Statistical Society of Canada Scheduling

Ryan McQueen
Mahdi Aziz
Norbert Eke

December 14, 2017

1 Introduction

Every year a conference is hosted by the Statistical Society of Canada for statisticians to present their research. Currently, each conference is manually scheduled by the organizers of the conference which consumes a significant amount of time and may result in schedules which are not as optimal as they could be. An automated system which extracts each presentation's information, determines similar presentations, constructs a schedule based on the similar presentations, and optimizes the schedule based on a list of specified constraints is required.

2 Method

The proposed method to solve the problem introduced is broken into three main components: data extraction, data analysis, and scheduling. A high level architecture which outlines the system can be seen within Figure 1.

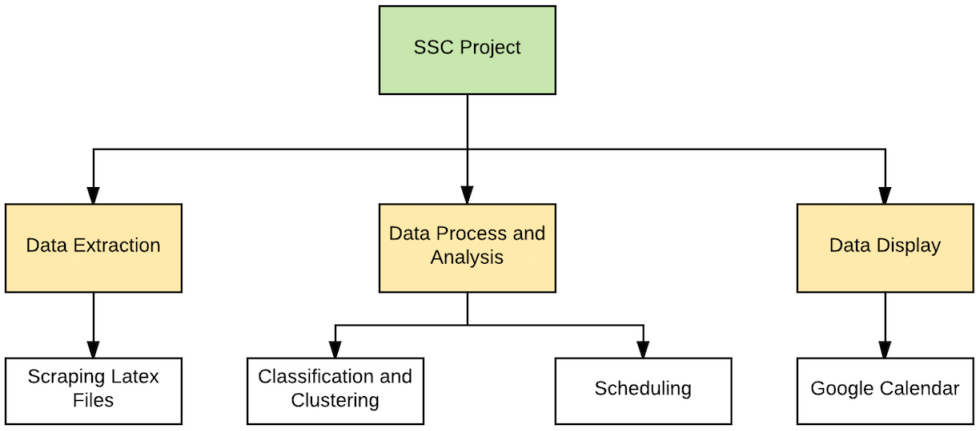

Figure 1: High-Level Architecture Diagram of the System

In addition to the high-level architecture, a box-system diagram is also illustrated within Figure 2. This diagram illustrates in more depth the interactions between each component and illustrates the items that make up each component. For example, the abstract CSV files from the Data Extraction component are passed to the Data Analysis component, and more specifically, the abstract text analysis process within this component. The final output, as shown in Figure 2, is a Google calendar file which allows for an easy visualization to the conference organizers.

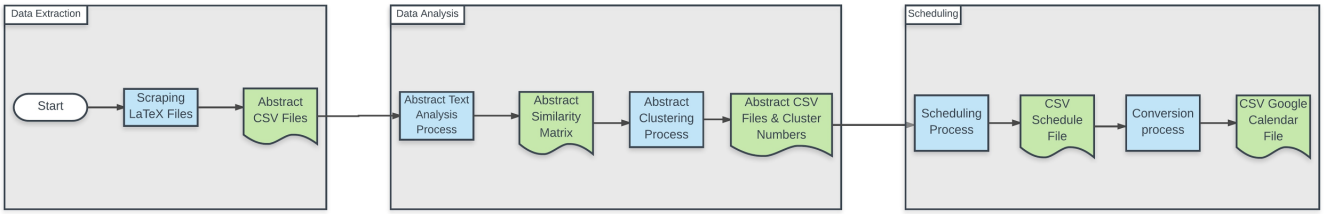

Figure 2: TO DO: better caption

2.1 Data Extraction

Provided to this project was a folder containing various LaTeX files which indicated useful information about the conference. The two LaTeX files of importance for the data extraction component are the following: *abstracts.tex* and *prog.tex* as they highlighted the details regarding each presenter.

In order to fully grasp the approaches conducted within this section, an overview of what each file contained is required. The file named *abstracts.tex* indicated the title of the abstract, the description of the talk, the time the talk took place during the last year's conference, and who the authors are. Within the *prog.tex* file, the abstract titles of each presenter and an associated flag which indicates whether the talk was invited, contributed, or a poster session, are present. From these files, there were distinct LaTeX commands which indicated each field, such as the abstract title being indicated by \absTitle. From this, it was apparent that Regular Expressions could be utilized in order to recognize these patterns present within the LaTeX files, and extract the associated information with them. As shown in Figure 3 and Figure 4 the LaTeX code to be parsed followed the same pattern for each presenter attending the conference.

```
\absTime{\Monday}{10:50-11:20}
{
\Author{Kelly M.}{Burkett}{University of Ottawa}
}
\abstitle{An Ancestral Tree-Based Approach to Detect Rare and Common Variants}{Approche arborescente ancestrale pour détecter les variants rares et communs}
\absSideBySide{For detecting genetic variants associated with a disease or trait, it is useful to consider the ancestral trees that gave rise to the sample's genetic variability. For both rare and common disease or trait influencing genetic variants, we expect to see haplotypes from individuals with similar values of the disease or trait clustered together in the ancestral tree corresponding to the genomic location of the variant. In this presentation, we describe how tree-based statistics can be used for detecting both rare and common genetic variants associated with either continuous or dichotomous outcomes. We summarize the performance of these statistics on simulated data having known and missing tree structures and we compare results to those obtained using conventional approaches to detect genetic association. Finally, application of the tree-based method to real data is also discussed.}{Afin de détecter les variants génétiques associés à une maladie ou à un caractère, il est utile de considérer les arbres ancestraux qui ont donné lieu à la variabilité génétique des échantillons. Pour les variants génétiques rares et communs qui influencent les maladies ou certaines caractéristiques, nous nous attendons à voir des haplotypes qui se regroupent chez des individus qui ont des valeurs de maladie ou de caractéristique semblables, dans un arbre ancestral correspondant à la localisation génomique du variant. Dans cet exposé, nous décrivons comment les statistiques arborescentes peuvent être utilisées pour détecter les variants génétiques rares et communs avec des variables dépendantes soit continues ou dichotomiques. Nous montrons l'efficacité de ces statistiques avec des données simulées qui ont des structures arborescentes connues et manquantes. Nous comparons les résultats à ceux qui ont été obtenus avec des approches conventionnelles afin de détecter une association génétique. Enfin, nous présentons également l'application de la méthode arborescente sur des données réelles.}
```

Figure 3: The Format of the abstracts.tex File

```
\grSciSession{10:20-11:50}{E3 270 (EITC)}{Analysis of Complex Traits in Families and Populations}{Analyse des caractères complexes dans les familles et populations}{\Invited}{Jinko Graham}{Jinko Graham}{Biostatistics Section / Groupe de biostatistique}{5425}

\grSchedTalk{10:20-10:50}
{
\Author{J. Concepcion}{Loredo-Osti}{Memorial University of Newfoundland}
}
{The Analysis of Longitudinal Multivariate (Discrete or Continuous) Traits under Irregular Time Measurements}{Analyse longitudinale de traits multivariés (discrets ou continus) avec des temps de mesure irréguliers}
{\bubbleE \enspace \screenE}
\grSchedTalk{10:50-11:20}
{
\Author{Kelly M.}{Burkett}{University of Ottawa}
}
{An Ancestral Tree-Based Approach to Detect Rare and Common Variants}{Approche arborescente ancestrale pour détecter les variants rares et communs}
{\bubbleE \enspace \screenE}
\grSchedTalk{11:20-11:50}
{
\Author{Fabrice}{Larribe}{Université du Québec à Montréal}
}
{Mapping Complex Traits, Rare Variants and Interaction via the Coalescent Process with Recombination }{Cartographie de traits complexes, de variants rares et d'interaction par le processus de coalescence avec recombinaison}
{\bubbleE \enspace \screenE}
```

Figure 4: The Format of the prog.tex File

Once each abstract title, abstract description, authors, and time had been extracted, a requirement of determining the associated flag with each abstract was required for the scheduling portion of the project. This associated flag would indicate whether an abstract was contributed, invited, or a poster session. These flags were indicated simply by \Contributed, \Invited, \Poster respectively, and Regular Expressions were again used to extract a block of abstracts belonging to each flag. An example of the formatting for the provided LaTeX files is shown within Figure 3 and Figure 4.

After each flag had been extracted, a mapping was needed in order to associate each flag with the relevant abstract. A simple, and effective approach to perform this action was to construct a hash map which would have the abstract title as the key, and the associated flag as the value. An example of this is presented within Figure 5

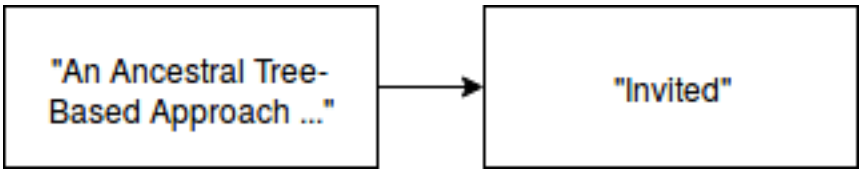

Figure 5: Hash Map of Abstract Title and the Associated Flag

From this mapping, each presenter from the *abstracts.tex* file now has an associated flag appended to the existing information previously extracted (as outlined within Figure 6). Due to the incorporation of the associated flag into the abstract data, the data is prepared to be exported to a CSV format with a UTF-8 encoding to allow for easier importing and manipulation in the following Method subsections.

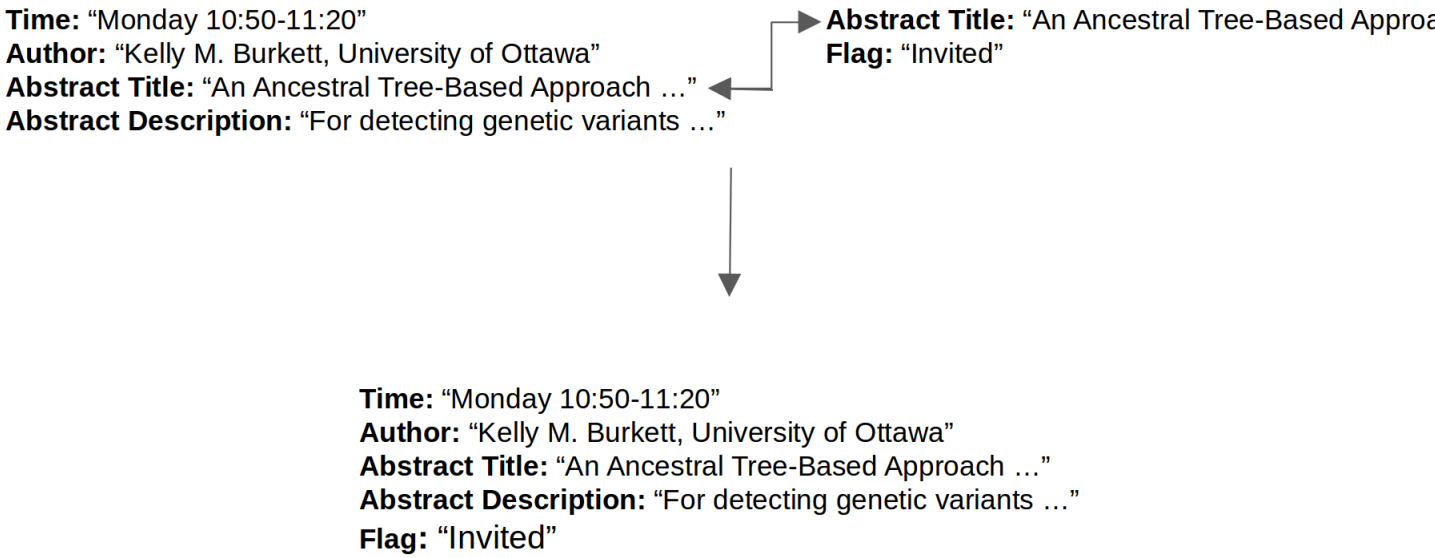

Figure 6: Mapped Associated Flag to the Abstracts Data

Due to the lack of the keynote, and student speaker labels within the original data set, a requirement of performing random labeling was required in order to satisfy the project requirements. As outlined by the client for this project, approximately thirty speakers out of the contributed presentations are student speakers, and there are three keynote speakers within the invited presentations. From this information, a random sampling on the presentations with the flag of "Contributed" and "Invited" is required in order to fulfill the project requirements. This random sampling re-labels thirty of the presentations with the flag of "Contributed" to be "Student" and three of the presentations with the flag of "Invited" to "Keynote". Due to this re-labeling, a subset of these presentations may truly be "Student", or "Keynote" presentations, while the remaining presentations are incorrect. Although this may seem like an inappropriate component of the project, it does not affect any of the results as it is known there are thirty student presentations and three key note speakers throughout the entire conference regardless.

An apparent issue that is present within the original LaTeX files is that the *abstracts.tex* file contains

entries which the *prog.tex* file does not. In order to have an associated flag with each abstract, it was deemed necessary to ignore any abstracts not present within *prog.tex*. Due to the scheduling component's requirement of the associated flag being present for each abstract in order to satisfy the list of given constraints, the removal of these conflicting abstract entries was deemed appropriate.

2.2 Statistical Analysis

The statistical analysis process, as shown in Figure 7, is composed of two main processes: *Abstract Text Analysis* and *Abstract Similarity Cluster Analysis*. Additionally the text analysis process has an extra step called *Abstract Text Processing*. The main idea behind the statistical analysis is to use self-trained and pre-trained deep learning language models to analyze and calculate the contextual, semantic similarity between abstracts. The language models generate an n x n similarity matrix quantifying how similar abstract i is to abstract j. Finally, the similarity matrix is undergoing cluster analysis in order to find groups within the data. The result of the statistical analysis is a table outlining the cluster memberships of each abstract.

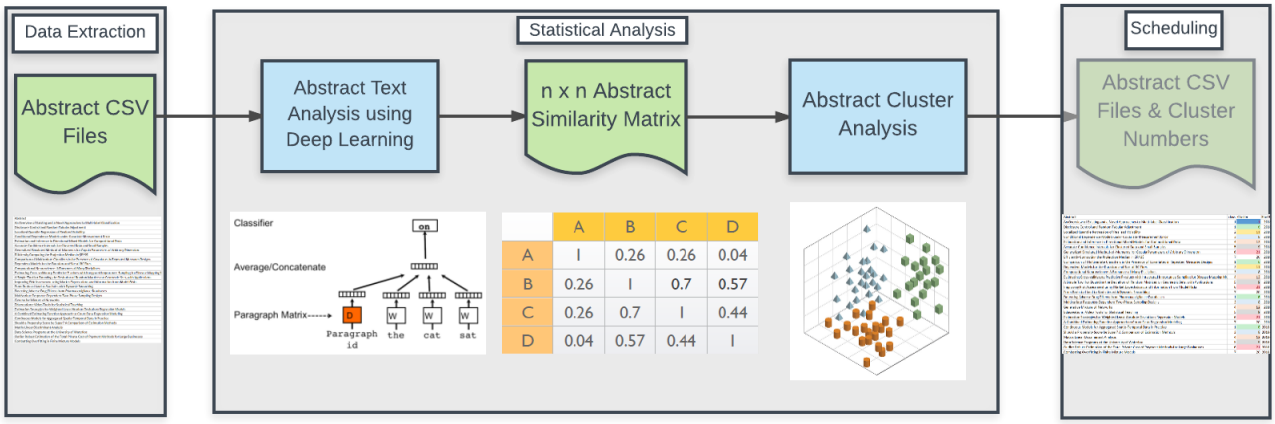

Figure 7: Detailed Overview of the Statistical Analysis Process

2.2.1 Abstract Text Processing

Before one is able to analyze the textual data (abstracts), the text needs to be processed according to what the language models expect as input. The deep learning language models take as input a list of cleaned up text documents. For text cleaning the deep learning models expect the removal of numerical values and punctuation, also word tokenization and stop-word removal is also expected. In text processing, word tokenization is referred to the separation of words within sentences into single tokens, also stop-word removal refers to the removal of words like "*the*", "*so*", "*such*", "*then*", "*an*",etc. Phrase detection techniques are recommended to be applied to the textual data, but in terms of the language model's input requirements this is optional.

2.2.2 Abstract Text Analysis using Deep Learning Models

The use of deep learning models is really common in the fields of Natural Language Processing and Understanding. Word and document representation language models like Word2Vec [9], [11] and Doc2Vec [8] have been really popular, since they have been introduced to the general public by researchers from Google. Word2Vec is a deep learning model that allows the computation of vector representations of words. This model takes a text corpus as input and produces high dimensional word vectors as output. In one of Google's blog posts the algorithm is described as "it constructs a vocabulary from the

training text data and then learns vector representation of words. The resulting word vector file can be used as features in many natural language processing and machine learning applications“ [12]. Essentially, with the help of the Word2Vec model, words get transformed into high dimensional vectors. The Word2Vec model is trained to reconstructs the linguistic context of words by placing the vectors of words used in the same context close to each other. The resulting word vectors are also useful, since they “capture the semantic similarity between words by placing the vectors of similar words closer to each other“[12]. Figure 8 offers a visual explanation of how Word2Vec works and how it is used.

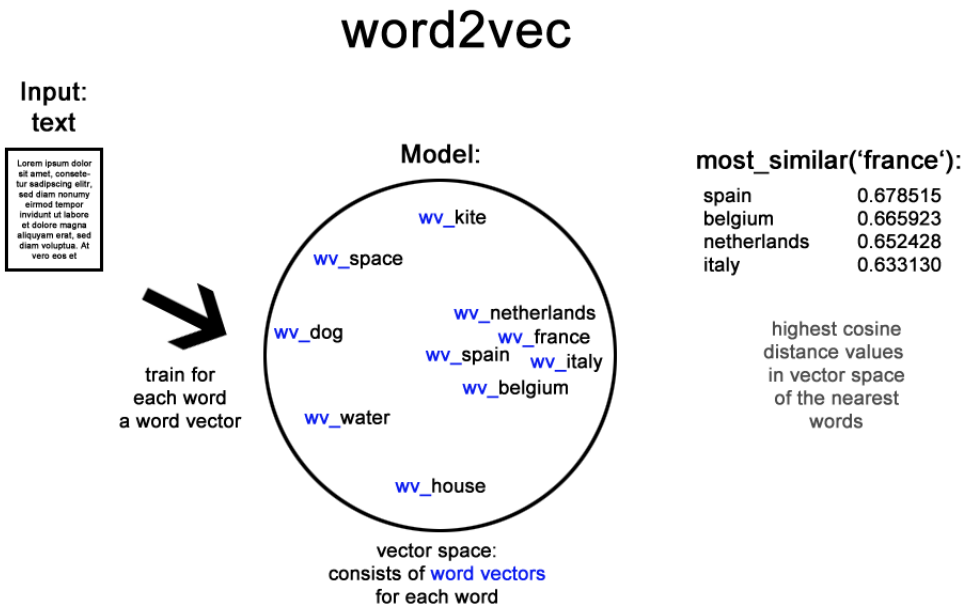

Figure 8: Detailed Overview of the Deep Learning Model Called Word2Vec. Image credits go to [6].

Doc2Vec is another deep learning model used in this project. Doc2Vec consists of the same principles as the previously introduced Word2Vec model, except that it created vector representations of documents (or paragraphs), instead of just words. Figure 9 offers a visual explanation of how Doc2Vec works and how it is used.

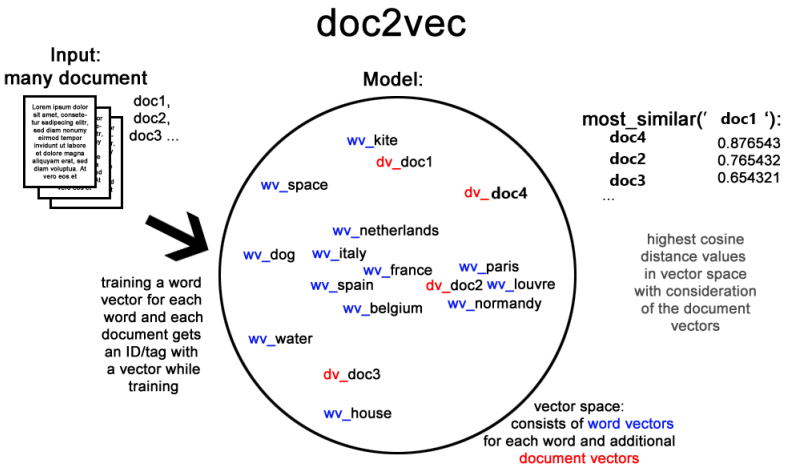

Figure 9: Detailed Overview of the Deep Learning Model Called Doc2Vec. Image credits go to [6].

List of Different Models to be Used in the Text Analysis Process:

1. **Self trained 300 dimensional Doc2Vec model:**
   All 283 talk abstracts have been used as training data for a self-trained 300 dimensional Doc2Vec model, which is just a paragraph (abstract document) vector representation of a document (abstract) from the training data. Since the training data sample is so small, this model is not expected to perform too well. Gensim [15], the python library's implementation of the Doc2Vec model needs various parameters. The parameters for this model are the following: $size = 300$, $min\_count = 2$, $window = 10$, $negative = 5$, $hs = 0$, $workers = 4$. The Doc2Vec model creates a vector representation of each abstract within the training data. Once vectors are obtained, an n x n similarity matrix is generated by calculating the cosine similarity value between each abstract i's vector and abstract j's vector.

2. **AP News pre-trained Doc2Vec model:**
   Same process as with the previous Doc2Vec model, except that this model is a pre-trained Doc2Vec model trained on AP News articles and it comes from outside sources [7]. Same process, the model creates a document vector representation of each abstract An n x n similarity matrix is generated by calculating the cosine similarity value between each abstract i's vector and abstract j's vector.

3. **Wikipedia pre-trained Doc2Vec model:**
   This model is a pre-trained Doc2Vec model trained on Wikipedia articles and it comes from outside sources [7]. Same process as before, the model creates a document vector representation of each abstract An n x n similarity matrix is generated by calculating the cosine similarity value between each abstract i's vector and abstract j's vector.

4. **InferSent using a pre-trained GloVe Model:**
   InferSent is a sentence embedding encoder that provides semantic sentence representation. This technique allows plenty of flexibility towards the interpretation of abstracts by learning the semantics of sentences. Since all abstract documents go though a text cleaning phase, for the purposes of this project each abstract is treated as a sentence. This makes sense, since after text processing, the whole abstract seems like a long sentence, having all punctuation stripped away. Alternative to this process could be to have the InferSent encoder process through each sentence within an abstract, then once each sentence is encoded into vectors, apply some vector aggregation technique to generate document based vectors from sentence based vectors. Once vectors are obtained, generate an n x n similarity matrix by calculating the cosine similarity value between each abstract i's vector and abstract j's vector. InferSent [1] and GloVe [14], the inner word vector space model are publicly available and usable.

5. **DocSim using self-trained Latent Semantic Indexing model:**
   The DocSim model consists of an inner Latent Semantic Indexing (LSI) model created from the 283 abstracts within the dataset. This LSI model is a topic modelling approach's result, which can be built from a simple dictionary and text corpus of the dataset, which is created inside this DocSim model using the cleaned up version of the abstract documents. The LSI model has its own special LSI vector space, thus each abtract is represented by a vector is the LSI vector space. With the help of the DocSim model's similarity queries for each abstract i in the dataset, an n x n similarity matrix is returned.

6. **Word Mover's Distance using GoogleNews300 model:**
   Word Mover's Distance (WMD) [5] is a technique that takes word vectors and calculates distances between documents. This technique creates an n x n distance matrix between all abstracts, which will be used to perform clustering analysis on. The WMD technique needs an inner word vector representation model and the WMD paper [5] suggests to use Google's GoogleNews 300 dimensional vector model [10], thus this pre-trained Word2Vec model is used as an inner model for WMD's similarity calculations.

2.2.3 Abstract Similarity Cluster Analysis

Once the Text Analysis Process finished its job, and generated the n x n similarity matrix, clustering algorithms can take over and find groups within the data, then output the cluster membership of each abstract document. The clustering process needs to investigate what clustering algorithm(s) suit the data better, and evaluate what is the best approach.

2.3 Scheduling Problem

In this section, we will explain how to schedule the talks for SSC conference based on the cluster numbers obtained from the cluster analysis. The scheduling problem can be viewed as an optimization problem $f(h, s)$, which includes two categories of constraints: hard constraints and soft constraints. The hard constraints refer to limitations that the schedule must follow them in order to be feasible. Soft constraints refer to constraints that are optional for the schedule to follow them. But, if it does, the schedule would be a better schedule in terms of efficiency or client's requirements.

2.3.1 Scheduling Problem- SSC special Specification

As mentioned, the SSC conference will be held in three consecutive days (usually Mon. to Wed.). Each day except the last day is divided into three sessions (the last day only has two sessions). Each session can have up to n periods where n is dependent on a number of talks in that session and the period length defined by the user. For the upcoming conference, we have 286 talks need to be scheduled. In these talks, we have four different groups (flags): keynote, student, invited and contributed talks, which here they are sorted in order. It means that with two speakers in the same cluster and same class and session, the one with the higher flag should go first. For each talk, we have priority number, which comes into play when there are two talks with the same flag in the same room and session.

2.3.2 Notation

Before jumping into the scheduling problem and how to solve it let us start with some notations.

Let R is the set of rooms, T the set of talks, C the set of clusters, P the set of periods, S the set sessions, D the set of days, n_r the number of rooms, n_d number of days, n_{sd} the number of sessions in day d, n_{psd} number of periods in session s in day d, n_c number of clusters and finally n_{tc} the number of talks in cluster c. In addition, $\bar{A}$ is a vector, showing the unavailability of speakers and X is the schedule, a six-dimensional binary matrix, defined as:

$X = \text{new boolean}[n_r, n_{ts}, n_c, n_{psd}, n_{sd}, n_d]$

2.3.3 Hard constraints

The set of hard constraints for our scheduling problem can be defined as:

- **H1** Talks with the same cluster should not happen at the same time.

  $\sum_{r\in R}\sum_{t\in T} X_{rtcspd} = 1, \quad \forall c \in C, s \in S, p \in P, d \in D$

- **H2** A talk must only appear once in the whole conference.

  $\sum_{r\in R}\sum_{c\in C}\sum_{p\in P}\sum_{s\in S}\sum_{d\in D} X_{rtcspd} = 1, \quad \forall t \in T$

- **H3** A cluster c is to be assigned T_c talks in the whole conference.

  $\sum_{r\in R}\sum_{t\in T}\sum_{p\in P}\sum_{s\in S}\sum_{d\in D} X_{rtcpsd} = K_c \quad , \forall c \in C$

- **H4** A talk cannot be assigned to more than one period.

  $\sum_{r\in R}\sum_{c\in C}\sum_{s\in S}\sum_{p\in P}\sum_{d\in D} X_{rtcpsd} = 1, \quad \forall t \in T$

- **H5** A room cannot be assigned to more than one talk at each timeslot.

  $\sum_{t\in T}\sum_{c\in C}\sum_{s\in S}\sum_{p\in P}\sum_{d\in D} X_{rtcpsd} = 1, \quad \forall r \in R$

- **H6** The talk is scheduled provided that the speaker is available.

  $\sum_{r\in R} X_{rtcpsd} = 0, \quad if (p,s,d) \notin A_t \quad \forall t \in T$

- **H7:** The keynote talks should happen only once in each day of conference.

The hard constraints mentioned above guarantee that the schedule is feasible as long as it follows all of them.

$\sum_{i=1}^{7} H_i = 0$

2.4 Soft Constraints

We also have a set of soft constraints.

- **S1:** The bigger size of a cluster, the bigger size of a room should be assigned.

  $S_1 = \sum_{r\in R}\sum_{c\in C}[(S_c - C_r)(\sum_{t\in T}\sum_{p\in P}\sum_{s\in S}\sum_{d\in D} X_{rtcpsd})]$

  where S_c is the cluster size for Cluster c and C_r is the capacity of room r.

- **S2:** This penalty counts the number of times that talks with the same cluster and the same session and the same day happens in different rooms.

  $S2 = \sum_{c\in C}\sum_{s\in S}\sum_{d\in D}\delta_{csd}$

  where $\delta_{csd} = 1$ if $\sum_{r\in R}\sum_{t\in T}\sum_{p\in P} X_{rtcpsd} > 1 \quad \forall c \in C, s \in S, d \in D$.

- **S3:** Talks with higher priorities should be taken place first in the schedule.

- **S4:** Keynote talks are preferred to happen in the first period of each day.

- **S5:** Keynote talks are preferred to be in the largest rooms.

- **S5:** No talk should be parallel with keynote talks. This means that this penalty counts the number of talks that collide with the keynote talks.

- **S6:** In each session, between two speakers with the same flag, the one with higher priority should go first.

These soft constraints are used an evaluation tool of our feasible schedule. The objective function of our optimization problem is defined as:

$f = \sum_{i=1}^{6} \theta_i S_i$

where i is the penalty for criterion i and θ_i is the importance of its respective importance.

2.5 Greedy algorithm for solving the scheduling problem

Since our problem is modeled as an optimization problem, we can solve it by a wide range of optimization techniques. Algorithm 10 presents the pseudo-code for the greedy algorithm for solving the scheduling problem.

Algorithm 1 A Greedy algorithm for solving the scheduling problem

```
Greedy (R,C,D,S,P,l)
    Sort rooms in R based on their capacities
    Sort clusters in C based on their sizes
    Assign keynote speakers to the largest room and the first period of the first session of each day
    Choose a cluster c in C with the largest size
    For each r in R
        For each d in D
            For each s in S
                Set s_l as session length for s
                Set n_p = s_l/l
                For p = 1 to n_p
                    If not the first p in the first s in d
                        Choose a talk t in c with highest flag and priority, which has not been scheduled
                        Assign t to our schedule X in day d, session s, period p.
                        Choose a new cluster if all talks in C are scheduled and go to the next session
    Return X
```

The description of Algorithm 1 is as follows:

The input parameters for the Greedy algorithm are the set of rooms R, the set of clusters C, the set of days of conference D, the set of sessions S, the set of periods P and l the period length, which all can be defined by the user.

In **Line 1 and 2**, we sort rooms and clusters based on their sizes since we want to make sure that cluster with bigger size would get a room with a higher capacity.

In **Line 3**, we set the keynote speakers to the largest room and at the first session and the first period of each day. In this way, we make sure that key note speakers only appear once in the whole conference ($H1$) and they got the largest room ($S4$) and they happen in the first period ($S5$).

In **Line 10**, we make sure that no speaker would be parallel with keynote speaker ($S6$).

In **Line 11**, we chose a talk in the specified cluster c with higher flag and higher priority, which has not been assigned. In this way, a talk only happens once in the conference ($H2$). Moreover, the condition that a talk with higher flag and higher priority make sure that soft constraints $S3$ and $S7$ would be followed. Figure 10 shows what constraint exactly is targeted by what line of code.

3 Results

3.1 Data Extraction

From the abstracts.tex file, there are three hundred and eight abstracts present; however, not all of them are utilized due to the issue with the data set mentioned within Method section of this report. Due to the data set issue, from the original three hundred and eight abstracts, two hundred and eighty seven are extracted.

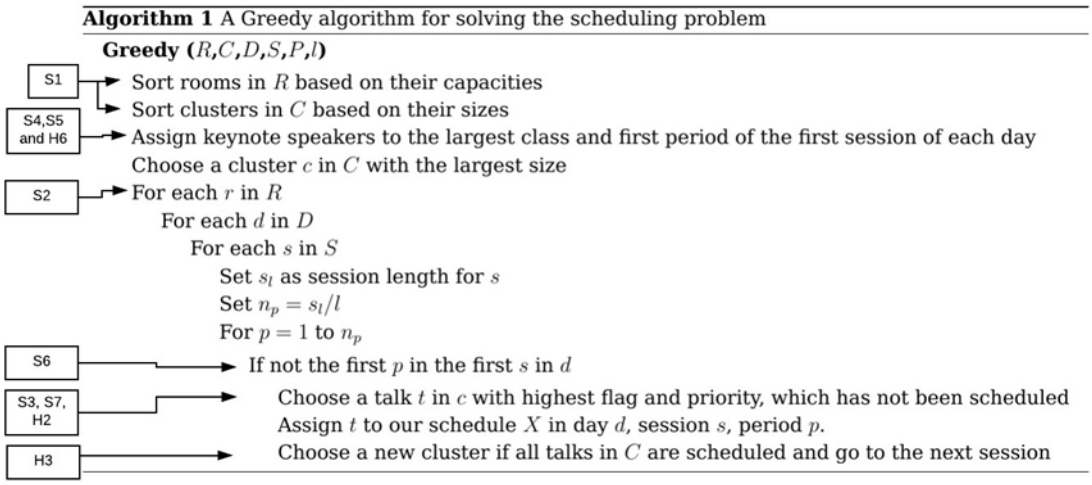

Figure 10: The greedy algorithm- targeting constraints

3.2 Data Analysis Process

3.2.1 Abstract Text Analysis

The pre-trained deep learning language models are evaluated using Pearson correlation coefficient and Spearman rank-order correlation coefficient between the similarities from the benchmark dataset and the similarities produced by the model itself. The following results are reported as a double measurement (pearson, spearman).

Model evaluations for Doc2Vec wikipedia:

1. On wordsim353.tsv benchmark dataset:
   Pearson correlation coefficient: 0.60742269433967655 with p-value 5.6638188016353266e-37
   Spearman rank-order correlation:0.60930652035520737 with p-value=2.9814873059166932e-37

2. On simlex999.txt benchmark dataset:
   Pearson correlation coefficient: 0.2972443165530957 with p-value 8.2913908167679857e-22
   Spearman rank-order correlation: 0.30044870012476527 with p-value=2.87886723010645e-22

Model evaluations for Doc2Vec apnews:

1. On wordsim353.tsv benchmark dataset:
   Pearson correlation coefficient: 0.51852547652963821 with p-value 1.5112639055393685e-25
   Spearman rank-order correlation: 0.51256622927130413 with p-value 6.5841239982882924e-25

2. On simlex999.txt benchmark dataset:
   Pearson correlation coefficient: 0.26378202357510744 with p-value 2.3864830525042184e-17
   Spearman rank-order correlation: 0.26566192766881647 with p-value 1.3908232313242245e-17

Model evaluations for Word2Vec GoogleNews inside the WMD model:

1. On wordsim353.tsv benchmark dataset:
   Pearson correlation coefficient: 0.62387734513046811 with p-value 1.7963247628700625e-39
   Spearman rank-order correlation: 0.65892158880092877 with p-value 2.5346056459149263e-45

2. On simlex999.txt benchmark dataset:
   Pearson correlation coefficient: 0.4471705601367435 with p-value 3.1602627605266494e-50
   Spearman rank-order correlation: 0.43607859778335434 with p-value 1.3896416353352258e-47

As an interpretation of these model evaluations, all 3 models mentioned above have a strong Pearson correlation and Spearman rank-order correlation of above 0.5 for the wordsim353 benchmark dataset with p values being almost 0, but most importantly definitely less than 0.05. The same can not be said for the model evaluations with simlex999 benchmark dataset, since the Pearson correlations and Spearman rank-order correlations are much lower than for the previous dataset. The wordsim353 benchmark dataset contains semantic similarity measures for concepts in the english language and it is a well respect and used benchmark [2]. On the other hand, Dr. Felix Hill, currently a Research Scientist at Deepmind and creator of the SimLex-999 benchmark dataset [4] says on his website that "SimLex-999 provides a way of measuring how well models capture similarity, rather than relatedness or association. The scores in SimLex-999 therefore differ from other well-known evaluation data-sets such as WordSim-353 (Finkelstein et al. 2002)" [3]. Based on the above background information on the two golder standard benchmark data-sets for word vector representation semantic similarity measurements, one can see that the two data-sets are measuring different things, thus it is totally normal to have different values for the Pearson and Spearman correlation coefficients. Based on the really low p values for the Pearson correlation coefficients one can conclude that there is a linear correlation between the model's similarity measurements and the benchmark dataset's measurements, making the models valid models. Spearman's correlation measures monotonic relationships between the model's similarity measurements and the benchmark dataset's measurements. Since Spearman's correlation values are between -1 and 1, values between 0.3 and 0.6 obtained in the model assessments show monotonic linear relationship between the model's similarity measurements and the benchmark dataset's measurements, thus making the models valid models, properly measuring semantic similarities within the data.

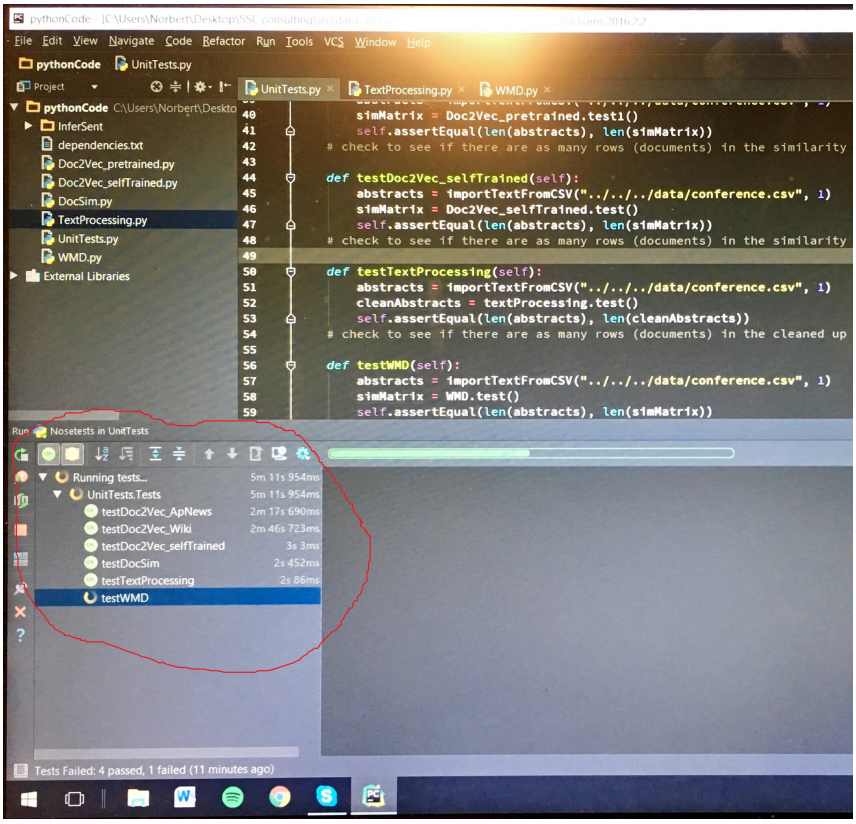

Figure 11: Screenshot of unit tests ran against the text analysis source files

Some unit testing was done for the text processing algorithm and for each model's source code. The following screenshot in figure 11. As the screenshots shows, all tests pass, except the last one, which

on an 8 GB ram laptop can not run. The reason being that this unit tests belongs to the WMD model's source code, which contains the loading in of a 7 GB model, which is not happening on any computer with only 8 GB of memory. Given a computer with larger memory, that unit would pass, and the model get loaded in.

3.2.2 Abstract Cluster Analysis

The abstract cluster analysis process takes in an abstract similarity matrix produced by the abstract text analysis process and outputs a table containing the cluster memberships for each abstract in the data-set. The first task within this cluster analysis process is how to determine the proper number of clusters within the data, also what is the optimal range of cluster sizes for this data-set. One small aside, worth noting is that most clustering algorithms take a distance matrix, or a dissimilarity matrix as input, thus the previously mentioned similarity matrix can easy be turned into a dissimilarity matrix, by knowing that a dissimilarity matrix equals to the similarity matrix subtracted from 1 ($DissimilarityMatrix = 1 - SimilarityMatrix$).

Figure 12 shows a sample cluster analysis on a dissimilarity matrix from Doc2Vec APNews model, and it shows how many groups different ways one could decide on the optimal number of groups within the data. Mixture model based cluster analysis, hierarchical cluster analysis and k-means cluster analysis algorithms were run on the sample similarity matrix.

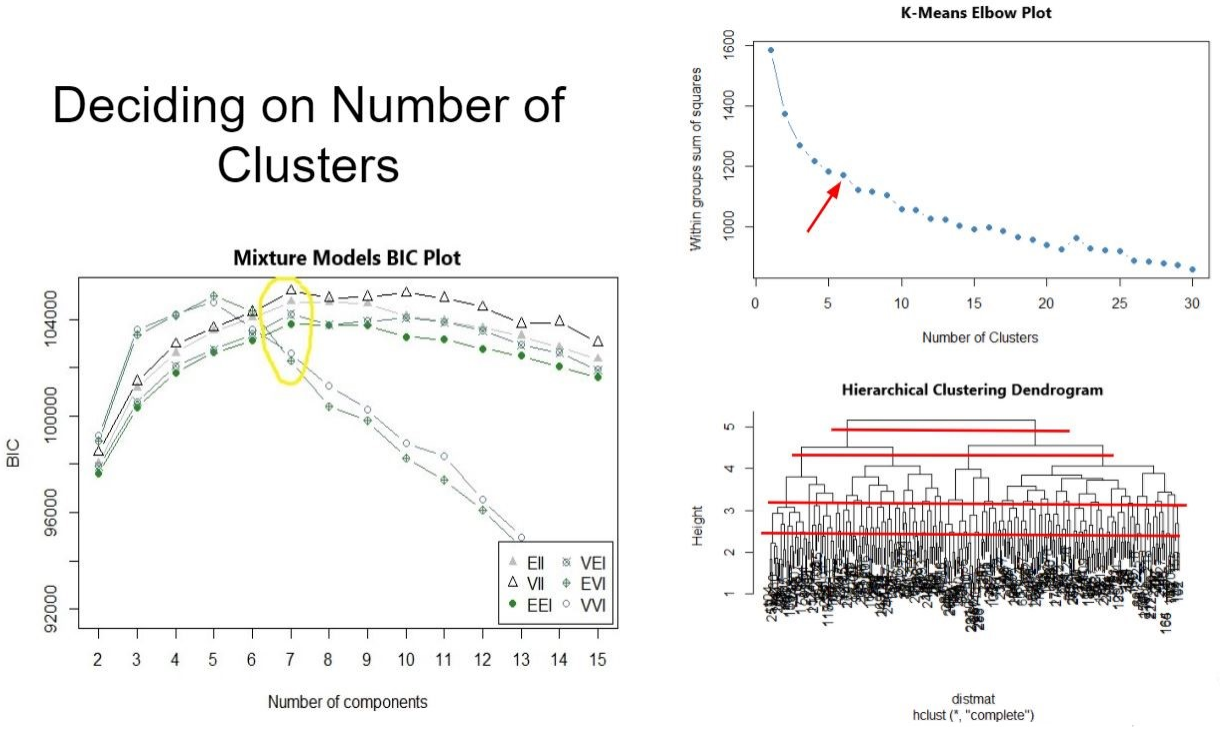

Figure 12: Various plots showing how many groups are within the data

On figure 12 the BIC plot of the mixture model clustering is choosing 7 as the optimal number of clusters within the data, as most types of mixture models have their BIC values maximized at 7 clusters. The K-means elbow plot is suggesting about 6 clusters, but one can look at the sudden increases in within cluster sum of squares and say that such plot can not be fully trusted, as the within clusters sum of squares fluctuates too many times as the number of clusters increase. As far as the hierarchical clustering goes, the dendrogram shown in Figure 12 has at least 4 reasonable places where the dendrogram could be cut, which massively impacts the cluster numbers and cluster sizes within the data.

The majority of the red lines shown in the image would make fairly unbalanced cluster sizes, as some cluster's size would be over 50, while there would also be a few clusters of size 1-5.

Based on figure 12 one would conclude that the BIC plot shows enough evidence towards deciding on a reasonable and optimal amount of clusters, while there is some truth in the dendrogram as well, since there might be many hierarchical relationships within the clusters. It can be concluded that one needs to pay a special attention to sub-clusters within the data. Perhaps some visualization from figure 13 might help decide how to pursue with the analysis.

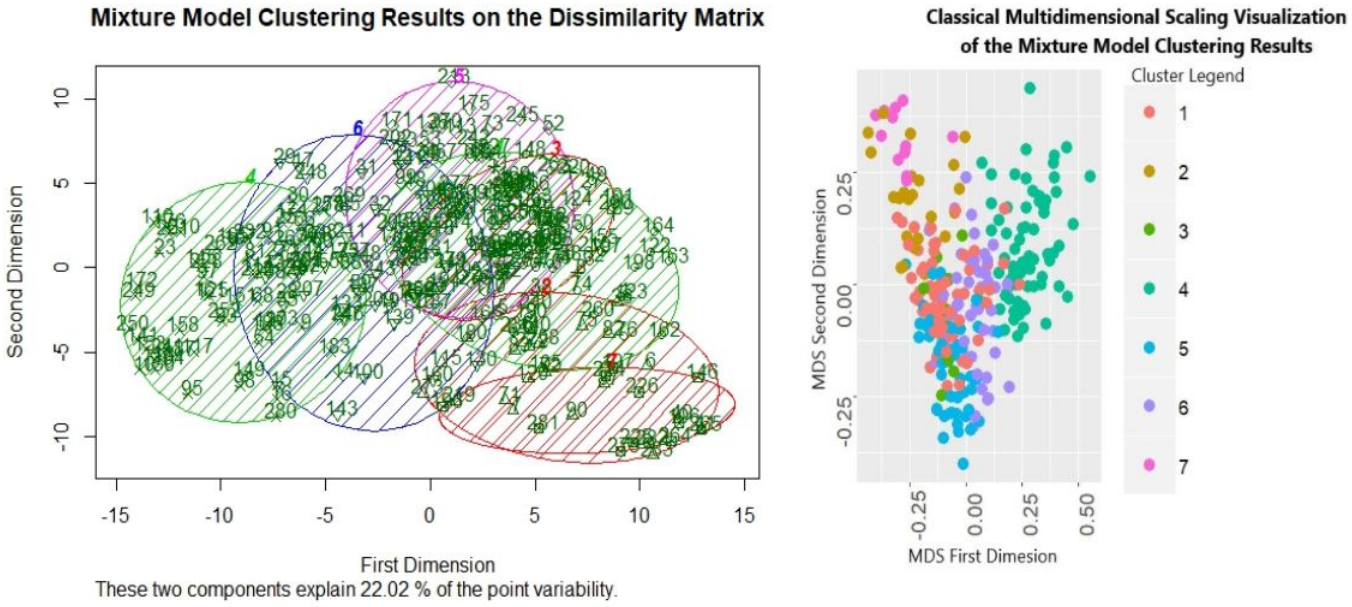

Figure 13: Visualization of Mixture Model Cluster Analysis Results using the First 2 Dimensions of the Dissimilarity Matrix and Multidimensional Scaling Applied to the Dissimilarity Matrix

Upon looking at figure 13, one can see the first two dimensions of the dissimilarity matrix plotted against each other, alongside with the 7 clusters given by the mixture model clustering algorithm. One can see that most clusters overlap in 2 dimensions, but one could imagine how in their original n x n space, those clusters could be real groups separated by some distance. These 7 groups are good, but there could be a way to narrow down the clusters to more specific clusters within each main cluster from the mixture model results, thus one needs to explore the sub-clusters within the data. In order to do so, a divisive clustering algorithm was applied to the whole dissimilarity matrix and the results are shown in figure 14.

All the sub-clusters circled with yellow in figure 14 would be nice to have as actual clusters within the data, so there needs to be a better way to cluster the data, in order to incorporate sub-clusters in the final results. There was a debate between which clustering algorithm to use, but after some discussion with the client and professor of the course, a consensus was reached: the best clustering approach for this data-set is to have mixture models obtain the main clusters within the data, then perform divisive clustering on each main cluster from the mixture model cluster memberships. The divisive clustering algorithm applied to the main clusters ends up identifying sub-clusters within the data, thus finding sub-topics within the conference abstracts. Figure 15 visually explains this best clustering approach.

Once sub-clusters are found with divisive clustering, all cluster memberships could be joined into one table, representing the overall cluster memberships within the data. clustering results.

In order to validate results, 5 fold cross validation was performed on the mixture model. The results of the cross validation can be seen in table 1. The results of the cross validation show that even with some k-fold subset of the sample data, the cluster size stay stable enough, at +- 1.5 clusters for each

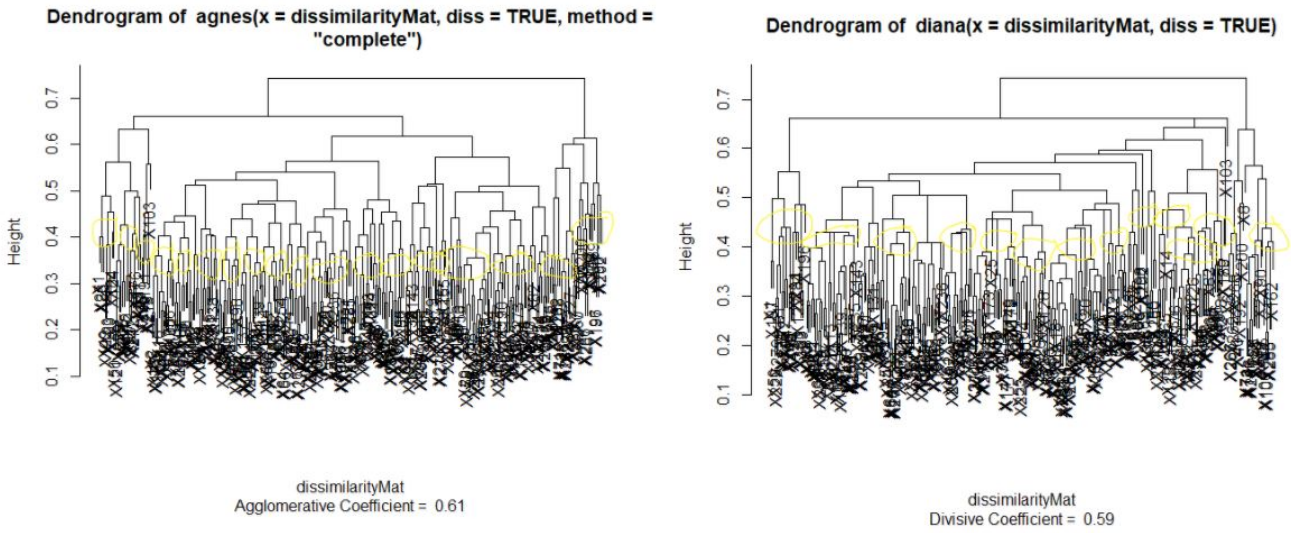

Figure 14: Exploring Hierarchical Relationship in the Sub-clusters of the Data

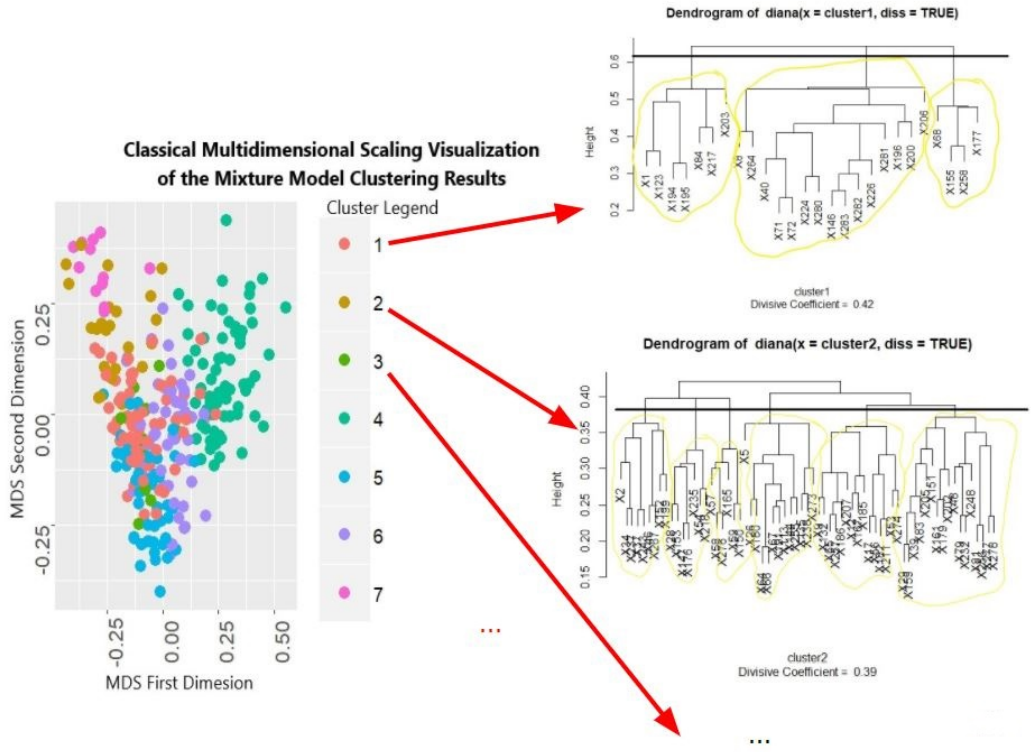

Figure 15: Best Clustering Approach: Joining Mixture Model Clustering and Divisive Clustering

average number of clusters from the cross validation results in table 1. The only outlier is the clusters coming from the WMD model's similarity matrix, but most likely this has been fixed by the subclusters found using the divisive clustering algorithm.

Overall Results

Sim Matrix from Model	Actual # of Clusters	Average # of Cluster from CV
InferSent with GloVe Model	13	11.5
Doc2Vec self-trained 300 dimensional model	12	10.8
Doc2Vec pre-trained apNews model	7	7.3
Doc2Vec pre-trained wiki Model	10	11.2
Word's Mover Distance using pre-trained Word2Vec	12	9
Document Similarity using a self-trained LSI model	7	7.8

Table 1: Cluster Stability Checking using 5 fold Cross Validation Results on the Mixture Model Clustering Results

6 different language models were used to get 6 different abstract similarity matrices: Doc2Vec_300dim, Doc2Vec_apnews, Doc2Vec_Wiki, InferSent_GloVe, DocSim_LSI and WMD_GoogleNews300.

The best clustering approach was applied to each language model, which resulted in 6 different cluster analyses. These 6 different analyses resulted in 6 potential schedules There are 6 potential schedules, but which one is the best ? Depends on the evaluation criteria. There needs to be some evaluation criteria that decides which is the best schedule. Clearly this evaluation criteria should be in the form of human input towards what sort of features within the schedule are more preferred/prioritized than other features within the schedule.

3.3 Scheduling Process

In this section, we first concentrate on the requirements for the final product and, then we will focus on the final product.

3.3.1 Client's Requirements

Our clients, (Dr. Jason Leoppky and Dr. Jeff Andrew) have told us that in order for our schedule to be acceptable, it must have the following criteria.

- **Code must be written in R:** Since the conference organizer and the conference is about statistics, and most statisticians know how to use R, the code should be written in R.
- **Scheduling the student talks:** The conference has some students talks that need to be scheduled properly. There are other types of speakers, but the most important part that has not been scheduled are student talks.
- **Satisfying the hard constraints:** There are some hard constraints mentioned by our clients that need to be followed. So, for an algorithm, in order to be an acceptable algorithm, it must find a schedule, which can find the schedule following all the hard constraints.
- **Exporting the schedule as a CSV file:** Our clients asked us to have the results as a CSV file. The CSV file should contain information about talks and also the timetable for student talks.

There is also a wish list that our final product follows:

- **Scheduling invited, contributed, keynote speakers:** In addition to student talks, we also have different types of speakers that need to be scheduled.
- **Scheduling panel discussion:** There is a panel discussion held for a conference that needs to be scheduled.
- **Scheduling poster sessions:** Some contributions are represented as poster sessions that need to be scheduled.

- **Exporting as a CSV Google calendar:** It would be an asset if our final product can be imported into Google calendar.

3.4 Results of Greedy Algorithm for Solving the Scheduling Problem

As we mentioned in the previous section, the greedy algorithm is used to solve the scheduling problem. We tested the greedy algorithm on the whole test set, which contains 6 tests (6 different analyses). Table 2 shows the results of the greedy algorithm.

Algorithm	Computation Time (s)	f
Type 1	6.5	0
Type 2	7.2	0
Type 3	7.6	0
Type 4	6.5	0
Type 5	7.5	0
Type 6	7.3	0

Table 2: The results of greedy algorithm for solving the scheduling problem where f was defined as the summation of all soft-constraints

As shown in Table 2, the results of the greedy algorithm is surprisingly good, which is intuitive. Since we have enough rooms, and there is no very hard constraints that make the greedy algorithm reconsider its choices, it will give us the best schedule.

There are other famous algorithms, geared for this problem such as heuristic algorithms. We did not utilize them as the computation time for function value computation is 20 s, which is more than the time required to run the greedy algorithm. So, it does not make sense to use any other algorithm that requires to compute the function call and iteratively solve the problem such as the heuristic algorithm.

3.4.1 Scheduling visualization

Here, in Figure 16 we provide a part of a schedule, generated by the Greedy algorithm for clustering algorithm type 1.

Monday											
	Start time	Cluster number	Flag	Start time	Cluster number	Flag	Priority	Start time	Cluster number	Flag	Priority
Room	2018-03-18 8:40			2018-03-18 9:01				2018-03-18 9:22			
Room 1	An Overview of Existing a	2	Keynote	Disclosure Control and Random Ta	6	Student	86	A Simple Tool for Boundi	6	Student	164
Room 2				Localized Quantile Regression of R	11	Contributed	233	Comparison of Multivaria	11	Contributed	281
Room 3				Conditional Dependence Models u	8	Contributed	179	Computational Neuroscie	8	Contributed	189
Room 4				Estimation and Inference in Directi	12	Student	278	Dependent Models for th	12	Invited	12
Room 5				Accurate Confidence Intervals for	9	Student	100	Estimating Cross-validatc	9	Student	157
Room 6				Generalized Simulated Method-of-	21	Invited	112	From Brain to Hand to St	21	Invited	193
Room 7				Efficiently Computing the Projectic	20	Student	148	Improving Risk Assessme	20	Invited	11

Room Name	Room number	Room capacity
AMT 100	1	100
AMT 101	2	75
AMT 102	3	75
ASC 165	4	50
ASC 163	5	50
SCI 3333	6	50
SCI 124	7	30
EME 100	8	30

Figure 16: A part of a schedule for clustering analysis 1 where the Greedy algorithm is used.

As shown in Figure 16, the schedule follows the constraints. For example, in the first period of the first session, the keynote speaker is scheduled in the largest room. Moreover, as you can see in the second period and the third period, all talks in the same room, belong to one cluster. Also, you can see that between student and contributed speaker, the student speaker was scheduled first.

3.4.2 Google CSV calendar- Final Product

In order to make our schedule easier to use, we convert it to Google CSV calendar that can be easily imported into one's Google Calendar. Figure 17 shows my calendar after importing the schedule. It should be noted that abstract of each talk is added to description part of each time slot.

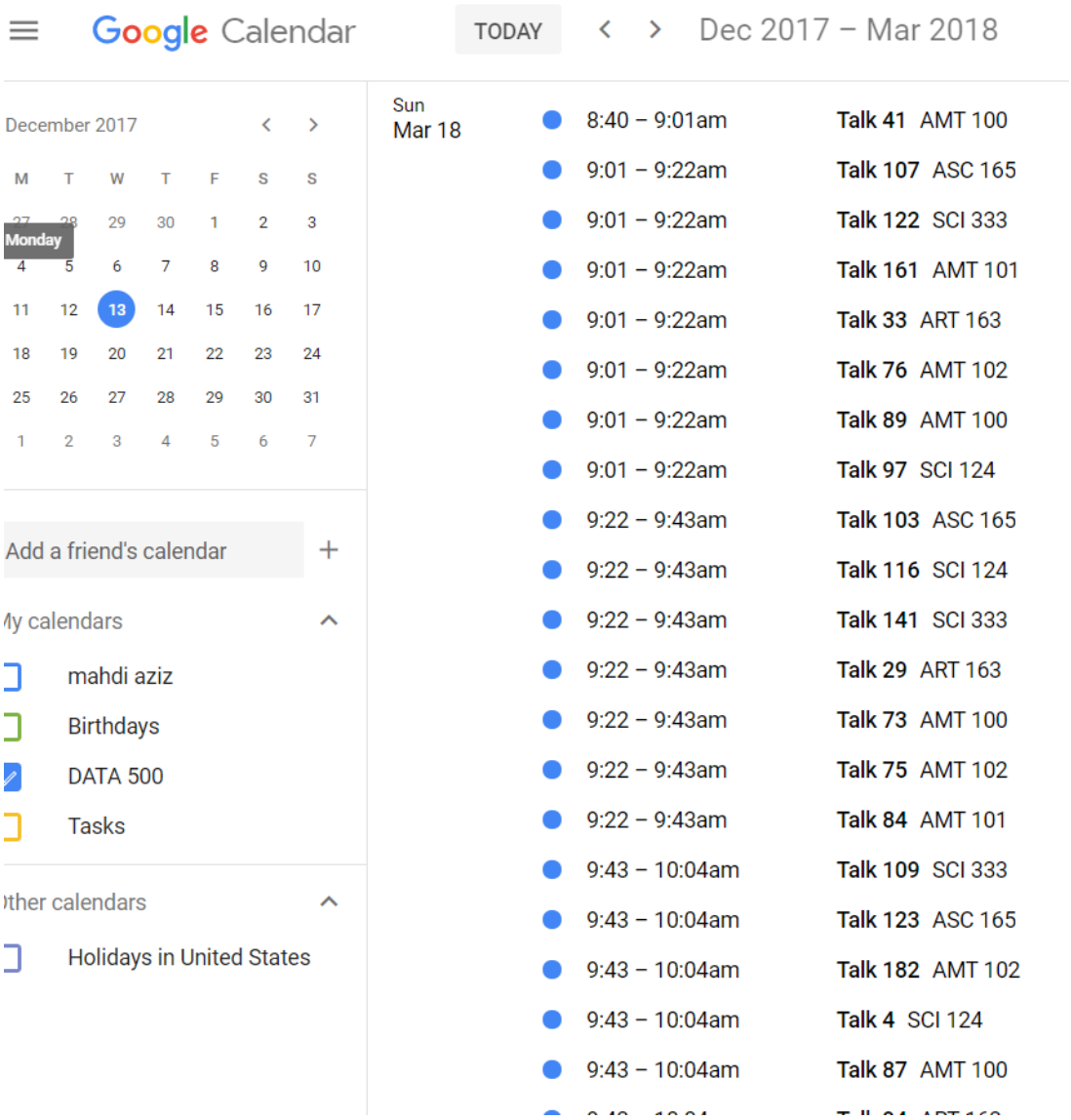

Figure 17: A part of my Google CSV calendar for clustering analysis 1 where the Greedy algorithm is used.

4 Conclusion

In this project, we constructed an automated system, which extracts the abstracts of the talks using some efficient scraping methods. Then, it will classify the abstracts based on their similarities using different clustering methods. Finally, using the clustering information provided, it will create a schedule that is feasible and promising. The final result is then presented in a CSV file that can be imported

into Google calendar easily. With regard to license, we chose MIT license as it is completely free, open source, and we are eager to see people to improve our project.

5 Future Work

From the data analysis point of view, one needs to create better deep learning language models, that better 'understand' the conceptual similarities in terms of statistical language semantics. It would be really useful, if a few dozen statistics textbooks would be fed into a Doc2Vec model, as training data, then use this new model trained on purely 'statistical domain knowledge' to compare similarities of abstracts. This would guarantee better results, since the contextual statistical semantics of a deep learning model trained purely on statistics would outperform a general model trained on news articles, or Wikipedia pages, since it would understand statistical concepts.

With regard to the scheduling perspective, we can define new algorithm that can provide decent results such as Quantum algorithm [13, 16] in case of having more complexities in our soft constraints.

6 Appendix

- Most of the deep learning text analysis would not be possible without the existence of the Python library called Gensim [15], so special thanks to the creators of the library.
- Some of the project files, alongside with the source files of this project can be found at this GitHub repository (https://github.com/norberte/Statistical-Consulting) .

References

[1] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[2] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.

[3] F. Hill. Computer laboratory, 2017.

[4] F. Hill, R. Reichart, and A. Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR*, abs/1408.3456, 2014.

[5] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *ICML*, 2015.

[6] K. L. Graphic representations of word2vec and doc2vec, Jun 2015.

[7] J. H. Lau and T. Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

[8] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

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

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Googlenews 300 dimensional word vectors. Statistical Model 12, Google, December 2013. Retrieved October 6, 2017.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[12] T. Mikolov, I. Sutskever, and Q. Le. Learning the meaning behind words. Google open source blog, Google Knowledge, 2013. Retrieved October 6, 2017.

[13] M. A. Nasser Yousefi and Bahram Yousefi M. H. Tayarani- N. A Novel Operator for Quantum Evolutionary Algorithm in Solving Course Timetabling Problems. In *Advanced in Soft Computing*. Springer, 2011.

[14] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[15] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

[16] D. Zhang, Y. Liu, R. M'Hallah, and S. C. Leung. A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. *European Journal of Operational Research*, 203(3):550 – 558, 2010.