Arc Summarization of Tv Series

Pedro Cristóvão

***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.**

***Index Terms*—Summarization, TV Series, Information Retrieval, Spectral Graph Theory**

I. 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, [1] 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. 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(extractive) 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 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.

II. Thesis Structure

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

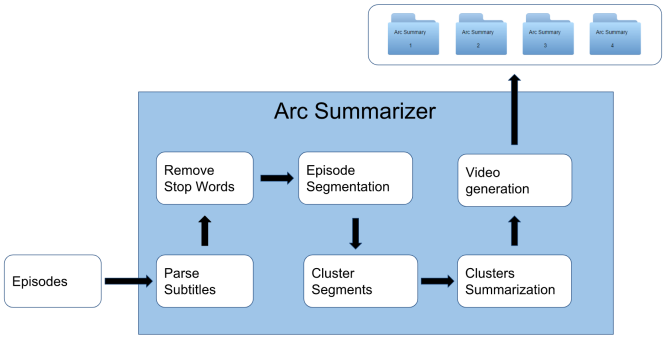


Fig. 1. Fundamental steps of the arc summarizer described in this dissertation

III. 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.

IV. Summarization Algorithms

A. 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, [2] 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 [3], 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.

[4] 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.

[5] 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 [6]. 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.

[1] describes a video summarization algorithm in narrative videos such as movies or series. Their solution was to first make analysis (parsing) of subtitles and the script of a movie or TV series episode, then compute a script-subtitles alignment, using Needleman–Wunsch algorithm [7], 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.

[1] 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.

[8] uses general text summarization algorithms to segment and summarize documentaries, again it focus essentially on one document.

B. 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 [9]. Text is a finite sequence of words with a finite vocabulary.

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

Where V is a vocabulary, here it is assumed to be a set of integers $V = \{1, \dots, |V|\}$. 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

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

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 [10] 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 [10].

1) Centroid Summarization: Centroid summarization is based on the work of [11] 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 [11] and [12].

2) Maximal Marginal Relevance: Maximal Marginal Relevance (MMR) proposed by [13] 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.

3) LexRank: LexRank [14] 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 [15]. 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:

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

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:

$$\mathbf{p}_{t+1} = \left[dU + (1-d)B\right]^T \mathbf{p}_t \tag{2}$$

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) 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 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) and it can be show to be the highest eigenvector of $\left[dU + (1-d)B\right]^T$ (more details see Perron-Frobenius Theorem). Once again similarity could be the cosine similarity.

4) Latent Semantic Analysis: Latent Semantic Analysis (LSA) [16] 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. [17] chooses k sentences, each sentences is chosen to be the max value of a row of V_k^T. [18] 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.

V. 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.

VI. 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.

VII. 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 [19]. We select two approaches to remove stop words. Remove stop words from a list of stop words and a method proposed by [20]. 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

$$H(\mathbf{p}_i) > (1 - \rho) \max\{H(\mathbf{x})\}. \tag{3}$$

Using $P(d = i|w = j)$ we compute its entropy and classify each word as stop if it fulfills 3. 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

$$H(\mathbf{p}_i) > (1 - \rho) \max\{H(\mathbf{x})\} \text{ or } H(\mathbf{p}_i) < \rho \max\{H(\mathbf{x})\}. \tag{4}$$

VIII. 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 we define a segment as a set $I = \{t \in \mathbb{N} : s_1 \leq t \leq s_2\}$. We want to partition a document to semantic cohesive segments with low time complexity. [1] 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. [1] 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 [9], [21] and [22]. 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) [9] 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 [23] for segmentation of 1D signals, later it was generalized to vector values 1D signals [24] and multivariate signals [25] and [26]. Although not mentioning scale-space segmentation, [9] 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 [27], machine learning [28] and functional data analysis [29] to name a few. This problem can formalized as Tikhonov regularization problem. Using a continuous notation for 1D signals it can be defined as :

$$\min_u \int_a^b (f(x) - u(x))^2 \, dx + \lambda \int_a^b u'(x)^2 \, dx, \ \lambda \geq 0 \tag{5}$$

This cost function 5 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 5 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, parameterized by t.

This approach can be formalized as a gradient descend flow of :

$$\int_a^b u'(x)^2 \, dx \tag{6}$$

which is the heat equation flow :

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} &= \frac{\partial^2 u}{\partial x^2} \\ u(x,0) &= f(x). \end{aligned} \tag{7}$$

Note that the cost function is convex since 6 is a squared L^2 norm of u'. From this we conclude that 6 has only one minima, see [30]. Therefore it is proved that the family of functions defined by 7 improves 6 as $t \to \infty$ and it is reaching the global minimum. For more about optimization of functions and gradient flows check [31] and [32]. Note that with the 7 f will get smoother as the parameter t increases. In the limit the solution to 6 is given by the Euler-Lagrange formula to be $u'' = 0$, whose solutions are linear and constant functions. A way to solve 7 is by convolution of f with the heat kernel $H(x,t)$ see [33]. 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 [23] for more information.

[21] and [22] 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. [22] represented a unit of text as a Latent Semantic Index vector and [21] tried many methods such as the simple bag of words and the Latent Dirichlet Allocation vector. The scale-space segmentation is done similar to [23] 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.

[9] introduces the Lowbow and one of its application is text segmentation. Lowbow uses categorical model defined in section IV-B with additive smoothing (or Laplace smoothing). A document is encoded as $N \times |V|$ matrix X where each row is defined as

$$\mathbf{x}_i = \begin{bmatrix} \dots & \frac{\delta_{y_i,j}+c}{1+|V|c} & \dots. \end{bmatrix}^T \tag{8}$$

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 $\gamma : [0,1] \times |V| \mapsto [0,1]$

$$\gamma_\sigma(\mu, j) = \int_0^1 \phi(s,j) K_{\mu,\sigma}(s) \, ds. \tag{9}$$

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 8. Now rather than define a discretization of a continuous problem like 6 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 [34] and [35], 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 interpreted as a graph, 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. Then in order to reduce the differences between adjacent vertices a discrete version of 5 or 6 should be defined.

Let

$$\min_{\mathbf{f}_1,\ldots,\mathbf{f}_N\in\mathbb{P}_{|V|-1}} \sum_{(i,j)\in E} |\mathbf{f}_i-\mathbf{f}_j|^2 = \sum_{k=1}^{|V|}\sum_{(i,j)\in E} ([\mathbf{f}_i]_k-[\mathbf{f}_j]_k)^2 \tag{10}$$

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 10 with X as the initial condition. It is useful to make a change of variable to 10 oprimization problem. This transformation is :

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

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 and D is a diagonal matrix with $D_{ii}=\deg(i)$ and zeros elsewhere. This cost function is re-written as :

$$2\sum_{k=1}^{|V|}\mathbf{g}_k^T(D-A)\mathbf{g}_k = 2\sum_{k=1}^{|V|}\mathbf{g}_k^T L\mathbf{g}_k. \tag{11}$$

Matrix L in literature is called the graph laplacian matrix. Now with 11 a gradient flow can be found in a simpler manner. Let $\mathcal{L}$ be discrete smoothing cost function, then :

$$\mathcal{L}(\mathbf{g}_1,\ldots,\mathbf{g}_{|V|}) = \frac{1}{2}\text{tr}(G^T LG)$$

Where G is a $N\times|V|$ matrix where each columns is given by $\mathbf{g}_k$. Note that the factor $\frac{1}{2}$ does not alter the solution of our problem (11), it is introduced just to simplify calculations. Then the gradient descend is given by

$$\dot{G} = -LG,\ \ G(0)=X. \tag{12}$$

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.

The solution of 12 is,

$$G(t) = \exp(-Lt)X. \tag{13}$$

It can be shown that $L=USU^T$ is a semi-positive definite matrix with positive eigenvalues $\lambda_1=0\leq\lambda_2\cdots\leq\lambda_N$ and orthonormal eigenvectors U [36]. Hence :

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

$$G(t) = U\exp(-St)\hat{X}. \tag{14}$$

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 [35], 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 [37] and [38]. We can still improve performance by noting that :

$$t\to\infty\ , e^{-\lambda_i t}\to 0,\ \forall i>1 \tag{15}$$

and

$$e^{-\lambda_{k+1}t} < e^{-\lambda_k t},\ \ k\in\{1,\ldots,N\}. \tag{16}$$

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

$$X(t)\simeq U_k\exp(-St)_k\hat{X}_k(0). \tag{17}$$

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 15 and 16 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 [9] 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:

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

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

$$|\mathbf{d}_{X(t)}(k)|^2 = \left\langle \mathbf{d}_{X(t)}(k), \mathbf{d}_{X(t)}(k)\right\rangle \tag{18}$$

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 is $O(Nk|V|)$. 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 only takes $O(Nb|V|)$, 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 [35] and there are lots of benefits of using discrete objects instead of discretizations of continuous objects, see [39].

IX. 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. One metric is used in this project:

- *Simplex distance*: defined in [40] to be the geodesics distance between to points in $\mathbb{P}_{N-1}$ using the Fisher information metric. end

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

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 the data. Two of them use the similarity graph as the basis of the cluster procedure while the third algorithm doesn't. The third algorithm uses a standard topic finding algorithm in natural language processing which is the Latent Dirichlet Allocation(LDA) [41], while the other are based in spectral graph theory.

A. 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. 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:

$$w_{ij} = e^{-\frac{d_{ij}^2}{2\sigma^2}}. \tag{19}$$

Where d_{ij} is the distance between segments i and j using the metrics above. An interpretation of the spectral clustering is to find an graph embedding such that similar vertices become close in the embedding. In this space it is easy to find clusters using traditional methods such as K-means. This formalizes in the following optimization problem:

$$\min_{\mathbf{f}_1, \ldots, \mathbf{f}_K} \sum_{k=1}^{K} \mathbf{f}_k^T (D - W) \mathbf{f}_k = \text{tr}(F^T L F), \text{ s.t } F^T F = I \tag{20}$$

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 11 but here W is used instead of the adjacency matrix. Note that each row of F corresponds to the coordinates of a vertex in the embedding. Also the constraint $F^T F = I$ is given such that the solution of 20 is not the trivial solution where all vertices have the same coordinate. 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([36]). This changes the cost function to :

$$\min_{F} \text{tr}(F^T L F), \text{ s.t } F^T D F = I \tag{21}$$

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 21 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 21 becomes :

$$\min_{H} \text{tr}(H^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} H), \text{ s.t } H^T H = I. \tag{22}$$

Using this transformation the solution becomes easier to compute since optimization of 22 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 [36] for discussion). After H is calculated F is discovered by $F = D^{-\frac{1}{2}} H$.

There is also the [42] 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.

B. Diffusion Distance Clustering

Based on the diffusion distance of [43], we perform diffusion distance clustering algorithm to find the arcs of a series. [20] 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 [44], 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 works similar to spectral clustering, it tries to find an embedding of a graph such that similar vertex stay

close in the embedding. In this embedding it will be easier to use standard clustering algorithms such as K-means. The difference between spectral clustering and diffusion clustering is that instead of constraining the coordinates of the graph to the space of orthogonal vectors it will do a gradient descent on the spectral clustering energy and stop when a good embedding is achieved. The energy that this method minimizes is :

$$\min_F \text{tr}(F^T L F) \tag{23}$$

and the gradient descend equation is defined as :

$$\dot{F} = -LF \tag{24}$$

$$F(0) = I$$

By similar arguments as section VIII we find that the solution is given in approximate way as :

$$F(t) = U_k \exp(-St)_k U_k^T$$

Diffusion clustering does not take F as the answer, instead uses a reduce dimensional representation of F given by $\exp(-St)_k U_k^T$. This representation has the property that the euclidean distance between two vertices is the diffusion distance. The diffusion distance can be written (check [45]) in continuous setting as:

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

Where $\mathcal{H}_t(x,y)$ is the continuous heat kernel on a manifold. In the discrete setting it translates to :

$$\mathcal{D}_t^2(x,z) = |H(t)\mathbf{e}_x - H(t)\mathbf{e}_z|^2 \tag{25}$$

Which can be approximated with just k eigenvectors as demonstrated in equation 17

$$\mathcal{D}_t^2(x,z) \approx |\exp(-St)_k U_k^T \mathbf{e}_x - \exp(-St)_k U_k^T \mathbf{e}_z|^2. \tag{26}$$

As one can see this is just the euclidean distance between two points in the embedding space. Hence by applying K-means a graph partition is obtained. 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 VIII. Such selection of t will be further discussed later in the text. This relation of the heat equation and computation of distances was also explored by [46] who also uses the heat kernel to compute geodesic distances of surfaces.

C. 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) [41]. 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

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

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 [41] and Markov chain Monte Carlo algorithms [47] and [48]. $\boldsymbol{\theta}_s$ 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 $\boldsymbol{\theta}_i$. 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, [49] library was used. This library implements [47] 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.

X. 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 IV-B3 using the clustered similarity graph as input of the method. Other approaches could be used such as [50] 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 $[i, j]$, $i, j \in \mathbb{N}$, 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 [51] 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.

XI. 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.

XII. 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 VII.
- $t(k)$: 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 VIII).
- $d : \mathbb{P}_{|V|-1} \times \mathbb{P}_{|V|-1} \to \mathbb{R}$: is the distance function that measures distance between histograms, this function is used to construct the similarity graph (see IX).
- κ : number of neighbors that is select when construction the similarity graph in IX.
- σ : parameter responsible for computing similarity between nodes, in the κ-nearest-neighbor graph (check equation 19).
- 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 IX-B).
- 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 4 we know that for $\rho \geq 0.5$ this ratio is one. From figure 2 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 3, it is clear that most of the common stop words happens when $\rho < 0.1$. Based on this simple experiments we used values of $\rho < 0.1$.

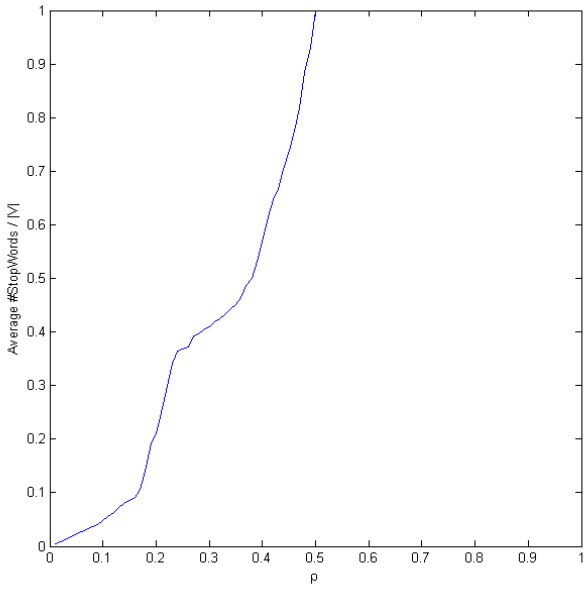


Fig. 2. $\frac{\#\text{StopWords}}{|V|}$ by ρ of four different series.

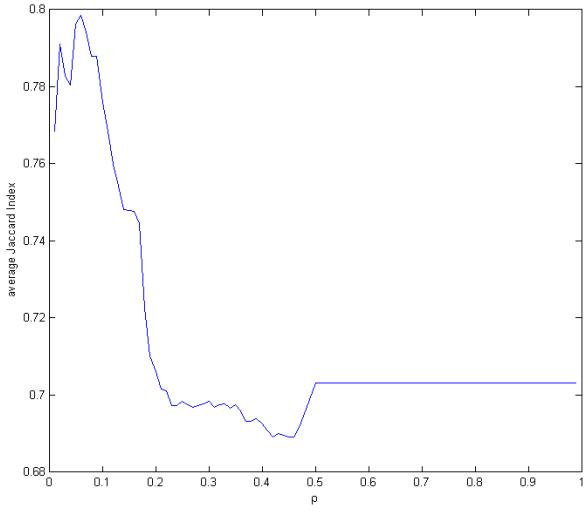


Fig. 3. Average Jaccard Index of stop words sets of different series by ρ.

The parameters κ and σ were chosen according to [36], 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 [49] with twice the samples of the default.

The distance function used by default will be the simplex distance [40](see IX). We use this distance function as default because it is a metric (unlike the cosine distance) made for histogram data (unlike the euclidean metric).

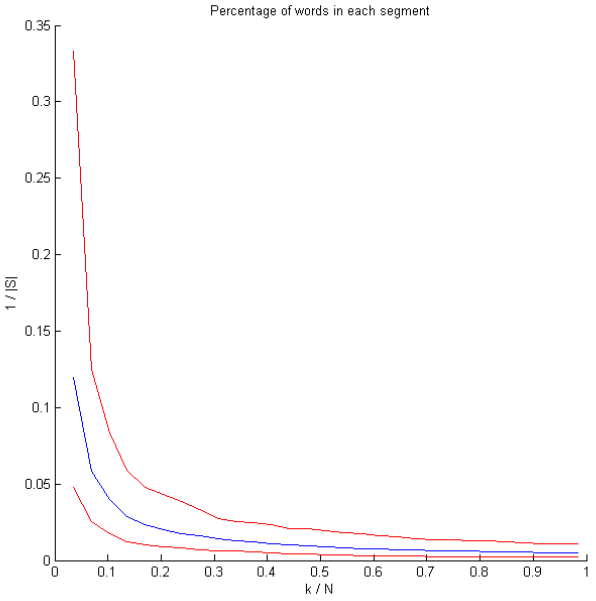
The selection of parameters $\tau(k)$ and $t(k)$ will be discussed in the next sections.

XIII. Segmentation Evaluation

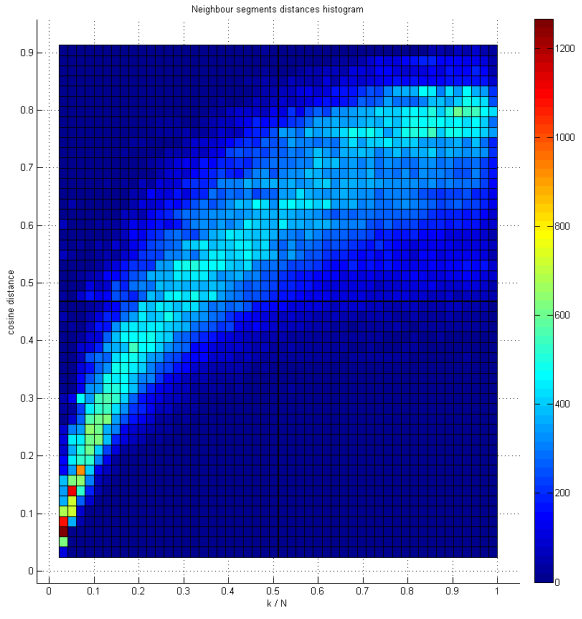
To evaluate our segmentation method we will first study some properties of this method in comparison with a trivial method. This preliminary study will build us an intuition for the value k in $t(k)$. Then we use our method in a benchmark segmentation data set and compared with the results of [52].

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 4a and 4b 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 5a and 5b 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 [53] and the segmentation evaluation metric P_k defined in [54]. To

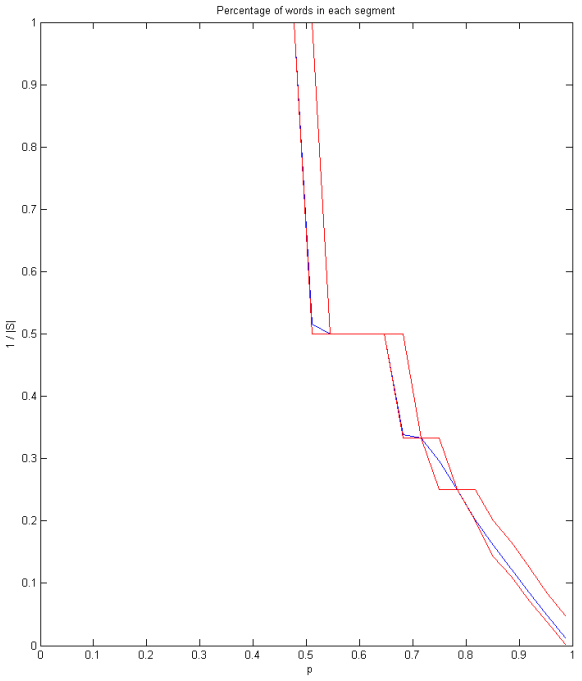


(a)

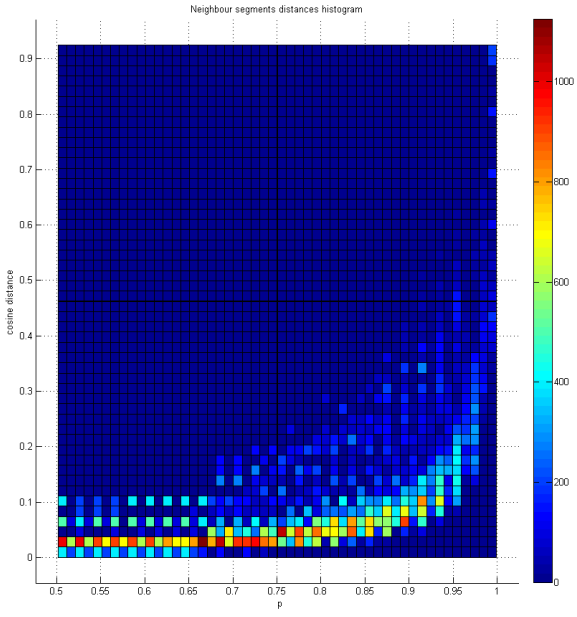


(b)

Fig. 4. (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)

Fig. 5. (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.

avoid misunderstandings we will use P_q as P_k. We compared results with the paper [52]. The P_q metric is simply described in [52] 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 :

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

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 [52] we decided to use the same value of q as in [52], which is :

$$q = \frac{1}{2} \frac{\#\text{elements}}{\#\text{segments}} - 1 \tag{27}$$

After selecting a value for q we computed P_q using the system segmentation method described in section VIII 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 6.

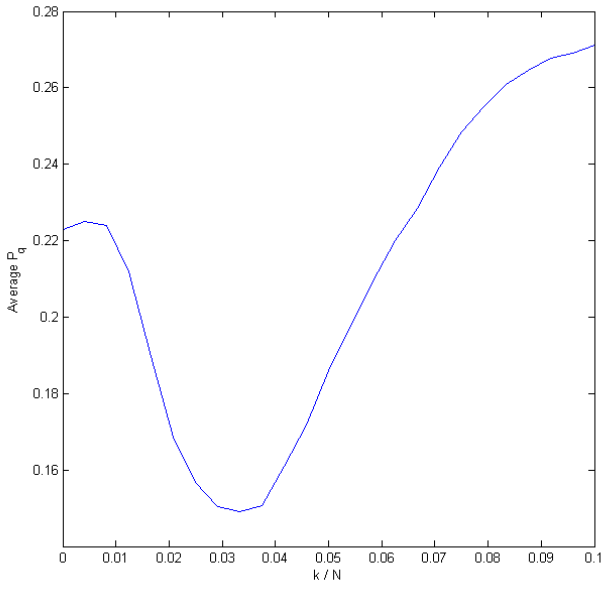


Fig. 6. Average P_q of C99 data set for various values of k/N.

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 [52], 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 [55] and our method is unsupervised.

XIV. 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 clustering method in section IX with four series. The number of episodes for each series are presented in table I.

TABLE I
SERIES DATA SET

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

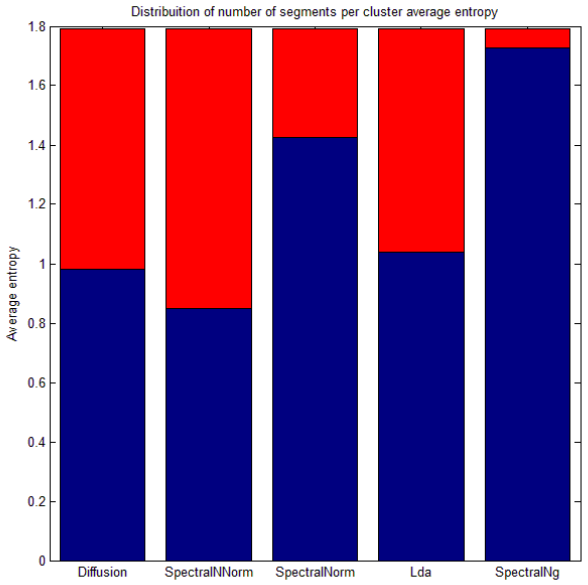
We choose these experiments because in a sense good arcs/topics have similar segments in the same cluster and dissimilar segments between different clusters. Therefore a good partition of segment should have low intra-cluster distance and high inter-cluster distance. It is important to note that these 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).

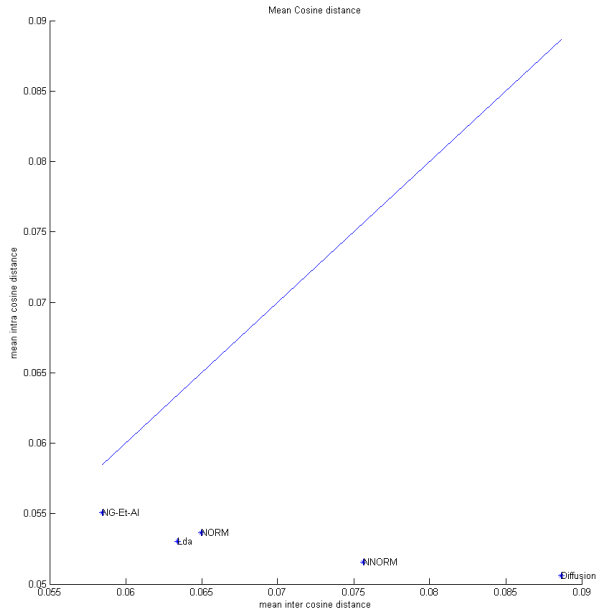
The parameters of the first experiment were the default with $t = -\frac{\log(0.1)}{\lambda_{\lfloor 0.04N \rfloor}}$, $\tau = -\frac{\log(0.1)}{\lambda_{\lfloor 0.04|S| \rfloor}}$ (note that the first lambda corresponds to eigenvalues mentioned in section VIII, while the second lambda corresponds to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

To evaluate this in an approximate way we compute the mean(expected value) of the inter and intra cluster distance distribution.

From images 7a and 8b 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.



(a)

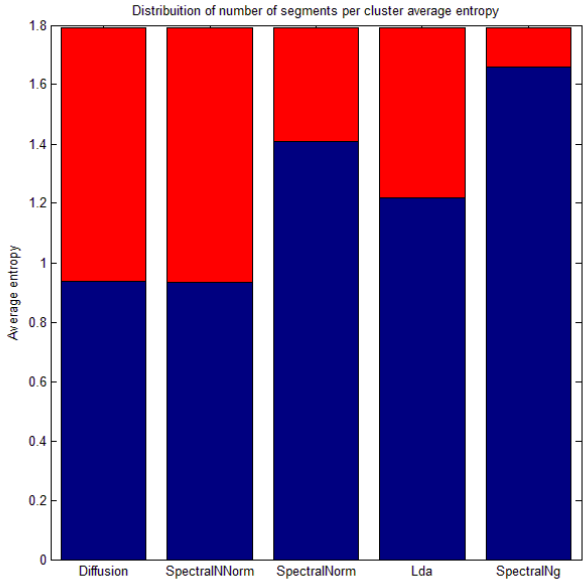


(b)

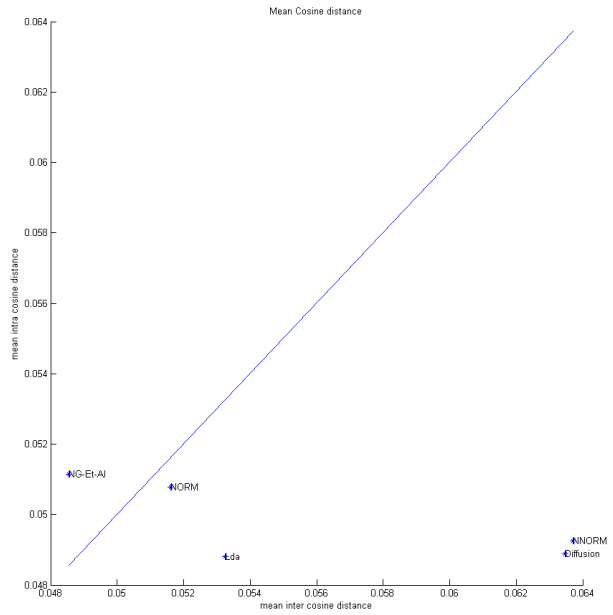
Fig. 7. (a)Average entropy of the second experiment. (b) Mean cosine distance plot of the first experiment, line represents function $y = x$.

In the second experiment we removed stop words using algorithm in VII with $\rho = 0.05$. Figures 8a and 8b 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

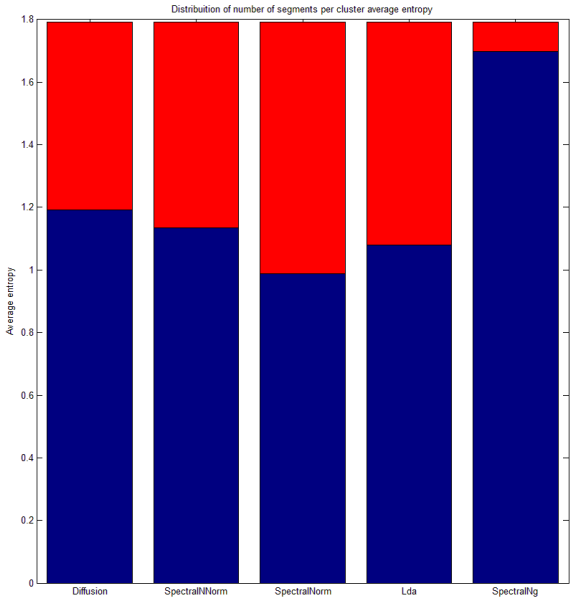


(a)

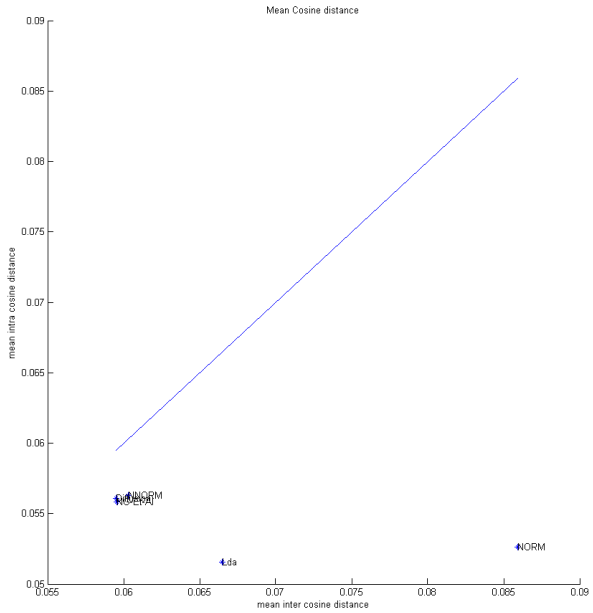


(b)

Fig. 8. (a)Average entropy of the second experiment. (b) Mean cosine distance plot of the second experiment, line represents function $y = x$.



(a)



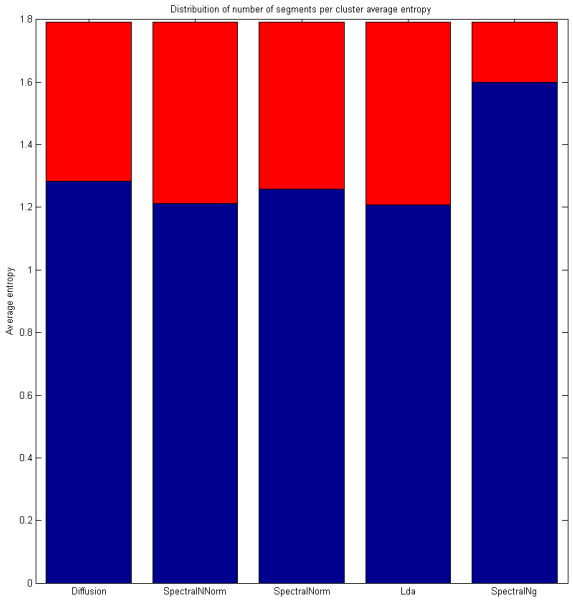
(b)

Fig. 9. (a)Average entropy of the third experiment. (b) Mean cosine distance plot of the third experiment, line represents function $y = x$.

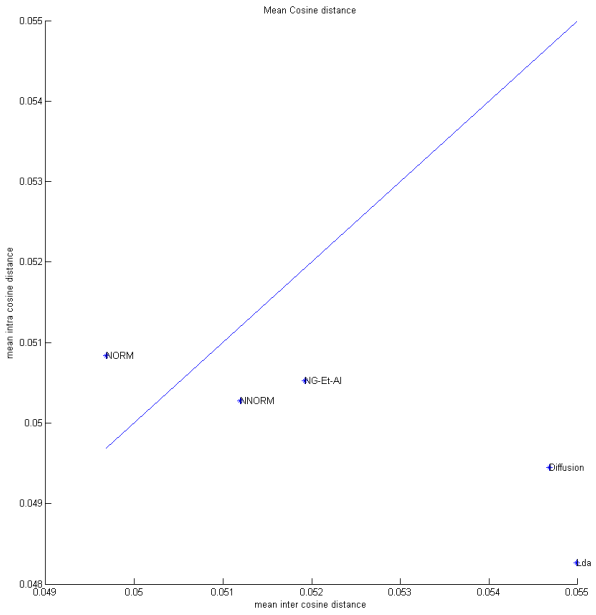
with more structure we decided to set $\kappa = 1$ and set the other parameters to the same as the first experiment. The Lda method didn't change with these new parameters, as expected. Diffusion, spectral non-normalized and [42] have very close distance distributions means with [42] still having highest average entropy although there was a great improvement in the average entropy of the diffusion method (see 9a and 9b).

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 10a and 10b).

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

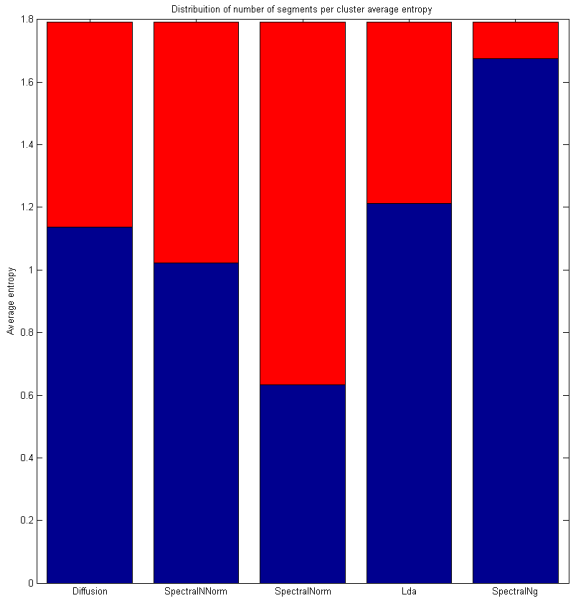


(a)

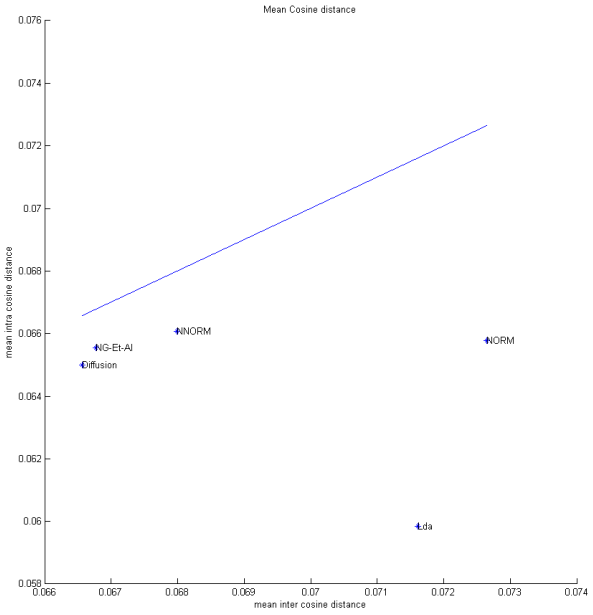


(b)

Fig. 10. (a)Average entropy of the fourth experiment. (b) Mean cosine distance plot of the fourth experiment, line represents function $y = x$.



(a)



(b)

Fig. 11. (a)Average entropy of the fifth experiment. (b) Mean cosine distance plot of the fifth experiment, line represents function $y = x$.

the other methods improve on their mean inter-cluster distance but degrade in their average entropy (see 10a and 10b).

XV. Conclusion

A. 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 [1] or just using visual and audio cues of general purpose video summarization (IV-A) without semantic context. We reviewed several text summarization in section IV-B, 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 VIII). 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 (XIII). We studied several clustering methods for topic/arc finding in a certain detail and conducted several intrinsic experiments to evaluate its performance (IX and XIV 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 ([56]) 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 [57].

REFERENCES

[1] T. Tsoneva, M. Barbieri, and H. Weda, "Automated summarization of narrative video on a semantic level." IEEE, Sep. 2007, pp. 169–176.

[2] M. Furini and V. Ghini, "An audio-video summarization scheme based on audio and video analysis," in *IEEE CCNC*, 2006.

[3] Y. Gong and X. Liu, "Video summarization using singular value decomposition," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2. IEEE, 2000, pp. 174–180.

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

[5] D. DeMenthon, V. Kobla, and D. Doermann, "Video summarization by curve simplification," in *Proceedings of the sixth ACM international conference on Multimedia*. ACM, 1998, pp. 211–218.

[6] A. G. Money and H. Agius, "Video summarisation: A conceptual framework and survey of the state of the art," *Journal of Visual Communication and Image Representation*, vol. 19, no. 2, pp. 121–143, 2008.

[7] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[8] K. Demirtas, I. Cicekli, and N. K. Cicekli, "Summarization of documentaries," in *Computer and Information Sciences*. Springer, 2011, pp. 105–108.

[9] G. Lebanon, Y. Mao, and J. Dillon, "The locally weighted bag of words framework for document representation," *J. Mach. Learn. Res.*, vol. 8, pp. 2405–2441, Dec. 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1314498.1314576

[10] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[11] D. R. Radev, H. Jing, and M. Budzikowska, "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*. Association for Computational Linguistics, 2000, pp. 21–30.

[12] D. Radev, A. Winkel, and M. Topper, "Multi document centroid-based text summarization," in *ACL 2002*, 2002.

[13] J. Carbonell and J. Goldstein, "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*. ACM, 1998, pp. 335–336.

[14] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *J. Artif. Int. Res.*, vol. 22, no. 1, pp. 457–479, Dec. 2004.

[15] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." 1999.

[16] S. T. Dumais, "Latent semantic analysis," *Annual review of information science and technology*, vol. 38, no. 1, pp. 188–230, 2004.

[17] Y. Gong and X. Liu, "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*. ACM, 2001, pp. 19–25.

[18] J. Steinberger and K. Jezek, "Using latent semantic analysis in text summarization and summary evaluation," in *Proc. ISIM'04*, 2004, pp. 93–100.

[19] W. J. Wilbur and K. Sirotkin, "The automatic identification of stop words," *Journal of information science*, vol. 18, no. 1, pp. 45–55, 1992.

[20] S. Lafon and A. B. Lee, "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*, vol. 28, no. 9, pp. 1393–1403, 2006.

[21] S. Yang, "A scale-space theory for text," *CoRR*, vol. abs/1212.2145, 2012. [Online]. Available: http://arxiv.org/abs/1212.2145

[22] M. Slaney and D. Ponceleon, "Hierarchical segmentation using latent semantic indexing in scale space," in *icassp*, vol. 1, 2001, pp. 1437–1440.

[23] A. Witkin, "Scale-space filtering: A new approach to multi-scale description," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, vol. 9. IEEE, 1984, pp. 150–153.

[24] R. Lyon, "Speech recognition in scale space," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87.*, vol. 12. IEEE, 1987, pp. 1265–1268.

[25] J. J. Koenderink, "The structure of images," *Biological cybernetics*, vol. 50, no. 5, pp. 363–370, 1984.

[26] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[27] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.

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

[29] J. O. Ramsay, *Functional data analysis*. Wiley Online Library, 2006.

[30] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[31] P. J. Olver, "Introduction to the calculus of variations."

[32] M. A. Peletier, "Energies, gradient flows, and large deviations: a modelling," 2011.

[33] L. Saloff-Coste, "The heat kernel and its estimates."

[34] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, vol. 92.

[35] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[36] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[37] D. Bindel, "Lecture notes in matrix computations," November 2011.

[38] J. Burkardt. (2013) Laplacian the discrete laplacian operator. https://people.sc.fsu.edu/~jburkardt/f77_src/laplacian/laplacian.html. [Online]. Available: https://people.sc.fsu.edu/~jburkardt/f77_src/laplacian/laplacian.html

[39] A. I. Bobenko, J. M. Sullivan, P. Schröder, and G. M. Ziegler, *Discrete differential geometry*. Springer, 2008.

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

[41] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[42] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[43] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and computational harmonic analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[44] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[45] F. De Goes, S. Goldenstein, and L. Velho, "A hierarchical segmentation of articulated bodies," in *Computer graphics forum*, vol. 27, no. 5. Wiley Online Library, 2008, pp. 1349–1356.

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

[47] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National academy of Sciences*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.

[48] W. M. Darling, "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*, 2011, pp. 642–647.

[49] D. Q. Nguyen, "jLDADMM: A Java package for the LDA and DMM topic models," http://jldadmm.sourceforge.net/, 2015.

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

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

[52] A. A. Alemi and P. Ginsparg, "Text segmentation based on semantic word embeddings," *arXiv preprint arXiv:1503.05543*, 2015.

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

[54] D. Beeferman, A. Berger, and J. Lafferty, "Statistical models for text segmentation," *Machine learning*, vol. 34, no. 1, pp. 177–210, 1999.

[55] R. JeffreyPennington and C. Manning, "Glove: Global vectors for word representation."

[56] F. Chung, "The heat kernel as the pagerank of a graph," *Proceedings of the National Academy of Sciences*, vol. 104, no. 50, pp. 19 735–19 740, 2007.

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