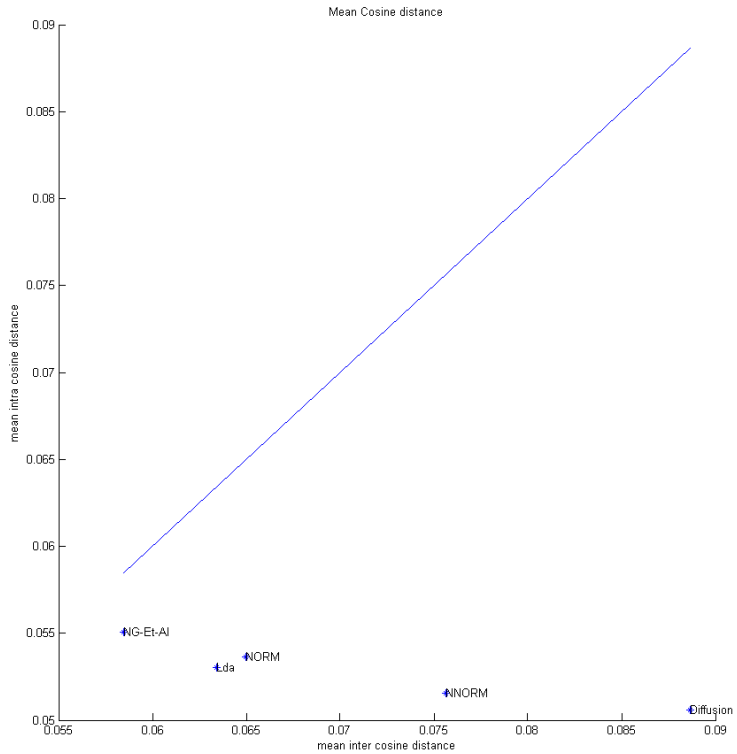
the same episode pattern remain and new topics like the lamp + beast + woodsman appear. Lda method created a more uniform distribution on the segments per topic/arc.

The third experiment there were some similarities with the first experiment. Overall this experiment performed better than the second experiment with less errors in the clustering process leading to more coherent summaries as the first experiment. Diffusion and Spectral non-normalized method performed better than the first experiment. We notice that there was a cluster(in some of the methods) that its topics was just some key words that were repeated in multiple episode like a song of going to Adelaide house and the words ”run,run run”, which is kind of a silly topic. (Ng, Jordan, Weiss, et al. 2002) was still the best method and spectral normalized performed worst than the first experiment. We noted many of the methods produced similar topics.

This section is subjective and so a real user study should be conducted perhaps in future work.

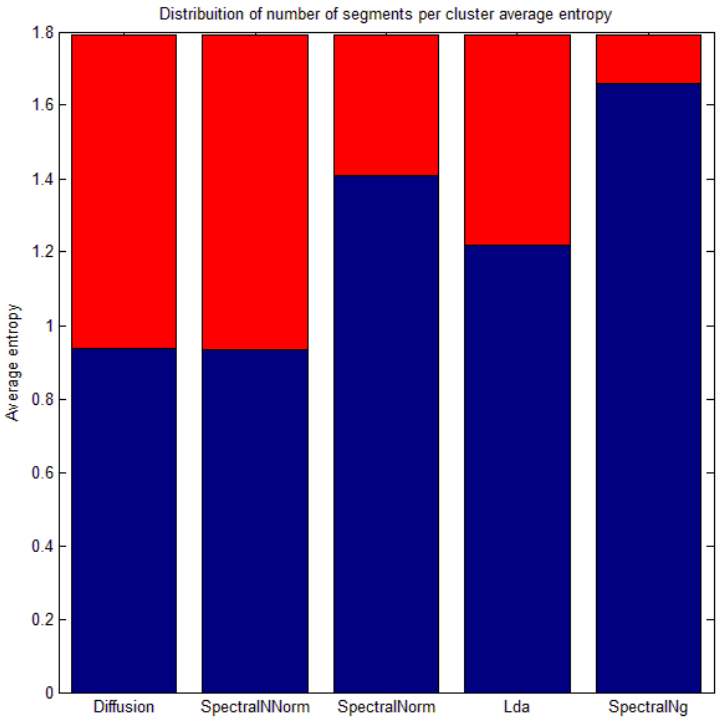


(a)

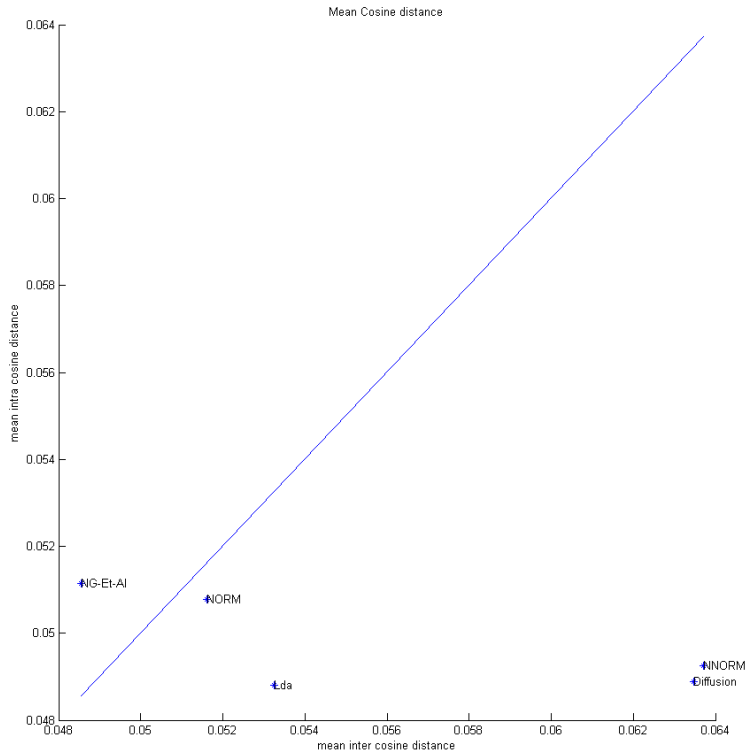


(b)

Figure 4.6: (a)Average entropy of the second experiment. (b) Mean cosine distance plot of the first experiment, line represents function $y = x$.

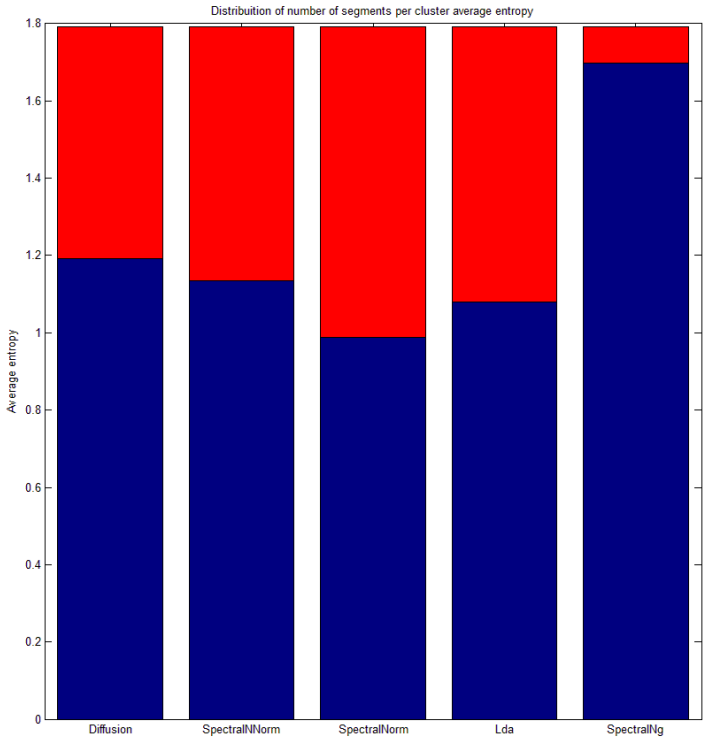


(a)

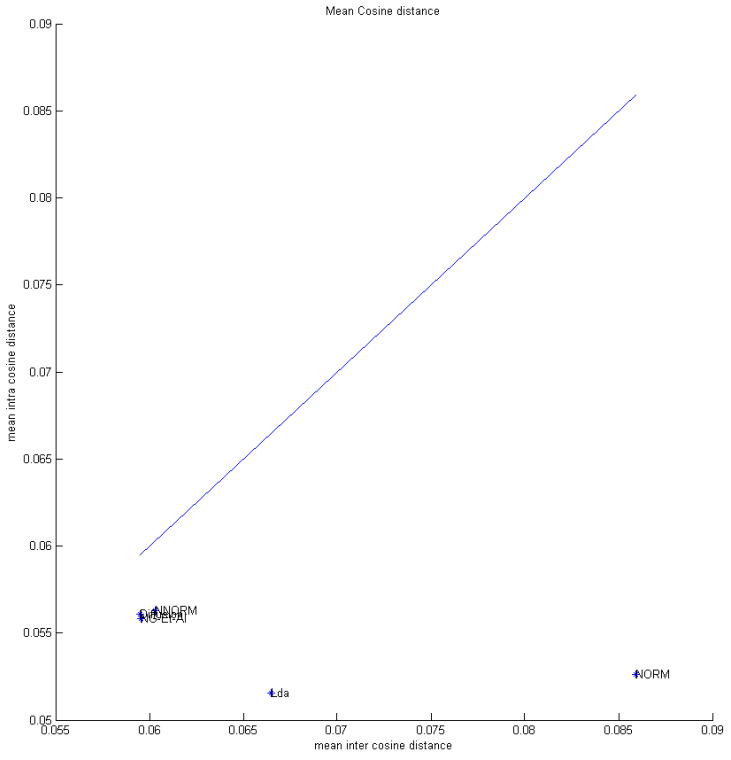


(b)

Figure 4.7: (a)Average entropy of the second experiment. (b) Mean cosine distance plot of the second experiment, line represents function $y = x$.

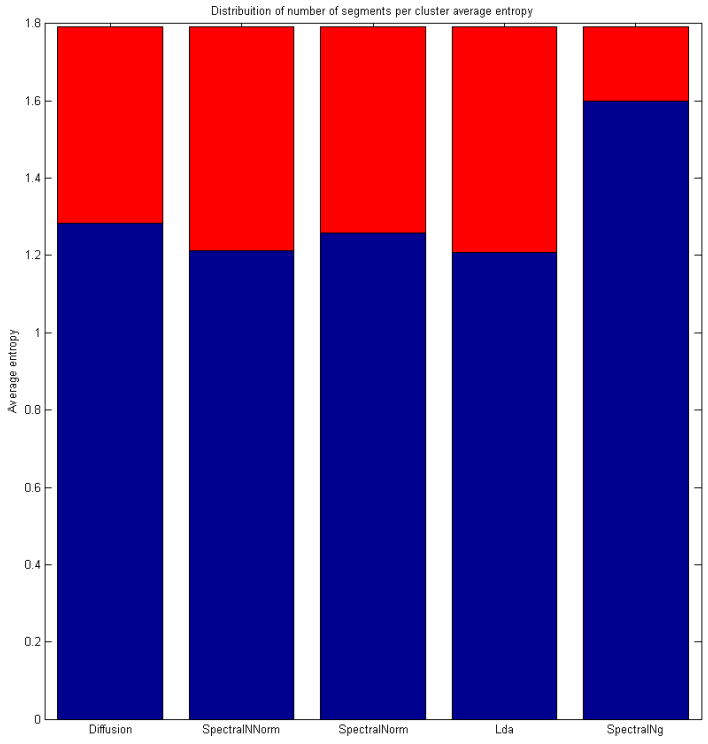


(a)

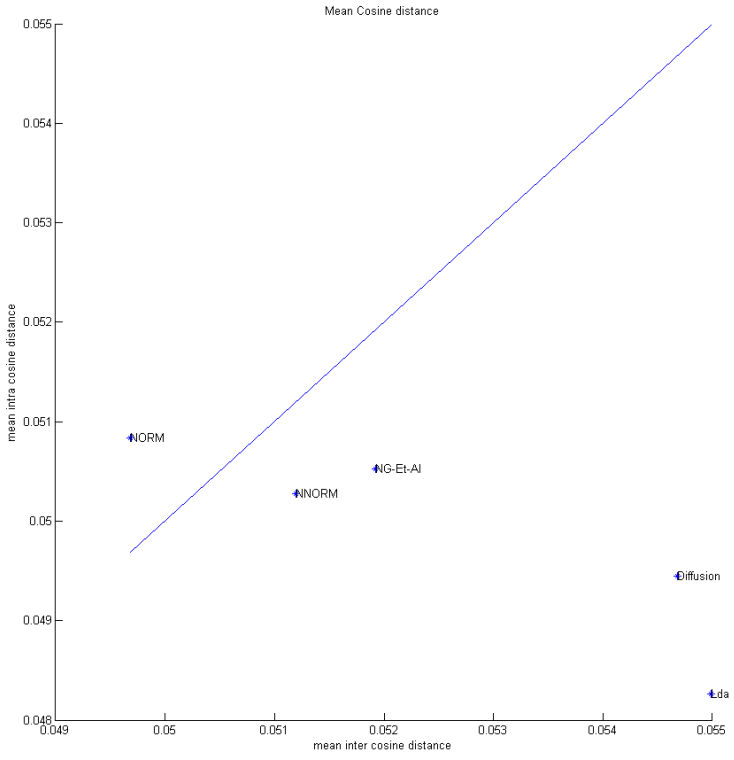


(b)

Figure 4.8: (a)Average entropy of the third experiment. (b) Mean cosine distance plot of the third experiment, line represents function $y = x$.

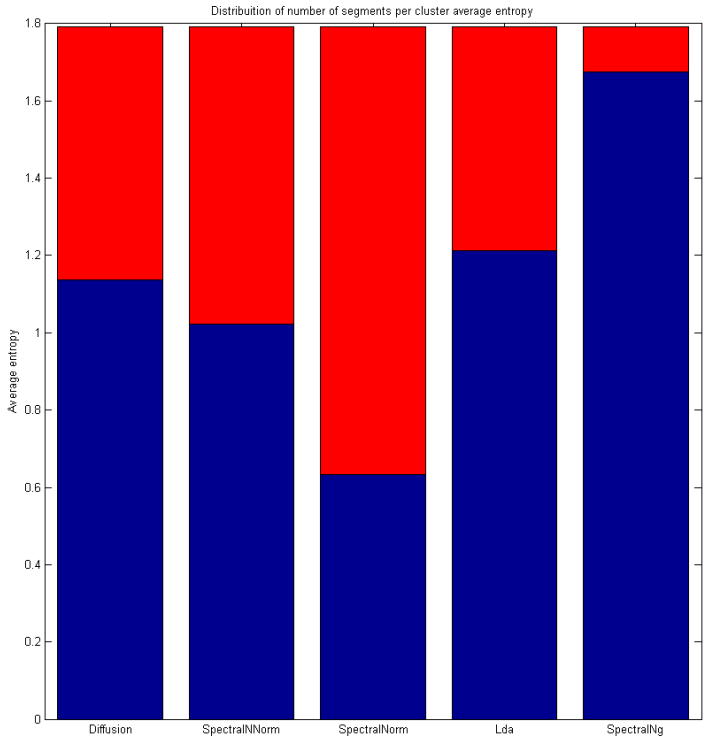


(a)

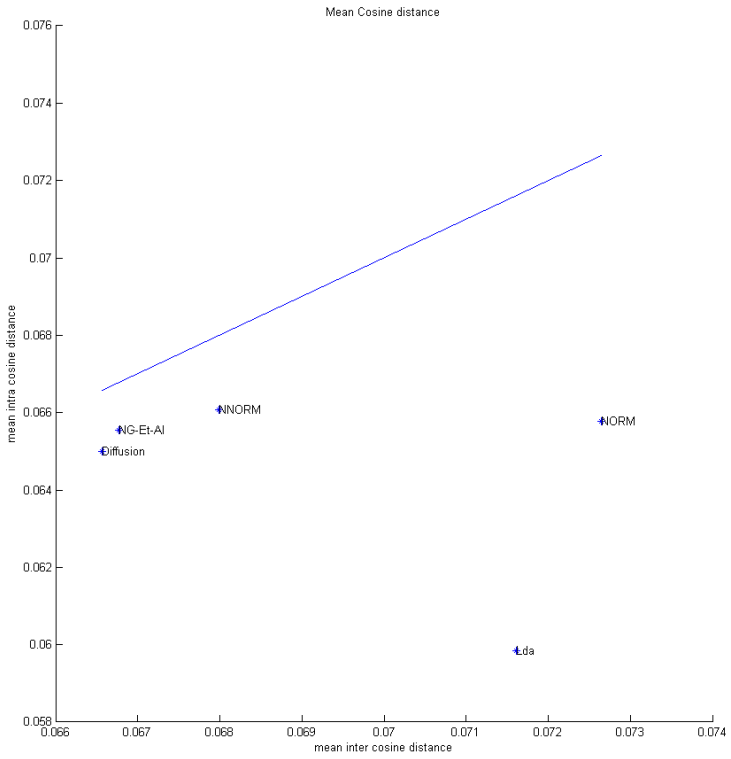


(b)

Figure 4.9: (a)Average entropy of the fourth experiment. (b) Mean cosine distance plot of the fourth experiment, line represents function $y = x$.



(a)



(b)

Figure 4.10: (a)Average entropy of the fifth experiment. (b) Mean cosine distance plot of the fifth experiment, line represents function $y = x$.

5 Conclusion

5.1 Conclusions and Future Work

In this thesis, we designed and implemented a system capable of segment, cluster arcs of the series and summarize those arcs based on just subtitle information. This method creates arc summaries that take in to account the story of the series namely because it uses textual information of the whole series instead of just using information of one episode like (Tsoneva, Barbieri, and Weda 2007) or just using visual and audio cues of general purpose video summarization (2.1) without semantic context. We reviewed several text summarization in section 2.2, since these methods are at the core of our summarization method. In order to cluster all episodes of a series to arcs, a good semantic episode segmentation is needed. We address this problem by using well established theory, namely scale-space segmentation and Lowbow curves. We showed that those two methods were equivalent and reformulated this methods using spectral graph theory improving its time complexity (section 3.3). This reformulation creates a compressed sequential text model (in part responsible for the decrease in time complexity) that should be further studied in future work, one improvement could be a dimensionality reduction of the word space while using this model. We tested this method on benchmark segmentation task with good results (4.2). We studied several clustering methods for topic/arc finding in a certain detail and conducted several intrinsic experiments to evaluate its performance (3.4 and 4.3 respectively). For future work there should be a user study to assess the output of this algorithm, since it was made for human consumption. We focus mainly on spectral clustering algorithms for topic/arc finding mainly because intuitively they connect segments to each under a geometrical graph structure called knn-graph(able to capture curved data manifolds) then this graph is embedded in a space such that similar segments to stay together, while Lda method(which was also studied) creates topics by embedding segments in a low dimensional linear space that may not capture curvature of the data. Some of our results favor spectral clustering methods since they produce close to uniform distribution of the segments per topic, with still high inter-cluster mean distance and low intra-cluster mean distance. The focus on this thesis on spectral graph methods comes because of the applicability of this theory in all phases of the process, such as random walks ((Chung 2007)) for the summarization, spectral clustering(and diffusion clustering) for the arc/topic finding and segmentation process with scale-space segmentation. This suggest for future work to unify all of these similar but different methods based on spectral graph theory to produce better models of text and documents. Finally while building the knn-graph of the segments some segments made connections to other segments that weren't similar but their distance was small, this implies that there is an issue on the documents distance functions. Future work on distance functions for document should be address, perhaps using other language models such as (Mikolov, Chen, Corrado, and Dean 2013).

Bibliography

Alemi, A. A. and P. Ginsparg (2015). Text segmentation based on semantic word embeddings. *arXiv preprint arXiv:1503.05543*.

Beeferman, D., A. Berger, and J. Lafferty (1999). Statistical models for text segmentation. *Machine learning 34*(1), 177–210.

Belkin, M. and P. Niyogi (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation 15*(6), 1373–1396.

Bindel, D. (2011, November). Lecture notes in matrix computations.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Blei, D. M., A. Y. Ng, and M. I. Jordan (2003). Latent dirichlet allocation. *Journal of machine Learning research 3*(Jan), 993–1022.

Bobenko, A. I., J. M. Sullivan, P. Schröder, and G. M. Ziegler (2008). *Discrete differential geometry*. Springer.

Boyd, S. and L. Vandenberghe (2004). *Convex optimization*. Cambridge university press.

Burkardt, J. (2013). Laplacian the discrete laplacian operator. https://people.sc.fsu.edu/~jburkardt/f77_src/laplacian/laplacian.html.

Carbonell, J. and J. Goldstein (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 335–336. ACM.

Choi, F. Y. (2000). Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pp. 26–33. Association for Computational Linguistics.

Chung, F. (2007). The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences 104*(50), 19735–19740.

Chung, F. R. (1997). *Spectral graph theory*, Volume 92. American Mathematical Soc.

Coifman, R. R. and S. Lafon (2006). Diffusion maps. *Applied and computational harmonic analysis 21*(1), 5–30.

Crane, K., C. Weischedel, and M. Wardetzky (2013). Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow. *ACM Trans. Graph. 32*.

Darling, W. M. (2011). A theoretical and practical implementation tutorial on topic modeling and gibbs sampling. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 642–647.

De Goes, F., S. Goldenstein, and L. Velho (2008). A hierarchical segmentation of articulated bodies. In *Computer graphics forum*, Volume 27, pp. 1349–1356. Wiley Online Library.

DeMenthon, D., V. Kobla, and D. Doermann (1998). Video summarization by curve simplification. In *Proceedings of the sixth ACM international conference on Multimedia*, pp. 211–218. ACM.

Demirtas, K., I. Cicekli, and N. K. Cicekli (2011). Summarization of documentaries. In *Computer and Information Sciences*, pp. 105–108. Springer.

Developers, F. (2016). ffmpeg tool (version be1d324) [software]. http://ffmpeg.org/.

Dumais, S. T. (2004). Latent semantic analysis. *Annual review of information science and technology 38*(1), 188–230.

Erkan, G. and D. R. Radev (2004, December). Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res. 22*(1), 457–479.

Furini, M. and V. Ghini (2006). An audio-video summarization scheme based on audio and video analysis. In *IEEE CCNC*.

Gong, Y. and X. Liu (2000). Video summarization using singular value decomposition. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, Volume 2, pp. 174–180. IEEE.

Gong, Y. and X. Liu (2001). Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 19–25. ACM.

Griffiths, T. L. and M. Steyvers (2004). Finding scientific topics. *Proceedings of the National academy of Sciences 101*(suppl 1), 5228–5235.

JeffreyPennington, R. and C. Manning. Glove: Global vectors for word representation.

Koenderink, J. J. (1984). The structure of images. *Biological cybernetics 50*(5), 363–370.

Kokiopoulou, E., J. Chen, and Y. Saad (2011). Trace optimization and eigenproblems in dimension reduction methods. *Numerical Linear Algebra with Applications 18*(3), 565–602.

Lafon, S. and A. B. Lee (2006). Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE transactions on pattern analysis and machine intelligence 28*(9), 1393–1403.

Lawler, G. F. (2010). *Random walk and the heat equation*, Volume 55. American Mathematical Society.

Lebanon, G. et al. (2005). *Riemannian geometry and statistical machine learning*. Carnegie Mellon University, Language Technologies Institute, School of Computer Science.

Lebanon, G., Y. Mao, and J. Dillon (2007, December). The locally weighted bag of words framework for document representation. *J. Mach. Learn. Res. 8*, 2405–2441.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory 28*(2), 129–137.

Lyon, R. (1987). Speech recognition in scale space. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87.*, Volume 12, pp. 1265–1268. IEEE.

Ma, Y.-F., L. Lu, H.-J. Zhang, and M. Li (2002). A user attention model for video summarization. In *Proceedings of the tenth ACM international conference on Multimedia*, pp. 533–542. ACM.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Money, A. G. and H. Agius (2008). Video summarisation: A conceptual framework and survey of the state of the art. *Journal of Visual Communication and Image Representation 19*(2), 121–143.

Needleman, S. B. and C. D. Wunsch (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology 48*(3), 443–453.

Newman, M. (2010). *Networks: an introduction*. Oxford university press.

Ng, A. Y., M. I. Jordan, Y. Weiss, et al. (2002). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems 2*, 849–856.

Nguyen, D. Q. (2015). jLDADMM: A Java package for the LDA and DMM topic models. http://jldadmm.sourceforge.net/.

Olver, P. J. Introduction to the calculus of variations.

Page, L., S. Brin, R. Motwani, and T. Winograd (1999). The pagerank citation ranking: Bringing order to the web.

Peletier, M. A. (2011). Energies, gradient flows, and large deviations: a modelling.

Perona, P. and J. Malik (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence 12*(7), 629–639.

Radev, D., A. Winkel, and M. Topper (2002). Multi document centroid-based text summarization. In *ACL 2002*.

Radev, D. R., H. Jing, and M. Budzikowska (2000). Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization*, pp. 21–30. Association for Computational Linguistics.

Ramsay, J. O. (2006). *Functional data analysis*. Wiley Online Library.

Rudin, L. I., S. Osher, and E. Fatemi (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena 60*(1), 259–268.

Saloff-Coste, L. The heat kernel and its estimates.

Salton, G., A. Wong, and C.-S. Yang (1975). A vector space model for automatic indexing. *Communications of the ACM 18*(11), 613–620.

Shuman, D. I., S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine 30*(3), 83–98.

Slaney, M. and D. Ponceleon (2001). Hierarchical segmentation using latent semantic indexing in scale space. In *icassp*, Volume 1, pp. 1437–1440.

Steinberger, J. and K. Jezek (2004). Using latent semantic analysis in text summarization and summary evaluation. In *Proc. ISIM'04*, pp. 93–100.

Tsoneva, T., M. Barbieri, and H. Weda (2007, September). Automated summarization of narrative video on a semantic level. pp. 169–176. IEEE.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing 17*(4), 395–416.

Wagner, D. and F. Wagner (1993). Between min cut and graph bisection. In *International Symposium on Mathematical Foundations of Computer Science*, pp. 744–750. Springer.

Wikia (2017). Over the garden plot summary. http://over-the-garden-wall.wikia.com/wiki/Over_the_Garden_Wall.

Wilbur, W. J. and K. Sirotkin (1992). The automatic identification of stop words. *Journal of information science 18*(1), 45–55.

Witkin, A. (1984). Scale-space filtering: A new approach to multi-scale description. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, Volume 9, pp. 150–153. IEEE.

Yang, S. (2012). A scale-space theory for text. *CoRR abs/1212.2145*.

Zhu, X., A. B. Goldberg, J. Van Gael, and D. Andrzejewski (2007). Improving diversity in ranking using absorbing random walks. In *HLT-NAACL*, pp. 97–104.

Appendices

A Over The Garden Wall plot summary

In this section we present a plot summary of the Over the garden wall series from (Wikia 2017).

"The central story involves Wirt (Elijah Wood) and Gregory (Collin Dean), two half-brothers who find themselves lost in a strange, magical forest called The Unknown. They come upon a grim Woodsman whose job involves cutting down "Edlewood" trees that he grinds into

oil in his mill, which he then uses to keep his lantern constantly alight. He warns the boys of The Beast, a terrible creature that haunts the woods in search of lost children. The Woodsman tells the boys to head North, look for a town, and escape the woods if they can.

The boys—along with Greg's pet frog (which is constantly being called new names)—head down the path and meet the acerbic Beatrice, a talking bluebird. Because Greg helps her get free of a bush, she agrees to help the children get home by taking them to see Adelaide of the Pasture, the Good Woman of the Woods. On the way to Adelaide's Pasture, the group have various adventures where they meet numerous denizens of The Unknown.

First they stumble upon the harvest festival in Pottsfield where they are found guilty of minor crimes by the fantastical mayor, Enoch, and are sentenced to a few hours of manual labor in the fields. The brothers then help save a small schoolhouse for animals from financial ruin and a rampaging gorilla. While hitching a ride on a hay cart, they become lost. In search of directions, they enter a creepy tavern populated by narrative archetypes (e.g. the Butcher, the Tailor, the Highwayman, etc). Beatrice is banished by the Tavern Keeper and Wirt is declared initially to be a simple-minded Fool, then the Young Lover, and finally a Pilgrim on a Sacred Journey. When Beatrice is heard screaming in the woods, Wirt (along with Greg and his frog) dash to the rescue on a horse he takes from the stable. They come upon the Woodsman, whom they have come to suspect might be the Beast. Greg is able to collect the unconscious Beatrice and they flee into the forest. The Beast then comes out of the woods to speak to the Woodsman, where it is revealed that the Woodsman keeps the lantern lit because his daughter's spirit is kept alive within it.

Back on the path to Adelaide's Pasture—thanks to the horse, which is able to talk and is named Fred—the group is invited into the sprawling mansion of the wealthy and eccentric tea magnate, Quincy Endicott. Beatrice reveals to Wirt that she was once human that she and her family were cursed after she threw a rock at a bluebird. Meanwhile, Greg helps Endicott face his fears and unite with his true love, whom he believes to be a ghost. Afterwards, Fred stays behind with Endicott while the two brothers and Beatrice ride the steamboat ferry (which is entirely populated by well-dressed society frogs) on the way to Adelaide's Pasture. They camp for the night, but Beatrice sneaks off to speak with Adelaide alone. It turns out that Beatrice was in cahoots with Adelaide to bring her a human child in exchange for a magical pair of scissors needed to turn the cursed family back into humans. Beatrice has changed her mind and offers herself instead, but Adelaide captures the boys as they burst in. Beatrice is able to dispatch Adelaide and free the brothers, who both run off into the woods without her.

Wirt and Greg (and his frog) then manage to save Lorna, a kind girl who has become possessed by a terrifying spirit that has devoured countless victims unfortunate enough to enter her home. Despite Wirt's newfound courage, he has begun to lose hope and becomes despondent. As both fall asleep under a tree, Gregory accepts the task of getting them home and makes a request to a star to help him be a good leader. In a dream, he is taken to the delightful Cloud City, where he defeats the odious North Wind. In gratitude, the Queen of the Clouds offers Greg a wish. Greg wishes to go home with Wirt, but the Queen explains that the Beast has already claimed Wirt and has started to turn him into an edlewood tree. Greg then wishes to free Wirt by offering himself to The Beast in his brother's place. Wirt awakens to find Greg gone—he chases after him, but falls through ice into a dark, frozen lake.

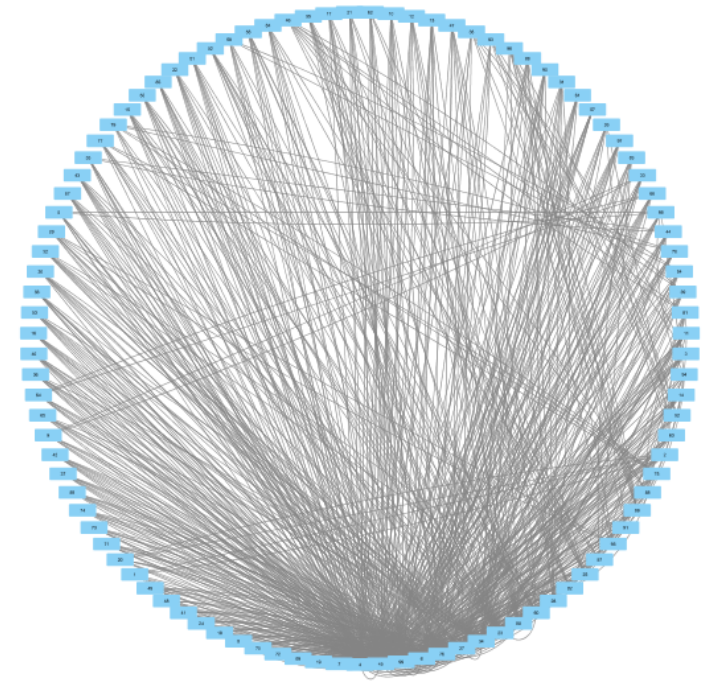
The final two episodes reveal that Wirt and Greg actually live in the modern world. It is Halloween night, which explains why they are dressed as they are (and why Greg has so much candy in Chapter/Episode 1). Wirt has made a tape for a girl named Sara with his poetry and

clarinet playing. After he finds out that another boy is interested in her, Wirt wants to get the tape back to save himself from embarrassment. Wirt and Greg find her with a group of kids at a graveyard, but they all scatter when a police officer drives up and jokingly tells them they are all under arrest. Wirt and Greg attempt to escape over the cemetery's garden wall, landing on railroad tracks on the other side. Greg finds his frog just as a train comes at them. They both roll down the hill and plunge into a body of water, both unconscious.

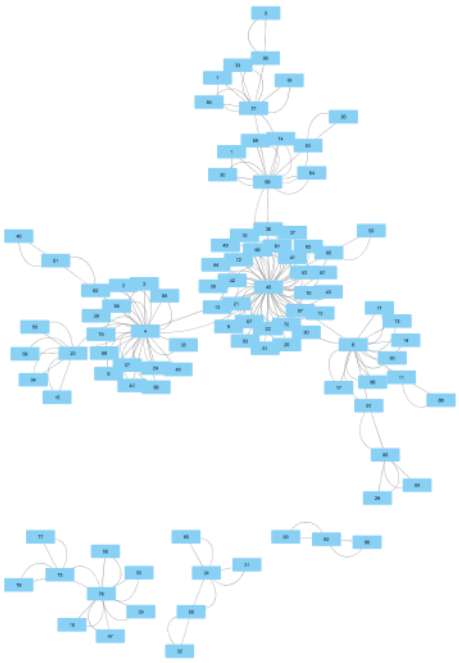
Back in The Unknown, Wirt wakes up surrounded by Beatrice's bluebird family after Beatrice saved him from the frozen lake and goes out to continue his search for Greg. Meanwhile, the Woodsman comes upon The Beast and Gregory, who is starting to be transformed into an edelwood tree. The Woodsman attacks The Beast as Wirt and Beatrice find Greg and attempt to free him. The Beast knocks out the Woodsman and offers to keep Greg's spirit in the lamp if Wirt agrees to keep it lit as the Woodsman did. Wirt realizes that it is, in fact, the spirit of the Beast that is trapped in the lantern and refuses the deal, giving the lantern back to The Woodsman. Wirt frees Greg and gives Beatrice the magical scissors he took from Adelaide's house. They say their fond farewells as The Woodsman blows out the lantern to the angry screams of the Beast.

Back in the modern world, Wirt comes to and drags Greg out of the water. Wirt wakes up in the hospital to the sight of Sara while Greg recaps their adventures in The Unknown to the other kids at the cemetery gathering. The episode ends with a montage showing how things turned out with the inhabitants of The Unknown, including the reunion of the Woodsman with his daughter and Beatrice and her family returned to their human forms."

B Supplementary images



(a)



(b)

Figure 1: (a)κ-nn graph of segments of the Over The Garden Wall series with $\kappa = log|S|$ where S is the set of segments (b) 1-nn graph of segments of the Over The Garden Wall series

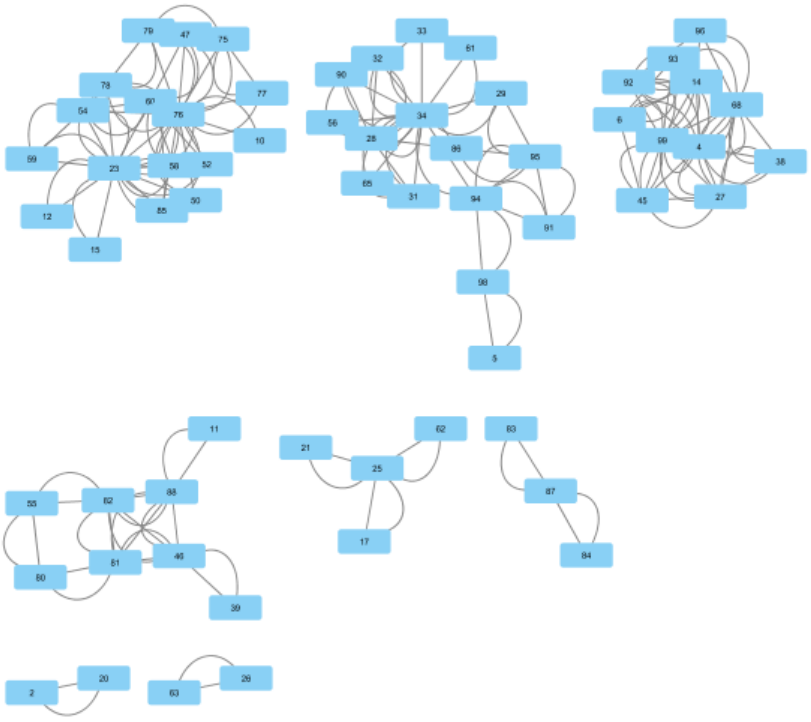


Figure 2: Partition of the graph in figure 1a using spectral Andrew Ng et al method