Conversational Pattern Mining using Motif Detection

by

Nicolle Garber

Submitted in partial fulfillment of the requirements for the degree
Master of Information Technology (Big Data Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

May 2020

Publication data:

Nicolle Garber. Conversational Pattern Mining using Motif Detection. Masters thesis, University of Pretoria, Department of Computer Science, Pretoria, South Africa, May 2020.

Electronic, hyperlinked versions of this thesis are available online, as Adobe PDF files, at:

http://https://dsfsi.github.io/

https://www.up.ac.za/

Conversational Pattern Mining using Motif Detection

by

Nicolle Garber
E-mail: nikigarber@gmail.com

Abstract

Conversational mining has become a subject of great interest due to the explosion of the consumption and generation of social and other related online media. Supplementing this is the advancement in pre-trained language models, and other embedding techniques which have helped us to leverage these important sources of information. Conversation is an interesting domain to analyze in terms of its complexity and value in society. Complexity arises because a conversation can be asynchronous and can involve multiple parties. Additionally, it is computationally intensive to process. A lot of value can be derived from mining conversations, particularly in goal-oriented domains. We use unsupervised methods in our work in order to develop a conversational pattern mining technique. This negates time consuming, knowledge demanding and resource intensive labeling exercises. The aim of our work is to extract short segments of repeating ideas in conversations, also known as motifs. This can be useful for searching for particular conversational cues as well as clustering, understanding and characterizing sets of conversations. The task of identifying repeating patterns in sequences is well researched in the field of Bioinformatics. In our work, we adapt this to the field of Natural Language Processing and make several extensions to a motif detection algorithm. We split the work into two major parts. The first is sequence creation, which entails converting sequences of text (conversations) into meaningful numeric sequences. The second is the emphasis of the work, whereby we focus on the explanation of the motif detection algorithm, its adaptation and extension. In this research, we use used the Gibbs Sampling algorithm. In order to use this algorithm in a language setting, we modify the algorithm to cater for vector-valued sequences. These sequences are representations

of conversations. They are constructed by grouping phrase-level embeddings via a community detection algorithm on a graph, in which relationships between phrases are edge weights. These groups of phrases form logical phrase abstractions which we call phrase classes. In order to use phrase classes as a base for the motif-detection algorithm, we reduced the phrase class dimension using the UMAP algorithm. We demonstrated the application of the algorithm on a dynamic, real world open-source film script data set. In this study we ran an exploratory investigation into the types of motifs we were able to mine.

Keywords: Conversations, Natural Language Processing, Pattern Mining, Bioinformatics, Motif Mining, Clustering, Sequences

Supervisors : Dr. V. Marivate
Department : Department of Computer Science
Degree : Master of Information Technology in Big Data Science

"The most fruitful and natural exercise for our minds is, in my opinion, conversation."

Michel de Montaigne, The Essays: A Selection

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

The Merriam-Webster dictionary has three definitions of the word "motif". They are:

- "A usually recurring salient thematic element (as in the arts), especially : a dominant idea or central theme",
- "A single or repeated design or color"
- "Biochemistry : a distinctive, usually recurrent, molecular sequence (as of amino acids or base pairs) or structural element (as of secondary protein structures)"

The central idea of a motif can be viewed as the varied expression of an abstraction of a recurring sequence. The specific definition of motif we use in our work will be defined in the next Chapter. In general, a motif is an abstraction of subsequences that are the most common among a group of sequences. In other words, one structure of an example describing various instances, allowing for local variation and differences among the instances. This grouping assists researchers with capturing the dynamics and different ways in which people's conversations can evolve. This is the focus of our work. If we are able to express conversations as sequences, we can extract frequently repeating subsequences. We mine conversational data in this way and demonstrate this by focusing on film scripts of a comedy genre in which tropes are well known to recur throughout films.

1.1 Research Rationale

Successful motif detection in conversations can prove useful in many applications. Upon application to a specific domain, finding conversational cues can assist with intelligent conversation labeling. In retail, finding motifs in conversations may yield to more instructive cues for chat-bots or finding commonalities in customer sentiments or requests. On social media platforms such as Twitter, it could lead to recognizable cues which might assist in event detection. In any context, quantifying conversations gives us the power to abstract, index and search the subsequences in meaningful ways. This helps us to extract important signals and cues from a specific sequence.

1.2 Research Aims

We investigate the field of Bioinformatics in which motif detection forms a large and fundamental part of understanding important biological sequence structures such as DNA and proteins. We expand on this in Chapter 2. Our aim is to find similar functional units of human conversation.

1.3 Objectives

In the real world, conversation produced from goal-oriented domains has the advantage of having more limited conversational cues. It is therefore expected that clearer motif patterns might emerge. In our work, we explore simulation of motif mining in a broader and more varied conversational setting such as an open-sourced dataset of film scripts. We focus our attention on one particular genre of the film scripts in order to constrain the diversity of the underlying conversations. It is well known that the comedy movie genre makes good use of tropes and movie motifs [31]. Additionally, it is clear after some investigation that the comedy genre has the largest set of conversations of any genre, which allows for richer sequence data in order to run motif detection. We show the types of motifs we are able to extract in Chapter 5. We achieve this by developing a pipeline which is fine tuned to our conversational dataset. This leads to understanding the feasibility of building a more generic tool. We avoid the need for expensive infrastructure

and complex language modeling by using pre-trained embeddings for the basis of our work.

Conversational mining is an important field which has not been explored enough. Most of the focus in the field is placed on conversation generation (this is expanded on in Chapter 2). An important idea we explore is converting conversations into meaningful sequences. Having such sequences allows practitioners to explore not only motif mining but many other sequence mining techniques. This can be used as a stand-alone analysis or for feature extraction which can aid in downstream supervised tasks. Another contribution we make is in combining the motif detection techniques used in Bioinformatics with those of Natural Language Processing (NLP). Both fields are largely centered on sequence structures where location and context are important. Lastly, in any domain where mining conversations can lead to increased insights, mining conversational motifs allows us to find generic representations of human communication. This alleviates the difficult task of making sense of the ever increasing language-rich data.

We make several extensions to our base motif detection algorithm (explained in Chapter 3) which on a high level include varying sequence lengths, finding a descriptive 'global' motif and altering the algorithm to recognize more sequence elements.

1.4 Research Questions

The research questions being addressed in this study are:

1. How can we characterize a conversation as a meaningful sequence?
2. How can we apply the Bioinformatics tool of motif mining on to a set of conversational sequences?

1.5 Contribution

For the benefit of the reader, we summarize the main contributions we put forward in our work.

1. Application of motif detection commonly used in Bioinformatics to the field of NLP.
2. The idea of applying sequence mining to the realm of conversations.
3. Gibbs sampling motif detection algorithm extensions including:
   (a) Adaptation for use on any sequences (In Bioinformatics there is a finite set of elements in the sequences (typically "A", "T", "G", "C").
   (b) Adaptation for use on varied sequence lengths.
   (c) Adaptation for algorithm to consider sequence elements which are not categorical.
   (d) Inclusion of the importance of finding the global motif - or abstract class which tries to characterize most of the underlying instances.
   (e) Ability to calculate a local sequence alignment score to understand how well certain sequences align with the global pattern. We are able to exclude irrelevant sequences which are less likely to contain this motif.
4. The idea and implementation of finding conversational phrase embeddings

Our research is presented as follows. In Chapter 2, we provide a view of previous work done and attempts to merge research from across the Bioinformatics and NLP domains. Next, we provide an overview of the system we have built at a component level in Chapter 3. A modular pipeline design is employed for the system which comprises of two main components: the conversion of conversations into sequences and the pattern mining component applied on top of these sequences. Each component has multiple parts and these are examined in the results Chapter 4. Once we understand the purpose and functionality of the system, we can look at general applications. Our pipeline is customized to the movie dataset in the application (Chapter 5). In conclusion, Chapter 6 distills the key-points and provides a short discussion. The possible extensions and further applications of the system are considered.

Chapter 2

Literature Review

It is important that we review different areas of prior work. Firstly explaining what a motif is, how it can be used and why we are using this concept is essential. Next, we describe an overview of what the field looks like and the reason we are considering it in this field. Thereafter, once we have understood this type of pattern mining approach, we look into which conversational mining approaches have been applied. We review whether or not there have been any similar attempts at combining the fields of NLP and Bioinformatics. The power of building motif detection algorithms on conversation lies not just in finding out new information but being able to use it for other applications such as clustering conversations based on patterns. In a conversational setting, especially in a goal-driven one, the generic structures of a conversation are just as important as any other type of clustering related to the immediate relevance of the conversation.

The reader might be interested to note that while motif detection associates very highly with the field of Bioinformatics, the concept is being applied in several different areas. For example, motifs can help to uncover network structures which can find application not only in Bioinformatics, but also in communication science and software engineering [33]. The former two are active areas of research and there are several survey papers on both [11, 39, 47, 15, 20], for example. Motif detection has been used before [21] to understand information processing in the brain via Variational Autoencoders. Motif learning was used in a very interesting way [49] to discover higher-order similarities in

heterogeneous information networks for use in recommendation systems.

2.1 Motifs in Bioinformatics

In this section, we first provide an explanation of what motifs are. We outline how they are being used in Bioinformatics. We then explain how we define them and why we are using them.

A motif is a characterization of a short subsequence of DNA [15]. This sequence typically indicates a significant biological structure such as DNA binding sites for regulatory proteins [15] (a common application in the field). Segments of DNA which are used for transcription processes are called genes, and the information contained in these help with making proteins in a process known as translation. In many cases, the transcription-translation process is a function of the genes of a sequence. Finding these patterns is one of the most important and challenging fields of molecular biology and computer science [15].

Motif discovery can be phrased as the detection of 'functionally significant short, statistically over-represented subsequence patterns in a set of biological sequences' [22]. In other words, this is the process finding a pattern that repeats or is more likely than just a random selection. Discovering motifs and their locations provides a clue about the transcription factors that control gene expression. Consequently, a lot of work has been done in the area to develop tools and different algorithms for this purpose [45]. We touch on some of the work done in the field to illustrate both the advancements and the depth of existing work. This is intended to provoke questions about how to leverage this more in other areas.

The challenges of motif detection lie in the fact that any real data as in DNA sequence data (or in our case conversational sequences) contains noise. In biological sequences, this is expressed as mutations, insertions or deletions of nucleotides [15]. Additionally, motifs can either occur in the same gene or in multiple genes. Some genes may not have

them [15]. More traditional techniques search for exactly one motif per sequence. This may miss some motifs or conversely detect motifs in sequences which may not contain them. Since most motif detection algorithms look for some type of global commonality between motifs in sequences, sequences that do not have this motif would add noise to this process. For these reasons, motif detection algorithms typically need to be more complex than pattern matching or brute force algorithms.

One way motifs are found is by searching for statistically over-represented subsequences in orthologous genes (genes from different species that evolved from a common ancestor and are likely to have the same function) [4]. Statistically over-represented subsequences are those that occur more frequently than random [15]. Algorithms based on this line of thought work better in 'lower organisms' such as yeast. This has been addressed using phylogenetic footprint algorithms. Subsequently, this method was combined with sequences from co-regulated DNA [15].

Earlier work relies on two main categories of algorithms in order to detect motifs. The first category relies on frequency and string counting based methods. Even though these led to globally optimal solutions, they were best used for finding short motifs [15]. The second category of approaches worked better with longer motifs. These are probabilistic models which make use of some type of position weight matrix. Our work incorporates this idea. Each position of a motif is represented by the various probabilities of a 'letter' (sequence element) in that position. This type of structure deals better with weakly constrained positions where variation is likely [15]. These methods are not guaranteed to find a global optimum and make use of more local search technique methods [15]. An example of a way to measure a subsequence as a motif has been described by Tompa [44] as explained by the review paper authors of [15]. The probability of finding a motif s in N number of sequences can be characterized with a statistical significance test which includes the formulation of a z-score. This is asymptotically normally distributed with mean 0 and standard deviation 1. This score makes it possible to compare different motifs. The z-score is calculated as follows:

$$M_s = \frac{(N_s - NP_s)}{\sqrt{(Np_s)(1 - p_s)}} \tag{2.1}$$

where N_s is the number of sequences containing occurrence of s (sequence motif) and p_s is the probability of at least one motif s in a sequence [15]. Not only have different measures and scores been developed, but motif detection has been applied to a number of different structures and organisms (for example yeast, mouse DNA, etc.) [15]. Different types of methods have been used, for example: word-based algorithms centered on regular expression pattern matching and a few variations of suffix trees. In terms of probabilistic algorithms, Hertz et al. [19] developed a greedy probabilistic sequence algorithm which found the site of highest information content - finding exactly one instance of a motif in every sequence. NestedMICA [16] uses Independent Component Analysis to separate multiple motifs from the signal simultaneously and is able to extract longer sequences. A number of expectation maximization algorithms have been formulated as follows: Lawrence and Reilly [25] formulated a method which looks for at least one common site, where a missing information principle is employed. The famous MEME algorithm by Bailey and Elkan [2] aims to discover new motifs with little prior knowledge in specific application to bio-polymers. A few new ideas introduced here are as follows: the starting point of the algorithm uses existing subsequences in order to increase the probability of finding globally optimal motifs. The authors remove the assumption of exactly one occurrence of the shared motif occurring in each sequence. The authors introduce a method of probabilistically removing shared motifs after they are found in order to find several distinct motifs. Many attempts have been made to model or discover motifs by the use of genetic algorithms [26]. Various model representations, fitness functions, genetic operators and data post processing techniques are surveyed in the afore-mentioned work. More recently, the field of Bioinformatics has also benefited from developments in deep learning. For example, a model called DeepBind [48] applied a Convolutional Neural Network (CNN) to identify DNA or RNA protein binding sites [37]. There are examples of other types of architectures used such as combinations of deep belief networks and CNN (iDeep) as well as an example of CNN and Long Short Term Memory Network (LSTM) combination in iDeepS [37].

It is natural to think about leveraging a field with such a concentration of work and relative maturity. With this understanding of how motifs can be used in a biological

context, we bring examples of how they are leveraged in general time series and explain why and how we make use of them. In our work we make use of some of the earlier or more classical techniques in order to perform motif detection to prove the concept.

2.2 Motifs in Time Series

The authors of [30] were among the first to use the idea of motif learning in genetics for application in time series. The definition the authors introduce for their idea of motifs is the patterns describing "the enumeration of previously unknown, frequently occurring patterns". Similarly to our work, the authors surmise that the mining of such patterns can supplement further downstream tasks such as clustering, classification and the building of association rules. Another benefit of learning motifs for time series, as noted by the authors, includes the visualization and summarizing of large time series databases. The authors introduce a method 'EMMA': Enumeration of Motifs through Matrix Approximation. This algorithm moves a sliding window of length n across the time series and uses a hash function to normalize it. The hash table serves as a heuristic for motif search which works on the premise that an over-represented pattern in the time series would hash to the same location. All subsequences that are similar enough but hashed to different locations are considered by computing candidates within a least upper bound.

More recent work on motifs in time series aims at addressing more computational type of problems. For example, the authors of [41] work on the problem of extracting motifs from a single very long sequence. At the time of its writing, it was the only algorithm to be able to efficiently scale to sequences of gigabyte lengths, handle large alphabets and support multi-core processing. [23] proposed a memory efficient, online algorithm capable of finding more than one key motif. There are many applications mentioned including: motion-capture, telemedicine and severe weather prediction. The authors highlight a differentiation between discrete representations of real-values time series and exact motif searches. It is argued that many domains including robotics, online compression and sensor-networks require online motif discovery of several motifs. The

largest emphasis was placed on an interesting application including finding significant motifs in brain activity which may predict seizures. Most recent work in motif detection uses sophisticated techniques. Since our study is exploratory, we focus on the more traditional motif detection techniques.

2.3 Conversational Mining

Before we can dive into techniques aimed at understanding conversation better, we briefly remind the reader about the more general field of Natural Language Processing (NLP). There is no single accepted definition; however, the general meaning is quite consistent. That is: a set of computational techniques aimed at understanding, summarizing and representing text. This is done with the aim of striving towards human-like capabilities of generating and processing text for a diverse set of tasks [7].

The most common type of work done in computational fields with regards to conversations is various methods and enhancements for dialogue systems which maintain dialogue state and can be used for generative conversational agents. The survey paper [8] describes a good investigation of advances in the space by dividing dialogue systems into task-oriented and non task-oriented models. Recent work conducted by [18] focuses on surveying different aspects of spoken language understanding in order to build better conversational systems. The recent work by [43] addresses building a domain driven bot by separating content selection for candidate responses from response composition. This was integrated into the Google assistant. The task of dialogue generation is approached in [9] by a hierarchical variational memory network. This aims to abstract variations as well as long-term dependency modeling such as is necessary for dialogue state tracking. A different approach is followed by [27] where multi-turn conversation is modeled using a dense semantic matching network which leverages context-response pairs to find an optimal candidate. An important aspect of research in dialogue systems includes examining evaluation metrics of generative models. The authors of [29] provide recommendations for building metrics that perform better than the technical metrics created for machine translation in this domain.

Some earlier work [28] places focus on performing analysis work on conversational patterns in social media. The focus of the work is the examination of product launches on Twitter. Conversational analysis is leveraged in order to create concept maps to compare conversations. A concept map is defined as a network that follows the conceptual progression of a conversation (in this context) by keyword representation which can lead to the discovery of topic clusters which are related to each other. The approach appears to be quite handcrafted - requiring specific pre-processing for domain and the creation of domain-specific dictionaries. Traditional network analysis is applied in order to extract the concept map and topic flows. Semantic complexity cannot be modeled through single keywords and the abstract structures that are extracted are vague. The notion of conversational structure abstractions is not brought forward. In other words, the abstraction is mined in the form of a word cloud so that semantic complexity can be modeled. Another example of analysis-style work is from a more recent paper which leverages a social media platform Gab in order to detect 'echo-chamber' patterns by [1]. The structural abstraction of conversations which we extract in the form of motifs is extracted in this work in the form of cascades. This is a directed graph where the set of vertices represent content (posts, replies, quotes etc.) and the edges represent replies and quotes (or interactions with content). This assembly evolves over time with the original post forming a 'root node' of sorts, and subsequent interactions between text form levels. While these cascades can model linear and non-linear interactions, there is only brief work done on modeling the content and conversational queues (in the form of hashtags) and most emphasis is placed on response time and spread modeling of the different types of cascades.

We see that motifs are not applied to conversations, to the best of our knowledge. In fact, the conversational pattern mining field is saturated with work on dialogue management systems, bots, assistants, dialogue act prediction and question answering. There is a lack of data mining techniques for conversation. There is no way to understand generic structures of conversations. There is very little work done on converting conversations

into either embeddings (as we have done) or at least discrete or continuous representations over time. Motifs assist with understanding generic conversational structures which can be used for a host of use cases - from segmentation of important parts of conversation, to search, understanding, prediction, visualisation and clustering. With the understanding of generic structures of conversation (just as in Bioinformatics) we can begin to assess individual deviation from global structures. This enables us to spot anomalies. Understanding human conversations in a domain better - (particularly cues in goal-oriented domains) may even allow us to spot negative conversational patterns on social media. There are many questions we can answer by applying data mining techniques to conversations. We have built our system with this goal in mind. The base of our pipeline is built on the prominent work of Lawrence et al. [24] using the traditional Gibbs sampling motif detection algorithm. In the field of Bioinformatics, sequence elements are finite sets consisting of four bases or twenty amino acids. Extending this into a more unstructured domain requires processing larger sets of sequence elements (which may reach thousands of classes). In our case, the sequence elements represent conversational phrases. In Chapter 3 we explain the motif detection algorithm as well as the pipeline built around it which enables us to process conversations.

Chapter 3

Methodology

3.1 High Level Overview

Detecting motifs in conversations is done in two parts. The first one entails converting conversational text into representative sequences. The second part involves the application of the motif mining algorithm on top of these constructed sequences. Each of these parts has a collection of algorithms and processes. In this Chapter we explain the technical details behind these. We justify how technical choices fit in to the broader picture, what these choices are and how they were made. The two parts are the underlying components are illustrated in a higher-level view in Figure 3.1.

3.2 Sequence Creation

We are mining motifs in conversations using sequence mining techniques. This section covers how we transform raw conversational text into sequences.

The level of modeling of the conversational text follows the logic of how we would like our sequences to look. Since one conversation is captured in one sequence, each element of the sequence should represent a phrase. Since a lot of reference is made to these phrases, we aim to represent these phrases in a generic way by referring to them as "phrase classes". Each phrase class should capture the semantic, emotional and

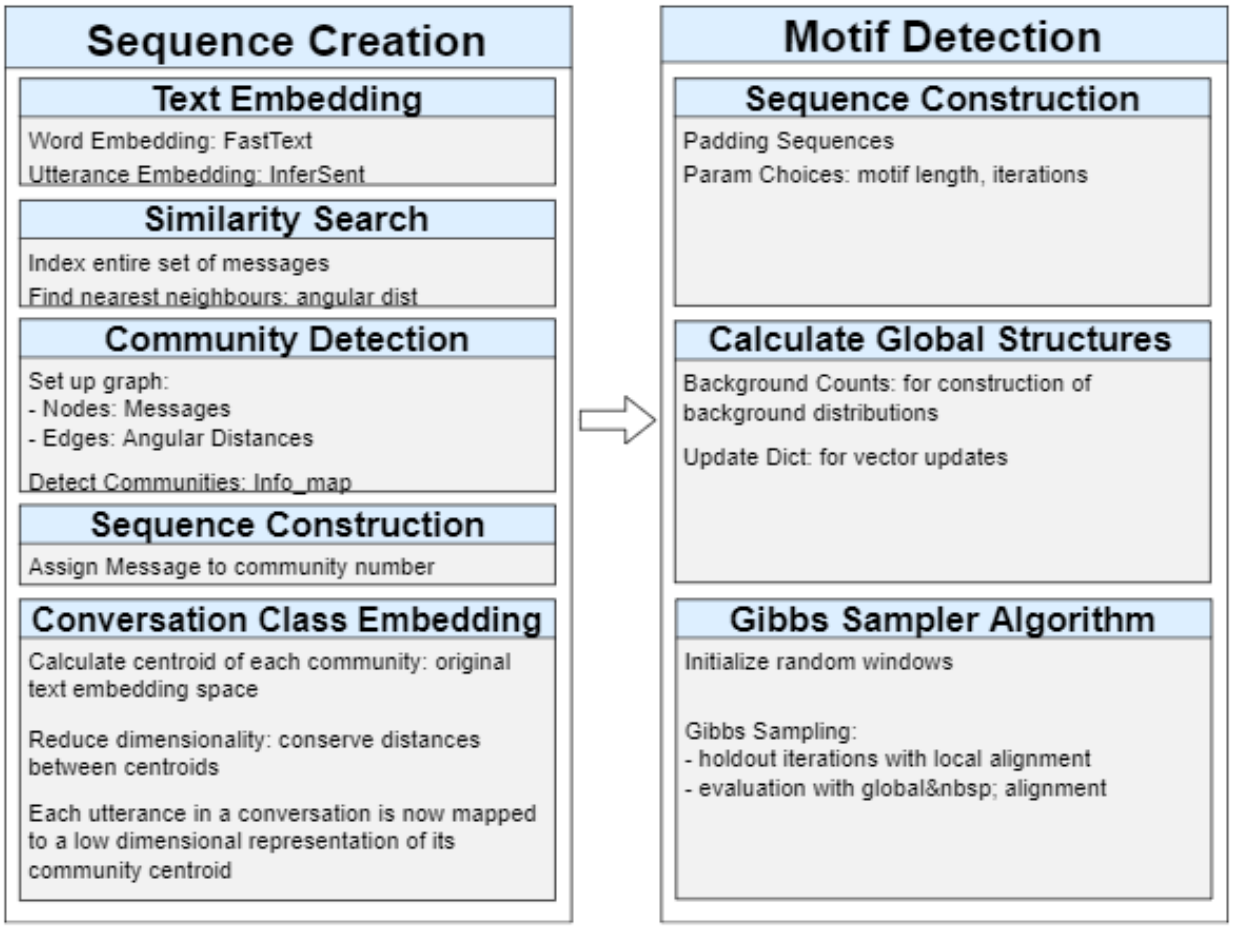

Figure 3.1: Modular representation of pipeline for conversational motif mining strategy

conversational meaning of a phrase within the context of the conversation. We can view this as an analogy of the biological DNA sequences. Within each sequence are a set of ordered elements. The unique set of these elements ("A","T","G","C") make up a vocabulary which is used for the motif detection. In our case, we would like to construct a vocabulary consisting of conversational phrase classes. An example of a phrase class could be "greeting". This would be a generic representation of the underlying phrases

such as “hello there”, “greetings”, “heya”.
The method we use to construct a generic representation of phrases is by grouping phrases together that contain similar semantic and conversational information. These groups of phrases can be assigned a generic class. Naturally, this calls for unsupervised methods which we describe below. The generic class we assign to a group of phrases can be categorical. By doing so we would be replicating the analogous biological sequence vocabulary. However, we acknowledge that some phrase groups may be more similar to each other than others. The higher-level methodology includes finding phrase embeddings, clustering the phrases on a graph and reducing their dimensionality. This is done using methods that try to preserve the distances between phrases. Each of these processes we explain in the subsections that follow.

3.2.1 Text Embedding

We find phrase level embeddings in order to capture the dynamics of spoken phrases in a conversation. We do this by performing word level embedding, followed by sentence embedding. In our application, the good structure of the text we are dealing with (minimal spelling mistakes, special characters and mixed languages) allows us to leverage pre-trained embeddings. This avoids the resource expensive task of training our own embeddings or the need for fine-tuning.

The word vector representations are derived using the well known continuous bag of words (CBOW) model where each word is represented by a bag of character n-grams [35]. The central idea is that by leveraging clever techniques and massive language datasets, it is possible to construct high quality, generalizable embeddings allowing NLP practitioners not to have to train their own. The authors make use of the common crawl dataset in order to train their model [36]. The authors describe a few main enhancements to the CBOW architecture. Very frequent words are sub-sampled, there is use of 10 negative examples for training. To prevent the loss of ordinal information in a context because of word averaging, positional weighting is included. This occurs by a varied weighting scheme which is dependent on position. The loss of word order information is countered by including a preprocessing method which iteratively merges tokens with

high mutual information. Repeated sentences are removed. All these enhancements are combined with taking sub-word information into account by representing words as a sum of their character n-grams. The most notable results highlighted by the authors include accuracy gains for seven benchmark tasks.

We obtain phrase embeddings by encoding the FastText word embeddings with the InferSent model [12]. The authors explore the idea of using supervised learning in order to create a sentence encoder generalizable to other tasks and suitable for transfer learning. The thinking behind this is that much like the field of image processing was propelled forward by pre-trained ImageNet features, the field of NLP can use similar concepts. Various implementations of word embeddings for transfer tasks have been successful. Sentence or phrase embeddings, on the other hand, have been a more difficult task because the relationships between words and phrases are harder to capture in a single vector. The authors hypothesize that structuring this problem as a Natural Language Inference (NLI) task will capture more abstracted reasoning about semantic relationships within sentences and thus have a good transfer accuracy. The training corpus chosen is the Stanford Natural Language Inference [5] which is the largest human-annotated and generated English sentence pair dataset with 570 thousand pairs. Each of these pairs falls into three classes: entailment, contradiction and neutral. Multiple architectures are tested by the authors [12] with a bi-directional LSTM including max pooling for the sentence encoder being emphasized. The authors explain how this outperforms most of the existing alternative unsupervised approaches at the time of writing. It is also found that the accuracy on 12 tasks with transfer learning is very high. The relations between the sentence pairs are extracted in three ways: a concatenation of the two vectors, an element-wise product and an absolute element difference. The bidirectional LSTM with max pooling has an architecture consisting of the concatenation of a forward LSTM and a backward LSTM applying max pooling over the hidden layers. We apply this sentence encoder to the phrases of our conversation in order to start leveraging phrase-level embeddings.

3.2.2 Phrase Classes

We have represented each phrase of a conversation as a vector. This vector contains a lot of information ($x \in R^{4096}$). We chose to work with phrase classes as inputs into the motif detection is because phrase embeddings are infeasible for analysis and not interpretable. The former is due to the computational costs associated with such high dimensional sequences used as inputs motif detection algorithm which performs sequence scans and comparisons. The latter is attributed to the loss of hierarchical information which allows us to abstract the meaning of a phrase in terms of conversation and content. Phrase classes allow us to look at sequences as conversational cues and model on a more abstract level. The results can then be interpreted and can tell us more information about the conversations we are working with.

In order to find phrase classes, we performed semantic grouping. From a practical consideration (such as application explained in Chapter 5), we have 93248 utterances which have 4096 dimensions. This dataset is too large to perform simple clustering on. We has to resort to techniques that outperform simple clustering because of the consideration of the type of embedding space as well as positional relativity of groups in the dataset. The result of this is a pipeline that consists of the following succession of tasks:

1. Run an index on the dataset for fast KNN query [6].
2. For each phrase in the dataset, search for the 10 nearest neighbours based on angular distance.
3. Create a dataset in which each row has phrase, neighbour and angular distance.
4. This forms the structure on top of which a graph is built. The phrase is the source node, the neighbour is the target node and the edge weight is represented by angular distance.
5. There will be groups of common phrases that form - these are picked out as communities.

6. Represent a community as a centroid of the embeddings of the phrases within it.

7. Centroids are still high dimensional, and some communities are more similar to each other than others.

8. Select a distance-preserving dimensionality reduction technique that best conserves the relationships between communities.

9. These dimensionally reduced vector representations of the centroids form the classes of each of the utterances in the conversation.

Each phrase class captures both the semantic similarities of phrases and conversational functions. Since we represent the phrase classes with a reduced representation of the phrase embedding, classes more similar to each other are located closer together as measured by cosine similarity. We have developed a way to work with continuous representations rather than categorical phrase types. This is a key feature, allowing motif detection to discover similarities in a more general way. In the next few paragraphs, we explain a few of the details related to the methods in the steps we have mentioned above.

After obtaining phrase embeddings which represent the nodes in our graph, we find the edges by detecting the 10 closest neighbours by angular distance. The KNN classifier as described by [17] is a decision rule-based classifier which assigns a classification to an unlabeled point based on the classification of the nearest classified points. We make use of NMSLIB [6] : Non-Metric Space Library which is specifically designed for efficient similarity search based on various non-metric spaces. It is very fast and efficient. We make use of the angular distance defined by:

$$d(x, y) = \arccos\left(\frac{\Sigma_{i=1}^{n} x_i y_i}{\sqrt{\Sigma_{i=1}^{n} x_i^2}\sqrt{\Sigma_{i=1}^{n} y_i^2}}\right)$$

It is found that this distance works the best when dealing with our embeddings. Additionally, it is known that cosine distance works well with embeddings produced by models where the objective function was represented by a dot product with softmax. The algorithm uses Hierarchical Navigable Small World Graphs (HNSW) which builds a representation of stored elements incrementally through proximity graphs. This search technique has a

logarithmic complexity with very high performance. The KNN algorithm is implemented with search for the 10 nearest neighbours of every phrase.

By utilizing the KNN techniques, we find close relationships of phrases. If many phrases share many neighbours (to varying degrees), this will be well captured in a graph structure. We set out the graph such that we set the source as the original query phrase and the target as the neighbour. The angular distance between source and target node forms the edge weight between them. The task is to find the communities that form. Phrases fall into communities where there are many similar phrases. The quality of embeddings used means that any semantic as well as syntactic similarity are captured within the communities.

The community info_map algorithm was first introduced in 2008 by the authors of [40]. The idea is to minimize the map equation (objective function) over all partitions of the network. The choice of partition of the network reflects information flow within the network with more flows within the modules (forming partitions). The map equation minimizes the description length of a random walk throughout the network. This equation describes the entropy of the movement between the modules and the entropy of the movement within the module. A deterministic greedy search algorithm minimizes the equation by looking at the fraction of time each node is visited by the random walker. The random walker will spend more time in certain clusters, affecting the probabilities. We use this algorithm as a base to obtain our community clustering (which in our application case consists of 3232 communities - Chapter 5). There are many smaller communities which may be very similar to larger ones. At this point in the pipeline, we can choose to label each community categorically. If we do this:

1. We lose relational information of phrase classes relative to other phrase classes.
2. We lose meaning about the phrase classes.
3. We are forced to deal with the phrase classes in a categorical manner which has many implications for the motif detection algorithm.

Instead, we embed these communities into a space where it is possible to retain information

about how communities relate to each other (in terms of semantic and phrase meaning). Here, it is possible to use the same distance measures as are applicable to the embedding space. This community embedding is achieved by calculating the centroid phrase embedding of a community. We choose a dimensionality reduction algorithm which conserves the distances between these centroids.

We have calculated phrase classes mapping each phrase to an abstract representation. Phrase class dimensionality is reduced by conserving the pairwise distances between the centroids. Manifold learning is a good approach for this task as we are dealing with a high dimensional space and the reduction method cannot be linear. This method is chosen after testing and comparing multiple algorithms. (We do not present these here as these details are not applicable). In order to select the best algorithm, we select the one that maximizes the correlation of pairwise cosine distances before and after reduction. We tune the parameters by grid search. The Uniform Manifold Approximation and Projection (UMAP) algorithm developed in [32] is a non-linear dimensionality reduction technique. It has a fast run time (including scalability) and preserves global structure. It is put forward by the authors as being faster and more accurate than the t-SNE algorithm. We observe this in our experiments. On a very high level, UMAP learns a topological representation of the high dimensional data. Given some low dimensional representation, the same process is applied to learn the local manifold patch representation of the data (fuzzy topological sets). The assumption is made that the data is uniformly distributed on the manifold as it is possible to find a metric such that the data will be uniformly distributed based on that metric, provided that the manifold has a Riemannian metric. In order to represent this in a computational way, the authors make use of weighted graphs. The approach consists of two phases in which the first phase constructs a weighted KNN graph. Given a metric and a chosen k, the set of nearest neighbours is computed for each point. These parameters must be tuned for application. The weights are calculated based on the distance between neighbours. A force directing algorithm is applied on the graph until convergence. This provides an estimation of the fuzzy topological representation. The low-dimensional representation of the data is then re-arranged based on minimizing cross-entropy between the high dimensional data and the

low dimensional data. The vectors produced after the dimensionality reduction are the centroids of the phrase vector embeddings on which we can keep using cosine distances.

We can now represent conversational data as sequences of five dimensional vectors (phrase classes) in a Euclidean space with cosine distances preserved. We achieve only marginal gains by increasing the number of reduced dimensions, which is how we select five dimensional representations of the centroids of the communities.

3.3 Motif Detection

In this section we explore the most important component of our pipeline. The inputs to this algorithm are contextual conversational sequences. We capture the nature of the abstract class as well as the relationships between these classes. We are now able to pattern mine these conversations (sequences). We are most interested in detecting the development of conversational cues that follow a generic pattern common to most or all the movies. To remind the reader of what we mean by a motif in this context, we start by explaining that a motif is a subsequence of a fixed length that is the most common to all sequences.

We look at a Gibbs sampling method that was originally proposed by Lawrence et al [24]. We first describe the method in this paper and then our implementation and algorithm extensions. Gibbs sampling methods have been used at length for motif detection. The benefits of this particular implementation as described by the authors includes the fact that it runs in N linear time for N sequences (it is computationally efficient) as well as allowing for great variability among the patterns that are found. Human conversations are not robotic or a prescribed set of exact rules and as such even if there are generic motifs that are found, they will have variation among them. For example, in a standard greeting sequence, there may be a generic pattern which includes the following turns:

1. Character1: greeting phrase
2. Character2: greeting phrase

3. Character1: generic response phrase

4. Character2: generic response phrase

5. Character1: queue for initiation of specific topic

6. Character1: response of initiation of specific topic

If a third character is introduced, variation will be introduced; however, the underlying conversational structure should not change. The abstract conversational cue must still represent a greeting exchange. There are many conversational examples of the idea of an abstract motif which may contain variations within its different instances. In gene sequences, this variation is caused by factors such as mutation and as such it is a consideration of the underlying algorithm.

The base algorithm as described by the authors has a key simplifying assumption. This is that each sequence is assumed to have exactly one motif. The Markov assumption is also applied. The objective is formulated as follows: given a set of N sequences $S_1, ..., S_N$, find the set of N subsequences which are the most similar to each other. A subsequence is of a fixed and specified length W. The measure of similarity of the sequences is one which maximizes the ratio of pattern (motif) probability to background probability. When there is a good alignment of motifs or positions in each sequence where the segment maximizes this ratio,the iterations of the algorithm stop.

This ratio looks as follows: $A_x = Q_x/P_x$. A_x is described as the probability that some sequence z contains a motif at position x. At each iteration, one of the sequences is held out. There is a sliding window over the sequence starting at some current position. At each position of the window of length W, the probability P_x of generating the sequence using background probability is computed. The probability profile elements $q_{i,j}$ [24] are described by:

$$q_{i,j} = \frac{(c_{i,j} + b_j)}{(N - 1 + B)} \tag{3.1}$$

where the c term represents the count of amino acids that occur in the i^{th} position and the b term is the background frequency of the amino acid. A weight equal to A_x is assigned

to a segment at position x. A random segment is selected from all the window candidates according to the weights A_x. Once some correct a_k have been selected by chance, $q_{i,j}$ begin to reflect a pattern existing in other sequences, the segments probabilities A_x become more strongly characterized and the algorithm tends to favour further positions in other sequences that confirm these patterns.
There are several constraints of the motif algorithm. These include:

- Constraining each sequence to have exactly one instance of a motif.
- A fixed pattern of length w.
- A fixed sequence length.
- Some support for variation of local motifs from global by using probabilistic alignment.
- The algorithm is stopped after a number of iterations as opposed to a convergence criterion.

These are considerable limitations which have practical effects on not just our particular application but on most applications we could consider motif mining for. For real applications, we may not know which pattern we are searching for or which sequences contain this pattern. Forcing each sequence to contribute equally to the pattern detection may introduce significant noise into the process. If a sequence has more than one instance of the same pattern, this would not be picked up by the algorithm - a separate match algorithm would have to be run. A fixed motif size is difficult to fine tune. We may not know what type of motif we are looking for and there may exist a range of possibilities to choose from. A fixed motif length may also impose a tight constraint on the detection of a global pattern against local sequence patterns. For example, some sequence may express a motif with more noise in between its elements and have a longer length. The next constraint, a fixed sequence length is also not a practical limitation. In our application, our movie conversations are of different lengths. In order to shape sequences to the same size we may need to add extra consideration to how and where we split them. This process adds extra steps and might result in information loss. Support for variation within the motifs is ideal for conversations and any time series, as we have mentioned

before. The fact that the algorithm is heuristic and does not have convergence criteria makes it easy to stop prematurely or conversely to have long and unnecessary run times which add marginal or no improvement.

We use the algorithm implemented by [42] as a base for our own implementation and extensions. We address multiple things. Firstly, we address the challenge of implementing motif detection in a completely different setting such as conversational mining. Secondly, we make extensions to the original algorithm that allow it to work not only in this setting but in a multivariate time series setting. This may allow a practitioner to overcome a seasonality constraint in searching for time series patterns and look for irregular patterns in a way that is more robust to noise. We start the description with a high-level explanation of the additions we have made and then a brief holistic discussion of the entirety of the algorithm. These include:

1. Adaptation for use on any sequences (In Bioinformatics there is a finite set of elements in the sequences (typically "A", "T", "G", "C").
2. Adaptation for use on varied sequence lengths.
3. Adaptation for algorithm to consider sequence elements which are not categorical.
4. Inclusion of the importance of finding the global motif - or abstract class which tries to characterize most of the underlying instances.
5. Ability to calculate a local sequence alignment score to understand how well certain sequences align with the global pattern. We can exclude irrelevant sequences which are less likely to contain this motif.

The following pseudo code illustrates a simplified framework of the motif detection algorithm.

Result: BestMotifs, GlobalPattern, Score
initialization - construct set of *BestMotifs* by placing random windows on sequences
for *i=0* $\rightarrow$ *numSeeds* **do**
  if *CurrentSet* > *BestMotifs* **then**
    BestMotifs = *CurrentSet*
    for *j=0* $\rightarrow$ *N* **do**
      select random holdout
      select best window on holdout
      if *currentSet* > *BestMotifs* **then**
        replace window on holdout
      else
        continue
      end
    end
  else
    continue
  end
end

Algorithm 3.1: Gibbs sampling motif detection - high level

In this paragraph, we describe the algorithm on a high level. One sequence is held out per iteration and an optimal window placement on the sequence is found. The placement of the window on a sequence as compared with the global best motif pattern we call "local alignment". The effect of each window placement on the full set of best motifs is considered. We term this "global alignment". This procedure is performed iteratively and randomly, allowing window selection on each hold out sequence to start improving global alignment, which improves local alignment and so on. The probabilistic nature of the matching allows for variations in local alignment while still finding optimal global alignment. The set of best motifs are only updated when a motif on the holdout sequence is found to improve the global score. The algorithm begins with random initialization of a set of windows on each sequence. These windows are of length w - the motif length. A random holdout sequence is selected from the set of sequences. An optimal placement

on the sequence is calculated by sliding the window on the sequence and choosing a best probabilistic score. Once this window is selected on the holdout sequence, its impact in relation to all the other windows on all the other sequences is assessed. If this global score is improved by the hold out sequence contribution, the motif is selected - otherwise the procedure continues, and another holdout sequence is selected. This follows the Gibbs sampling procedure.

We consider the probabilistic selection of the optimal window placement on the holdout sequence relative to the current set of best motifs. In order to assess this effect, we construct a set of foreground motif probabilities of the elements present in the current window. For each position $p_1, .., p_w$ where w is the length of the motif, we calculate the probability of a specific element occurring across all motifs (full set of current best motifs). This creates a motif 'profile' in which we have per position a probability per element of our vocabulary. We calculate a score for each window placement on the holdout sequence. This is the probability profile of the element in the window. Each of these positional probabilities (calculated as joint probabilities by the original author), we combine. We select the window which has a cumulative probability that is larger than a random probability. This element of randomness is introduced to avoid getting stuck in any particular local optimum. The sequence that maximizes the foreground motif probability is selected. If the contribution of the motif selected by the holdout sequence improves the global score, we update the set of current best motifs. This follows the base implementation of [42] and [24]. We now consider the adaptation of the update and scoring mechanisms we needed to introduce.

As previously alluded to, we need a different way of calculating the updates because our phrase classes are not discrete. The case of DNA bases or amino acids typically involves 4 classes. In our case, we have thousands of phrase classes. We need to consider how tiny our probabilities become when dealing with so many possible classes in relatively small sequences. Our probabilities tend to zero as the ratio of classes to sequence length grows. The base algorithm updates occur using simple frequencies. In our case, vectors which are closer together should be treated as such as they have a closer semantic

meaning in the space. Vectors which are further should not have as much influence. We therefore need to consider a distance measure in which the smaller the distance between the points, the larger score and this needs to be normalized between zero and one. A class should have maximum score with itself (one). Classes closer to each other need to have proportionately high scores. This is captured in the notion of similarity rather than distance. We continue using cosine similarity. We map the range from negative one to one to a score between zero and one. We pre-compute these pairwise distances as input for the algorithm for efficiency.

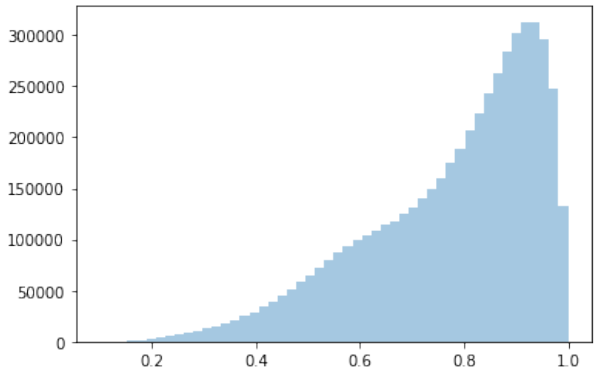

Figure 3.2: Distribution of Transformed Cosine Pairwise Cosine Distances

The Figure 3.2 illustrates pairwise distances generated between all the phrases. We use this distribution as a guide for the selection of a similarity threshold. Phrases with similarity below this threshold are not considered together for updates. If we make this point too low, we will be working with a large portion of the data for every update. If we select it too high we will miss elements that may be similar to each other in the consideration of one motif position. Following the logic, two phrases with zero cosine similarity have vectors that are pointing away from each other. Their meaning may be completely opposite.

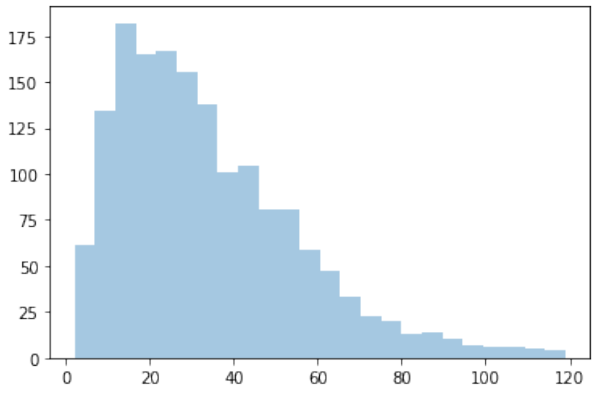

Figure 3.3: Distribution of transformed cosine pairwise cosine distances

Due to the highly left skewed distribution, we select a narrow cut off at one percentile of the data post transformation (at threshold value of 0.99). For one phrase class being calculated, the effect of approximately 20 closest phrase classes is taken into consideration. Figure 3.3 illustrates the number of elements that are considered with the update of one phrase class.

An example which illustrates the concept of phrase classes which get considered together during updates can be seen in Table 3.1. We consider a large phrase class: Class 5. Class 24, 30 and 8 have successively decreasing similarities with Class 5. Class 5 is described by phrases of different variations of the negative: "no". It is very assertive. The next most similar phrase class has a polite negative: "no thank you". Class 30 which is less similar describes something that is not as close to a direct negative - it is a negation. This includes different variations of "nothing". Class 8 which is the least similar in our sample consists of phrases calling entities - different names. It is clear why these need to be considered together and not as separate elements in the motif detection algorithm. A phrase cue can have any variation of similar phrases. These cannot be treated as discrete.

	Phrase Class 5	**Phrase Class 24**	**Phrase Class 30**	**Phrase Class 8**
Similarity with Class 5	1	0.9963925	0.9928540	0.50894997
Phrase1	No.	No, thanks.	Nothing.	Eric?
Phrase2	No!	No, sir.	Nothing's with me.	Marty...
Phrase3	No, to you –M'sieu	No, no.	Never!	Doug...
Phrase4	No, m'sieu!	No thanks.	Nothing comes.	Scooby?
Phrase5	No way.	No, Mr. Charles.	See? Nothing.	Jeff. Jeff.
Phrase6	No...	No... no.	Nothing at all?	Jeff!
Phrase7	...No!	No, sir?	Nowhere.	Gus?
Phrase8	No, Knots!	Sir! No Sir!	Nothing happened.	Jimmy!

Table 3.1: Table illustrating phrase class similarity

If we choose a cut off at 0.99, we are left with a score between 0.99 and 1. We would like updates to be performed in a manner which is scaled proportionately to this. We re-scale the score to $s \in (0, 1]$. The procedure is then as follows:

- We construct the similarity score dictionary in which the key is each phrase class and the values are similar phrases.
- For a holdout iteration, we build the probability profile. Here we accumulate counts for each element. This includes all the values of the similar elements (between zero and one) and thus account not only for the frequency of the element but for the frequency of similar elements. The more similar elements there are, the more heavily weighted the profile for that element becomes.
- In the calculation of the score of the sliding window, we map each element of the sliding window to its corresponding foreground probability profile. A similar procedure of updates is performed in calculating background probabilities in which we consider total probabilities of elements and their similar phrases. To calculate a probability of the position, we take the ratio of foreground to background (or total). In order to describe a score for the window, we sum each of these probabilities.

- We select a placement of a window when the cumulative probability of the window placement is high enough (i.e. higher than a random number with upper bound total probability).
- This window becomes our optimal local window for the holdout sequence which we then assess in terms of the global alignment in order to determine whether it is selected for our set of best motifs.

Both the foreground and background frequencies are computed as weighted counts - the ratio of the foreground to the background then results in a probability.

Since our sequences consist of vectors, we follow the following procedure to calculate the global alignment score. This score computes the alignment of the set of best motifs across all sequences. We calculate a global pattern which is the best representation of all current motifs. To calculate local alignment, a motif is compared to the global motif pattern. The global pattern consists of positions populated by phrase classes with the greatest counts (considering similar elements). Each position is compared across motifs to the global pattern. We transform it as always to scale between zero and one. We construct a score vector which accumulates these alignments per position. Each position has the added similarities and is normalized by dividing by number of motifs. The score vector is then transformed to a single global score by summing individual elements and dividing by w. The score vector and w are highly interpretable. The global pattern is instructive in the understanding of the dataset and the patterns that are found. This structure tells us what motif abstraction looks like. We are also able to cluster our sequences based on how well each motif aligns locally to the global pattern and use this to select the highest aligning local patterns.

To summarize, in this Chapter, we have examined the components making up our conversational motif mining pipeline. This pipeline consists of two high level parts. The first one is the construction of meaningful sequences from text that lend themselves to pattern mining. The second one is the pattern mining algorithm which is an extension of the Gibbs Sampler traditionally used for motif detection. We explain the workings of the algorithm and give a view of how the components work together. The next two

steps in demonstrating the functionality and application of the pipeline is:

1. Understanding the motif mining algorithm and its extensions by generating data with known outcomes and running it through the algorithm.

2. Viewing the results of running the motif mining pipeline on our real world demonstration dataset.

In the next Chapter, we follow these steps and explain our observations.

Chapter 4

Results

The work we have been describing consists of two parts: the first is the embedding of conversations. The second is the application of the motif detection algorithm. Each of the two parts we have mentioned have their own considerations and assumptions. The conversation embedding creation is easier to demonstrate in application. We therefore show this in the next Chapter 5. The purpose of this Chapter is to demonstrate the functionality of the motif detection algorithm. We do so by constructing a series of experiments which use simulated data.

4.1 Demonstration of Motif Detection

In this section, we construct three experiments in order to illustrate the concept of motif detection. In both experiments we input simulated input data in order to examine the behaviour of the algorithm.

4.1.1 One-Dimensional Motif Detection

We use the simplest test case as the first of our experiments. We construct a simple one-dimensional set of sequences and place motifs into known positions. These motifs contain no variations and are all equal lists. The data we set up in this experiment is used as a base for the next experiments. The structure of the data is as shown in

Table 4.1. The blue text represents motif positions. The following points explain the composition of the data:

- 22 sequences are generated.
- Each position in a sequence can take one of 50 elements. A random uniform generator is used in order to populate them.
- The sequences are generated to be of a random length.
- The motifs planted are of length 3.
- We select motif positions and randomly. Not every sequence is assigned a motif.

SequenceID	Generated Sequence										
Sequence1	[28	,36	,44	,**3**	,**5**	,**7**	,42	,7	,10	,7]	
Sequence2	[9	,14	,41	,39	,24	,47	,10	,38	,10	,29]	
Sequence3	[37	,43	,18	,16	,10	,37	,18	,38	,35	,2]	
Sequence4	[17	,4	,46	,35	,**3**	,**5**	,**7**	,33	,18]		
Sequence5	[18	,9	,**3**	,**5**	,**7**	,26	,38	,16	,46]		
Sequence6	[19	,43	,11	,23	,41	,30	,33	,31	,25	,25]	
Sequence7	[33	,5	,2	,19	,25	,12	,37	,14	,24	,10]	
Sequence8	[2	,45	,25	,22	,**3**	,**5**	,**7**	,32	,28]		
Sequence9	[**3**	,**5**	,**7**	,36	,38	,31	,21	,43	,22	,15	,46]
Sequence10	[14	,15	,28	,33	,35	,5	,16	,33	,9	,1]	
Sequence11	[**3**	,**5**	,**7**	,10	,23	,2	,20	,47	,25	,22]	
Sequence12	[23	,27	,48	,48	,2	,48	,12	,15	,37]		
Sequence13	[22	,47	,2	,31	,10	,9	,23	,**3**	,**5**	,**7**]	
Sequence14	[27	,15	,26	,18	,30	,38	,16	,13	,12	,5]	
Sequence15	[6	,14	,29	,1	,21	,13	,14	,24	,39]		
Sequence16	[33	,20	,1	,26	,44	,**3**	,**5**	,**7**	,30	,43]	
Sequence17	[22	,24	,21	,39	,48	,46	,43	,11	,12	,46]	
Sequence18	[1	,41	,28	,21	,10	,7	,34	,28	,14	,18]	
Sequence19	[2	,17	,19	,10	,9	,2	,9	,**3**	,**5**	,**7**]	
Sequence20	[27	,38	,21	,5	,33	,8	,3	,35	,5	,48]	
Sequence21	[24	,20	,33	,17	,15	,3	,37	,40	,31	,34]	
Sequence22	[**3**	,**5**	,**7**	,**3**	,**5**	,**7**	,17	,9	,34	,31]	

Table 4.1: Simple generated sequences

We ran the algorithm on the constructed input set with some obvious parameter choices. The motif length is known so we set $k = 3$ as the window length. Each run of the algorithm consists of 400 iterations. We have 20 randomly initialized runs. The results of the motif detection algorithm applied on the sequences can be seen in Table 4.2.

SequenceID	Generated Sequence										
Sequence1	[28	,36	,44	,**3**	,**5**	,**7**	,42	,7	,10	,7]	
Sequence2	[9	,14	,41	,39	,24	,47	,10	,38	,10	,29]	
Sequence3	[37	,43	,18	,16	,10	,37	,18	,38	,35	,2]	
Sequence4	[17	,4	,46	,35	,**3**	,**5**	,**7**	,33	,18]		
Sequence5	[18	,9	,**3**	,**5**	,**7**	,26	,38	,16	,46]		
Sequence6	[19	,43	,11	,23	,41	,30	,33	,31	,25	,25]	
Sequence7	[33	,5	,2	,19	,25	,12	,37	,14	,24	,10]	
Sequence8	[2	,45	,25	,22	,**3**	,**5**	,**7**	,32	,28]		
Sequence9	[**3**	,**5**	,**7**	,36	,38	,31	,21	,43	,22	,15	,46]
Sequence10	[14	,15	,28	,33	,35	,5	,16	,33	,9	,1]	
Sequence11	[**3**	,**5**	,**7**	,10	,23	,2	,20	,47	,25	,22]	
Sequence12	[23	,27	,48	,48	,2	,48	,12	,15	,37]		
Sequence13	[22	,47	,2	,31	,10	,9	,23	,**3**	,**5**	,**7**]	
Sequence14	[27	,15	,26	,18	,30	,38	,16	,13	,12	,5]	
Sequence15	[6	,14	,29	,1	,21	,13	,14	,24	,39]		
Sequence16	[33	,20	,1	,26	,44	,**3**	,**5**	,**7**	,30	,43]	
Sequence17	[22	,24	,21	,39	,48	,46	,43	,11	,12	,46]	
Sequence18	[1	,41	,28	,21	,10	,7	,34	,28	,14	,18]	
Sequence19	[2	,17	,19	,10	,9	,2	,9	,**3**	,**5**	,**7**]	
Sequence20	[27	,38	,21	,5	,33	,8	,3	,35	,5	,48]	
Sequence21	[24	,20	,33	,17	,15	,3	,37	,40	,31	,34]	
Sequence22	[**3**	,**5**	,**7**	,3	,5	,7	,17	,9	,34	,31]	

Table 4.2: Simple case motif detection results

Correctly identified motifs are represented in blue text. Motifs identified on sequences with no assigned motifs are represented in purple. They arise due to the algorithmic assumption of one motif per sequence. All placed motifs are correctly identified. The purple motifs appear closest to the generic motif on the sequence by cosine similarity. The detection run time is fast. This may be attributed to a small number of sequences, unique elements and window length.

4.1.2 One-Dimensional Motif Detection with Variation

We use the data generated above as the base for this next experiment. In it, the algorithm input motifs contain variations. We observe how well they are identified. Table 4.3 visually displays the input data. Motifs containing variations are coloured pink.

SequenceID	Generated Sequence										
Sequence1	[28	,36	,44	,**3**	,**5**	,**7**	,42	,7	,10	,7]	
Sequence2	[9	,14	,41	,39	,24	,47	,10	,38	,10	,29]	
Sequence3	[37	,43	,18	,16	,10	,37	,18	,38	,35	,2]	
Sequence4	[17	,4	,46	,35	,3	,6	,7	,33	,18]		
Sequence5	[18	,9	,**3**	,**5**	,**7**	,26	,38	,16	,46]		
Sequence6	[19	,43	,11	,23	,41	,30	,33	,31	,25	,25]	
Sequence7	[33	,5	,2	,19	,25	,12	,37	,14	,24	,10]	
Sequence8	[2	,45	,25	,22	,**3**	,**5**	,**7**	,32	,28]		
Sequence9	[**3**	,**5**	,**7**	,36	,38	,31	,21	,43	,22	,15	,46]
Sequence10	[14	,15	,28	,33	,35	,5	,16	,33	,9	,1]	
Sequence11	[**3**	,**5**	,**7**	,10	,23	,2	,20	,47	,25	,22]	
Sequence12	[23	,27	,48	,48	,2	,48	,12	,15	,37]		
Sequence13	[22	,47	,2	,31	,10	,9	,23	,3	,5	,8]	
Sequence14	[27	,15	,26	,18	,30	,38	,16	,13	,12	,5]	
Sequence15	[6	,14	,29	,1	,21	,13	,14	,24	,39]		
Sequence16	[33	,20	,1	,26	,44	,**3**	,**5**	,**7**	,30	,43]	
Sequence17	[22	,24	,21	,39	,48	,46	,43	,11	,12	,46]	
Sequence18	[1	,41	,28	,21	,10	,7	,34	,28	,14	,18]	
Sequence19	[2	,17	,19	,10	,9	,2	,9	,**3**	,**5**	,**7**]	
Sequence20	[27	,38	,21	,5	,33	,8	,3	,35	,5	,48]	
Sequence21	[24	,20	,33	,17	,15	,3	,37	,40	,31	,34]	
Sequence22	[**3**	,**5**	,**7**	,2	,5	,7	,17	,9	,34	,31]	

Table 4.3: Generated one-dimensional sequences with motif variations

Table 4.4 shows the outcome of the motif detection algorithm.

SequenceID	Generated Sequence										
Sequence1	[28	,36	,44	,**3**	,**5**	,**7**	,42	,7	,10	,7]	
Sequence2	[9	,14	,41	,39	,24	,47	,10	,38	,10	,29]	
Sequence3	[37	,43	,18	,16	,10	,37	,18	,38	,35	,2]	
Sequence4	[17	,4	,46	,35	,3	,6	,7	,33	,18]		
Sequence5	[18	,9	,**3**	,**5**	,**7**	,26	,38	,16	,46]		
Sequence6	[19	,43	,11	,23	,41	,30	,33	,31	,25	,25]	
Sequence7	[33	,5	,2	,19	,25	,12	,37	,14	,24	,10]	
Sequence8	[2	,45	,25	,22	,**3**	,**5**	,**7**	,32	,28]		
Sequence9	[**3**	,**5**	,**7**	,36	,38	,31	,21	,43	,22	,15	,46]
Sequence10	[14	,15	,28	,33	,35	,5	,16	,33	,9	,1]	
Sequence11	[**3**	,**5**	,**7**	,10	,23	,2	,20	,47	,25	,22]	
Sequence12	[23	,27	,48	,48	,2	,48	,12	,15	,37]		
Sequence13	[22	,47	,2	,31	,10	,9	,23	,3	,5	,8]	
Sequence14	[27	,15	,26	,18	,30	,38	,16	,13	,12	,5]	
Sequence15	[6	,14	,29	,1	,21	,13	,14	,24	,39]		
Sequence16	[33	,20	,1	,26	,44	,**3**	,**5**	,**7**	,30	,43]	
Sequence17	[22	,24	,21	,39	,48	,46	,43	,11	,12	,46]	
Sequence18	[1	,41	,28	,21	,10	,7	,34	,28	,14	,18]	
Sequence19	[2	,17	,19	,10	,9	,2	,9	,**3**	,**5**	,**7**]	
Sequence20	[27	,38	,21	,5	,33	,8	,3	,35	,5	,48]	
Sequence21	[24	,20	,33	,17	,15	,3	,37	,40	,31	,34]	
Sequence22	[**3**	,**5**	,**7**	,2	,5	,7	,17	,9	,34	,31]	

Table 4.4: Generated one-dimensional sequences with motif variations

In the results Table 4.4, identified instances of the original motif pattern [3,5,7] are coloured blue, instances of the motif with variations are coloured pink and sequences with no assigned motif are coloured purple as before. The results show that the algorithm can select the correct motifs even in the presence of variation. The final sequence is interesting to pay attention to. Here the stronger motif ([3,5,7]) was detected by the algorithm. The motif with the variation ([2,5,7]) was not selected.

4.1.3 Motif Detection on Sequences of Vectors

In the third experiment, we again build on the same sequence dataset generated for the base case. For vector representations of the sequence elements of the motif [3,5,7], we take care not to generate motif elements 3,5 and 7 that are too similar to each other. We do this by generating three vectors which are all under 0.4 cosine similarity from each other. All other vectors are randomly chosen in a two-dimensional space for the purposes of visualization. We use a uniform random generator: $x \in [-1, 1], y \in [0, 1]$.

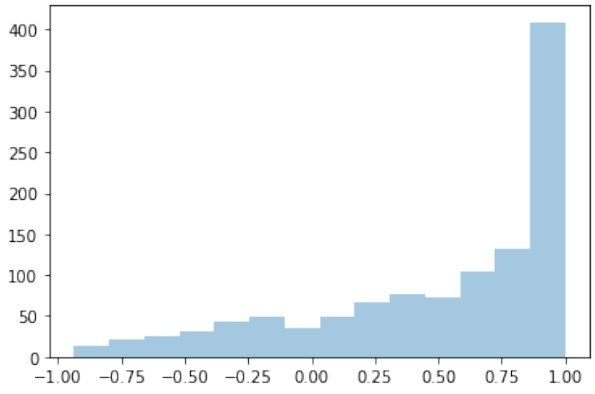

Figure 4.1: Pairwise cosine distances of generated vectors

The distribution of the pairwise cosine distances between all the vectors is shown in Figure 4.1. This distribution guides our choice of threshold for construction of the update dictionary. Elements which are similar to each other are considered in the motif detection algorithm. Phrase classes below a certain similarity are not considered in the updates of the motif detection for the Gibbs sampled sequence. This threshold is guided by the distribution shown. For example, since most phrase classes are very similar to each other (heavily left skewed distribution), choosing a threshold which is too low would mean that a huge portion of the phrase classes get updated at every update.

In Figure 4.2, we show the first 16 generated sequences. The following points describe how to read the graphs:

1. Each subplot is a sequence.

2. Each point is labeled with its position in the sequence. For example, a label of 3

would indicate the third position in the sequence.

3. The colour of the sequence element indicates the categorical element the vector is mapped to. We have 50 possible classes.

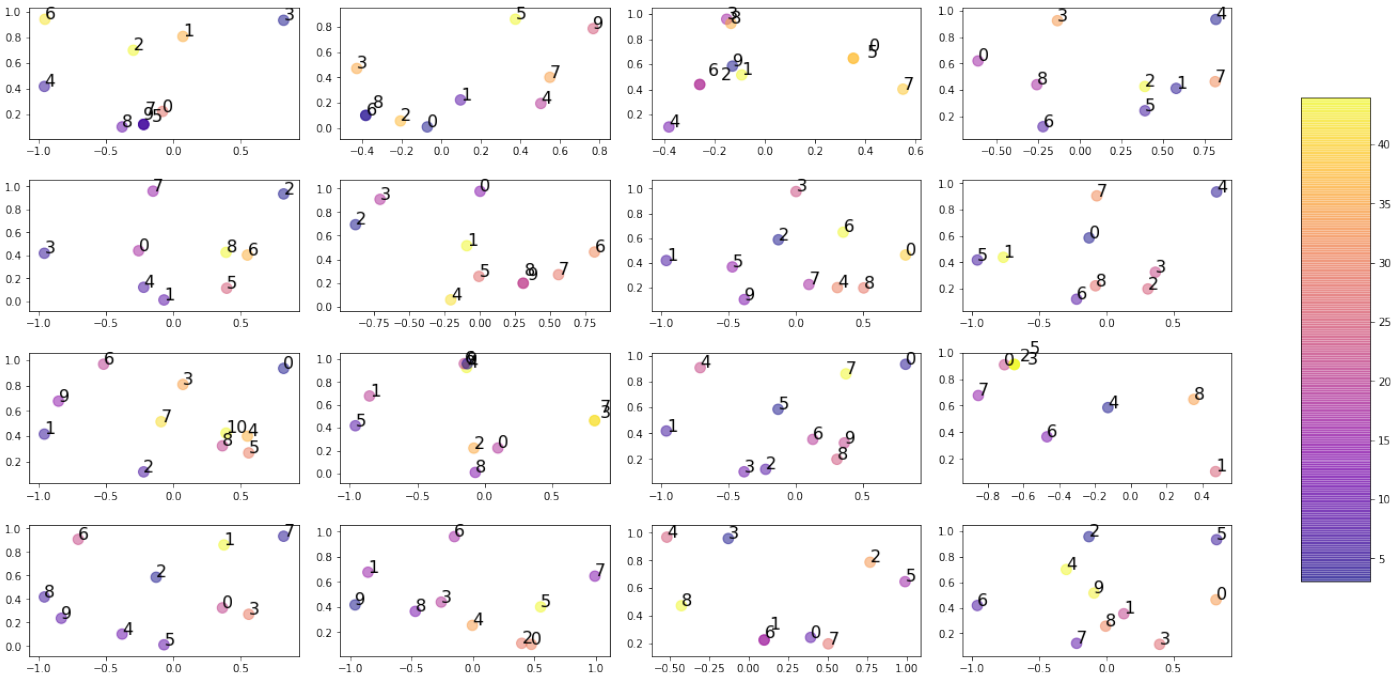

Figure 4.2: Sequences of vector elements dummy input for motif detection algorithm

Once we run the dummy input data as illustrated in Figure 4.2, we get the following output (figure 4.3). The red connecting lines display the detected motifs.

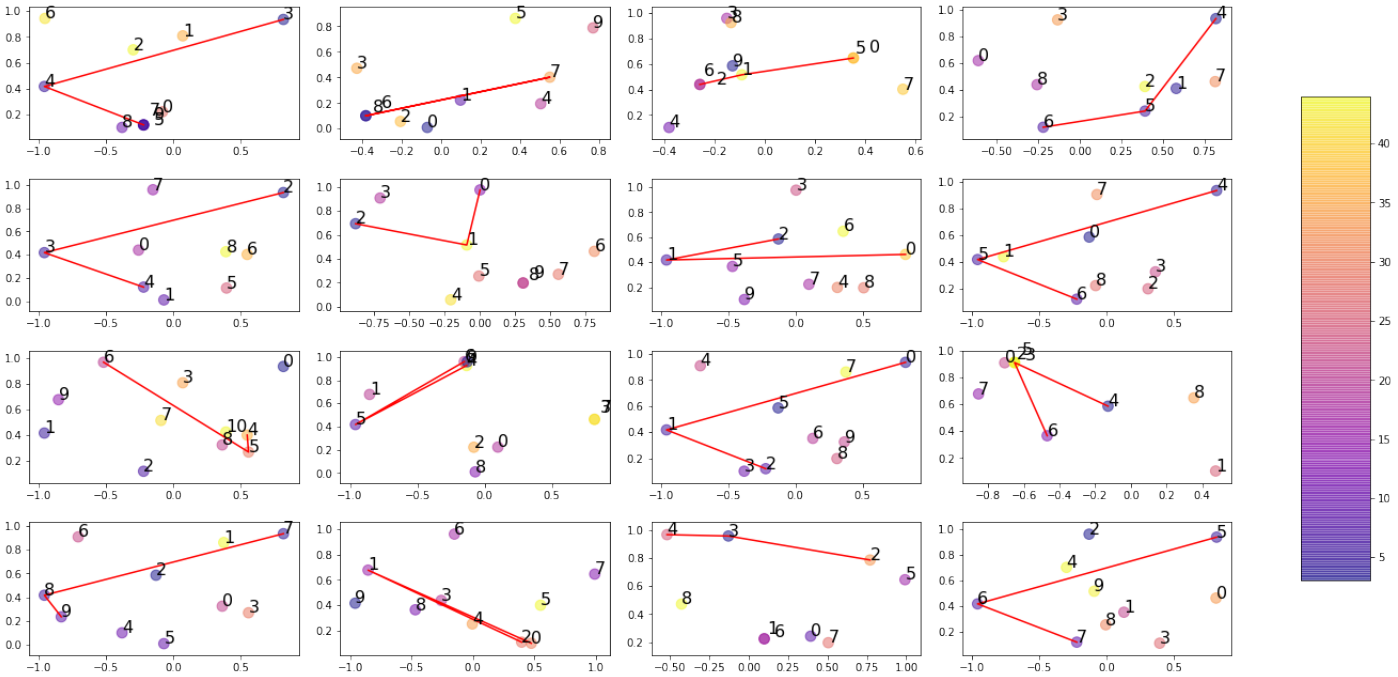

Figure 4.3: Motif detection on vector sequence: results

We detect every instance of assigned motifs except for one positioned at sequence 9. We post a table of the detected motifs. The global pattern score is 0.88993719 and the global pattern is as expected: [3,5,7]. The Table 4.5 tabulates result figures. True motifs with variations, for example sequence 13, have alignment scores in line with expectations. For the positions that are the same as the true motif pattern, the alignment score is 1. For the third position, the alignment is approximately 0.92 - in line with the fact that it is a slight variation. We never achieve an alignment of 1 for any element that is not equal to the true motif pattern element at that position.

SequenceID	Motif			Local Alignment Score		
Sequence1	[3,	5,	7]	[1. ,	1.,	1.]
Sequence2	[10,	38,	10]	[0.27780976,	0.24709723,	0.98573074]
Sequence3	[37,	43,	18]	[0.98789209,	0.77629991,	0.92873099]
Sequence4	[3,	6,	7]	[1. ,	0.21379846,	1.]
Sequence5	[3,	5,	7]	[1. ,	1.,	1.]
Sequence6	[19,	43,	11]	[0.8770001 ,	0.77629991,	0.99301574]
Sequence7	[33,	5,	2]	[0.97217542,	1,	0.82751016]
Sequence8	[3,	5,	7]	[1. ,	1.,	1.]
Sequence9	[38,	31,	21]	[0.98788806,	0.17339304,	0.91828136]
Sequence10	[35,	5,	16]	[0.82474704,	1.,	0.80465436]
Sequence11	[3,	5,	7]	[1. ,	1.,	1.]
Sequence12	[2,	48,	12]	[0.79682382,	0.92919543,	0.99336005]
Sequence13	[3,	5,	8]	[1. ,	1.,	0.98757645]
Sequence14	[27,	15,	26]	[0.90221222,	0.98324186,	0.14273206]
Sequence15	[29,	1,	21]	[0.99925315,	0.75884455,	0.91828136]
Sequence16	[3,	5,	7]	[1. ,	1.,	1.]
Sequence17	[21,	39,	48]	[0.67558232,	0.95545101,	0.9502422]
Sequence18	[21,	10,	7]	[0.67558232,	0.99433143,	1.]
Sequence19	[3,	5,	7]	[1. ,	1.,	1.]
Sequence20	[35,	5,	48]	[0.82474704,	1.,	0.9502422]
Sequence21	[33,	17,	15]	[0.97217542,	0.96411321,	0.99271795]
Sequence22	[3,	5,	7]	[0.79682382,	1.,	1.]

Table 4.5: Motifs detected in multi-dimensional sequences

The next few graphs supplement a visual analysis we perform on the experiment 3 results. Since the underlying elements contain multiple dimensions, it is more intuitive to read relationships off of graphs.

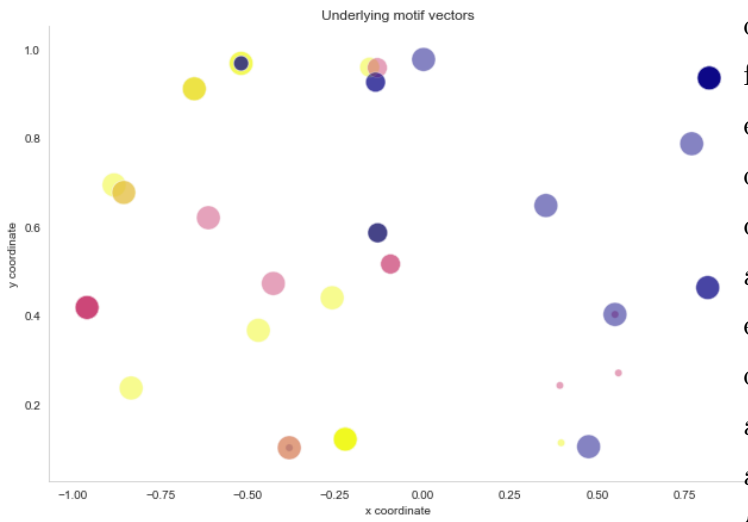

Figure 4.4: Motif detection on vector sequence

We begin to consider the similarity of the sequence elements identified for different motif positions. For example, Figure 4.4 shows a view of the three motif positions in different colours. The blue points are representative of all sequence elements identified in position 1 of the motifs. The purple points are the plotted coordinates of all sequence elements mapping to the second position and the yellow points are those of the third position. We observe that the coordinate space segments according to motif position. This is a result that indicates the motif detection is successfully detecting subsequences with clear alignment among the elements. The saturation of colour indicates density of points. The darkest blue, purple and yellow points represent the motif pattern [3,5,7]. lighter points do not have as many underlying points. Smaller points are further away from other points of that position. These further highlight that there are clear groups of points corresponding to motif position.

Next, we consider how well individual sequence motifs align with the true global motif pattern [3,5,7]. This is best illustrated through the Figure 4.5 showing the distribution of alignments across positions per sequence.

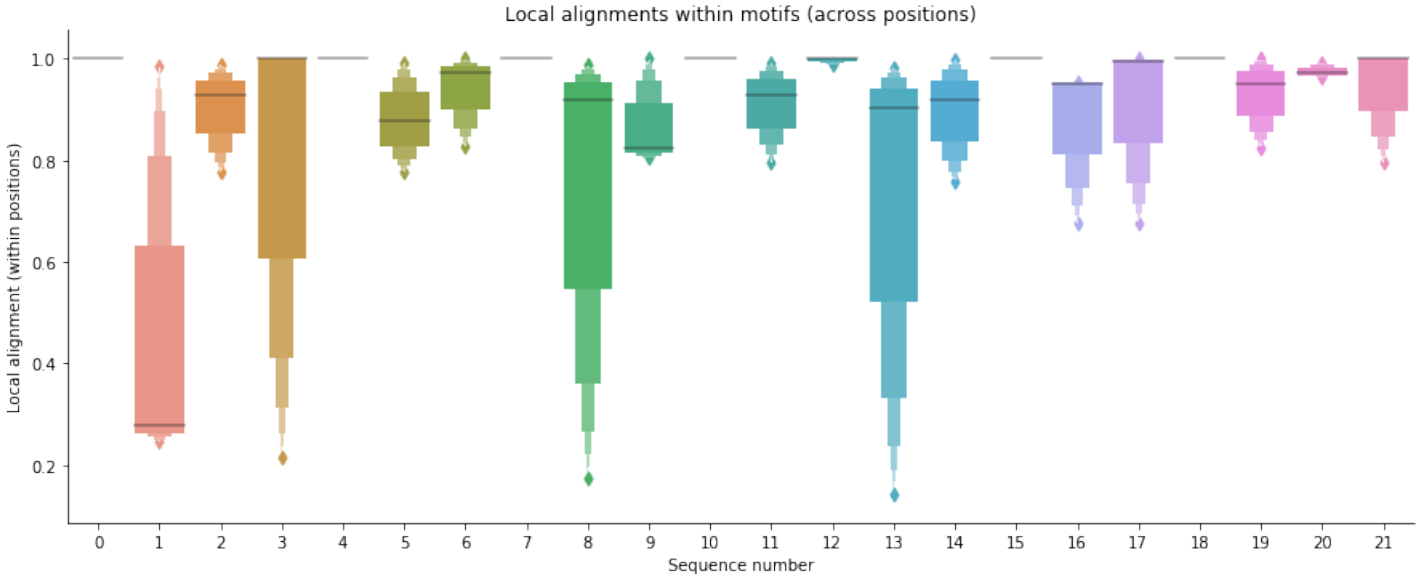

Figure 4.5: Motifs detected across sequences

Most sequences in Figure 4.5 show a very tight distribution in positional alignment scores. All sequences with the exact motif detected are perfectly uniform at 1. This includes: sequences 0,4,7,10,15,18. Non exact motifs detected where the true motif contains variations can be seen at sequences which are very closely distributed close to one. Sequence which have a wide range of alignment scores or have lower scores in general have lower medians and are immediately visible as the sequences with long boxes. The fact that sequence 21 ([33,17,15]) has very high alignment scores shows the functionality of the motif detection extension. The algorithm is able to detect similarity in underlying vectors of sequences. A clear further extension would be to specifically retain sequences with a very tight distribution and discard sequences with flat/long distributions. These sequences appear not to be aligned to the global motif. This would firstly remove noise and secondly would allow us to identify sequences where the patterns are different to those being searched for in an unsupervised way.

Chapter 5

Pipeline Application

In this Chapter we investigate the use of the pipeline we have set up in Chapter 3 and apply it to the Cornell Movie–Dialogs Corpus [14]. This is a set of conversational movie scripts extracted from 617 movies, encompassing a diverse set of conversations. The language in movie scripts is clean and therefore easier to process. This allows us to focus on the pipeline concepts instead of text wrangling. The Chapter addresses the application of the pipeline specifically to extract patterns in conversations using motif mining. We consider whether or not we can leverage this technique for something like scene detection in movie scripts.

This Chapter follows the application of each component of the pipeline on the dataset. We consider sequence creation and the application of motif detection on to these sequences. The motif mining analysis we perform consists of two experiments. The first experiment attempts to treat movie quarters separately with a specific focus on the second quarter of comedy movie scripts. This is because it is assumed that the first quarter of a film contains scene setting and initial character development. The second analysis then performs motif detection on the entire length of the movie scripts. We begin the Chapter by giving a brief overview of the dataset in order to understand what we are working with in section 5.1.

5.1 Dataset

We make use of the dataset developed by the authors in [14] and retrieve all movies with comedy in the genre tags. This results in 159 movies and 93248 unique utterances. We consider the distribution of phrase lengths in Figure 5.1. This assists with data understanding and preprocessing.

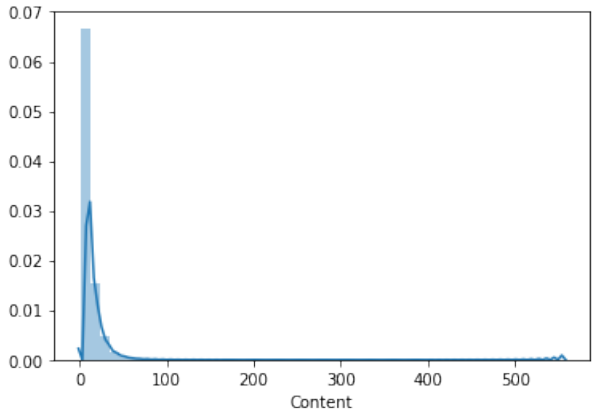

Figure 5.1: Distribution of phrase lengths

The length of the utterances is typically less than 22 words (which describes the 90^{th} percentile of this distribution). The short phrase lengths highlight a need to preserve stop words. Short text is typically more difficult to model with traditional methods such as topic modeling due to the sparsity arising from lack of word co-occurrences [10].

As mentioned, we partition each movie into quartiles. This is because we assume there are common patterns of plot, character and scene development in different movie genres.

5.2 Preprocessing

Preprocessing is typically not described explicitly. We choose to address this because the shaping and treatment of data is crucial to consider in answering different questions. More specifically, depending on the sequences we produce, different motifs will be found, and different questions will be answered.

One of the constraints of the motif detection algorithm is that of finding one motif per sequence. This instructs how we shape our sequences. For a conversational dataset, there

is typically either one long conversation or many different sequences of conversations. The nature of the application aided us in understanding how to break up our conversational sequences. In our case, each sequence is a movie script and we consider whether every movie may contain an instance of a motif. We begin by considering that a motif may describe half an hour of a movie each quartile of the movie script forms a sequence.

Since the text we are dealing with has been authored by screen writers, the language is clean and has minimal use of strange characters, mixed languages and colloquialisms. This enables us to use pre-trained word or character embeddings which have been trained on large corpora of information such as FastText embeddings trained on common crawl which we explain in Chapter 3.

We employ the standard practice of lower-casing the data and removing punctuation. We can remove punctuation for the following reasons:

1. Any unexpected characters are dealt with.
2. Movie scripts do not place as much emphasis on characters to convey meaning unlike social media where a lot of meaning is carried in symbols, emojis and the like.
3. In social media, often symbols replace words while in more formal writing, punctuation supplements words. Therefore, the inherent meaning is contained in the actual words and is not removed with punctuation.

In Table 5.1, we can see the importance of preserving stop words. Entire conversational queues are removed by removing stop words. The entailment of the phrases: “I see” (an affirmative phrase), “where is he” (an inquiry) and “go” (a command) are different. They serve particular and distinct functions in a conversation. Stop words create a lot of context which is lost if we remove them. The quality of text and embeddings that we use means we do not need to cater for misspelling, different variations of the same word etc. In fact, if we try to apply lexical normalization, we propagate error throughout the embeddings which is undesirable.

Raw	No Punctuation	No Punctuation or Stop Words
Nowhere.	nowhere	NaN
What?	what	NaN
Where is he?	where is he	NaN
I see.	i see	NaN
...now!	now	NaN
There.	there	NaN
Will do.	will do	NaN
So?	do	NaN
Why not???	why not	NaN
It is?	it is	NaN
Is he there?	is he there	NaN
Go, go, go!	go go go	NaN
What are you doing here?	what are you doing here	NaN

Table 5.1: Illustration of loss of context information through stop word removal

5.3 Sequence Creation

5.3.1 Text Embedding

The author of the word embeddings evaluation survey paper [3] explains that there are two main methods of evaluations for word embeddings. The first is extrinsic evaluation, where the quality of word embeddings is judged on how well they allow text to be used for downstream supervised learning tasks. The second is intrinsic evaluation where importance is placed on comparing human judgement with relationships between words. Our work is based on [34] and [13] where numerous experiments are performed to show the quality of embeddings. We do not construct experiments or comparisons of word embeddings since we trust the benchmark results obtained by the authors. We evaluate the quality of the embeddings by looking at downstream results.

After we have found word embeddings, we find sentence embeddings for our utterances.

With these phrase level embeddings, we create a graph. This is done by finding the 10 nearest neighbours to each phrase using angular distance. These distances form edges on the graph between the phrases which form the nodes. The greater the angular distance between two phrases, the smaller their edge weight should be so the derivation of the edge weight is as follows: 1-minmax(angular_dist). We use the community walktrap algorithm to detect prominent communities of phrases. Figure 5.2 shows an interesting view of what the phrase embedding graph looks like.

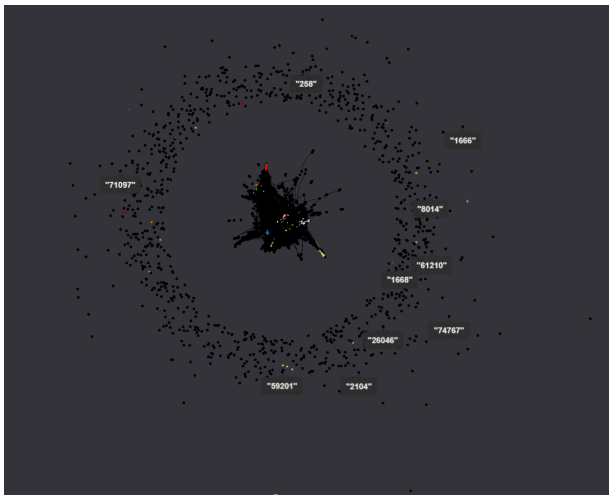

Figure 5.2: High level view of network structure

In figure 5.2, we observe many highly interconnected phrases which appear as the dense and darkly coloured middle piece of the network. The phrases that are not so highly connected appear on the outer edges of the graph. These are phrases that are not similar to many other phrases. This may be attributed to their unique content expressing ideas not common to usual conversational flow. These are difficult to fit into a phrase class. Even though they may give insight into what makes one movie different from others, we treat these as outliers. Figure 5.3 shows a closer view of the densely connected phrases.

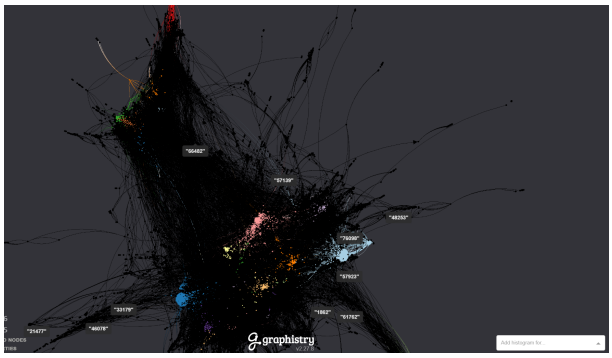

Figure 5.3: Closer view of closely connected phrases

The colourful nodes represent the communities that have been detected. The algorithm detects 3232 communities. Of these, communities with less than five members in them are not considered in our analysis. These are the outliers which are not associated with generic phrase classes. We obtain 1442 communities.

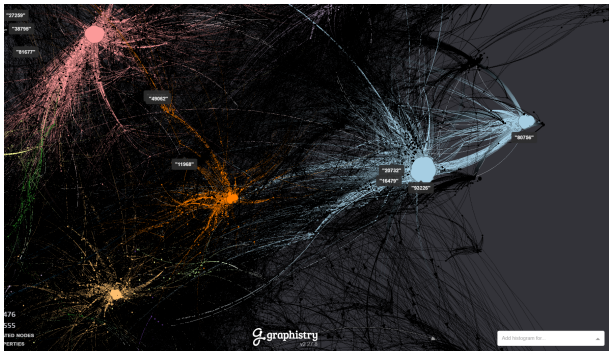

Figure 5.4: Closer view of closely connected phrases

Figure 5.4 shows the central connected phrases in the network. Each community is representative of a group of phrases and this leads to the formation of abstract phrase classes. These phrase classes can be quantified by the centroid of the community phrase positions in the space. This moves us away from dealing with categorical phrase class elements. We are now dealing with phrase classes represented in our phrase embedding space.

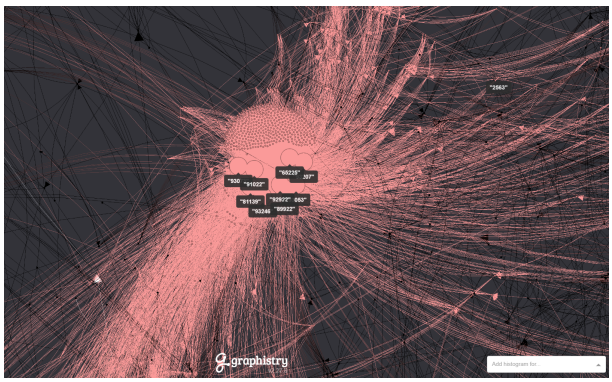

Figure 5.5: View of a detected phrase community

Figure 5.5 provides a closer view of a single phrase community. This forms a phrase class. The reason phrase classes are better represented by centroids than by a categorical class can be illustrated with an example. Communities with greeting phrases should be closer to each other than to clusters expressing a conversationally, semantically and emotionally different phrase type (for example frustration). Since we focus on keeping the relationships of the phrase classes while finding with a lower dimensionality, we use the UMAP algorithm. We optimize the correlation between the distances of the centroids in the original space and in the reduced space. We find six

dimensions after fine tuning to give the best results before the addition of dimensions corresponds with insignificant gains in correlation.

5.4 Motif Detection

In this section we leverage the low dimensional phrase class sequences as inputs into the motif detection algorithm. The sequences are composed of comedy movie scripts split up into quarters. Our intention was to perform a comparative analysis of these quarters. The assumption was that we might pick out scene-setting motifs in quarter one as opposed to actual comedy tropes in other quarters. We discover that this idea was too fine grained for the initial analysis as we explain in this section. The final analysis focuses on the types of patterns we are able to extract from the whole dataset.

5.4.1 Quarter Two comedy movie analysis

It is assumed that in the first quarter of the movie script there is scene setting and character development. We focus on the second quarter of the movie scripts with the intention of extracting motifs related to comedy tropes. This yields 110 sequences of varied lengths (we have extended the motif detection algorithm to cater for different lengths Chapter 3). These sequences contain logical conversational gaps due to the phrases that do not map to large enough communities.

We ran the motif detection algorithm with a motif length of 5 over 400 iterations. The global score (alignment of all motifs) is 0.85399728 and the extracted global pattern contained the following elements: [0,57,0,37,0]. Due to the large number of sequences present, we observe the variation in local alignments visually.

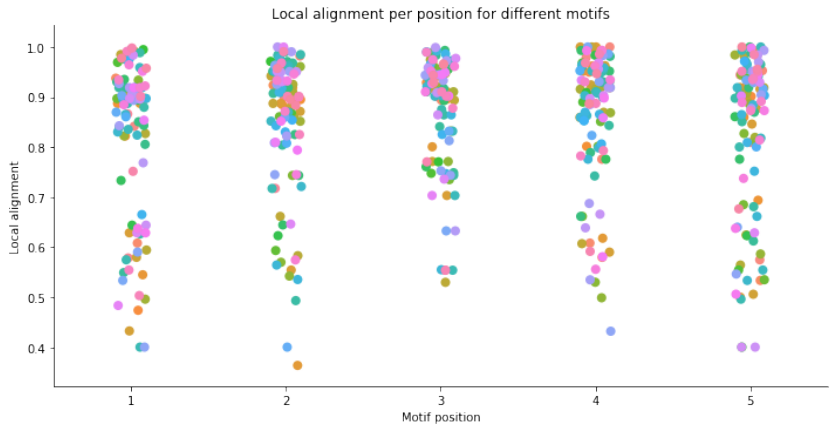

Figure 5.6: Distribution of alignments for different positions across sequences

The graph in Figure 5.6 shows each sequence as a point. The horizontal axis describes the motif position while the values on the vertical axis describe the alignment of the sequence at that position to the global pattern. The distribution of how well sequences align with the global pattern is spread out for each position. Interestingly, this distribution does not become tighter after increasing the number of iterations the algorithm performs. We expand on this in the next analysis.

The second result plot in Figure 5.7 shows a view across sequences as opposed to positions. We represent the alignment of every motif with the global motif as a boxplot. This will show the median alignment of the sequence motif. In a sequence, if most positions of the motif are well aligned, the box plot will be tight. On the other hand, if there are some positions with very poor alignment scores and other positions with good alignment scores, the boxplot will have a large range. The graph is very difficult to read even though it is easy to interpret. Instead, we use a different technique to summarize the difference of ranges in each sequence. For each motif we calculate the range of alignment scores and plot this distribution. This is shown in Figure 5.8

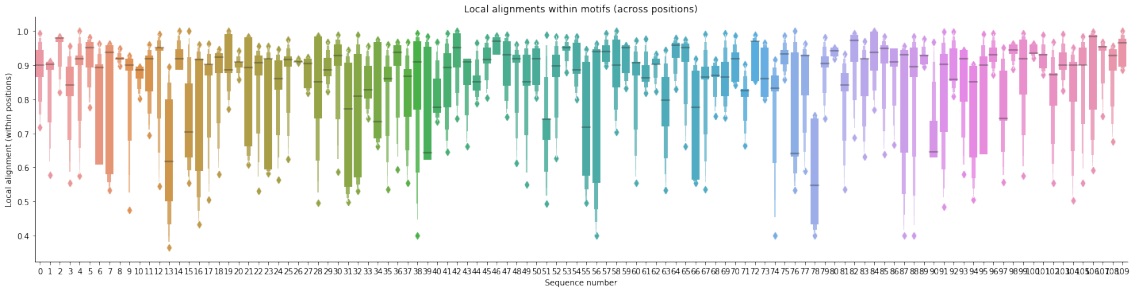

Figure 5.7: Distribution of alignments for different positions across sequences

Primarily, the two graphs shown in Figure 5.7 and Figure 5.8 illustrate the wide range of alignments present in each motif across the positions.

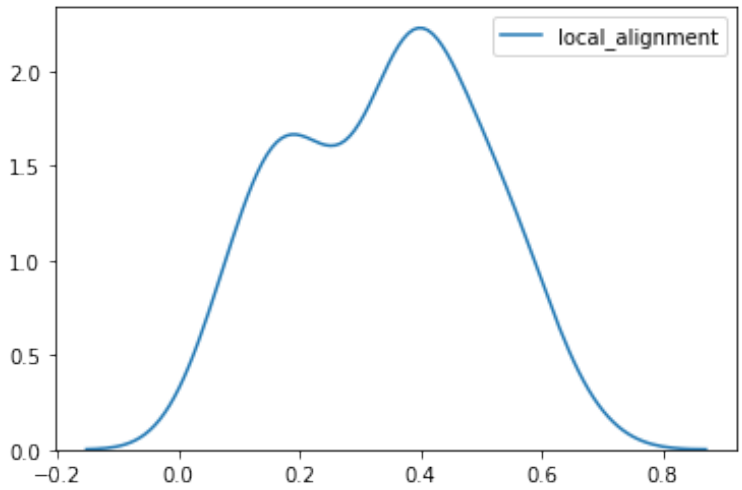

Figure 5.8: Distribution of alignments for different positions across sequences

For each position in the motif, we have calculated the most representative element across all motifs. The motif made up in this way we term the global motif. Each of the phrase classes in the positions of the global pattern can be more easily understood by looking at the phrases which make up a phrase class. Table 5.2 shows these underlying classes.

0	**57**	**0**	**37**	**0**
'Yeah, him.'	'Me and John'	'Yeah, him.'	'Why?'	'Yeah, him.'
'Yeah.'	'Me!'	'Yeah.'	'Why certainly'	'Yeah.'
'Yes.'	'Me, me, me!'	'Yes.'	'Sure. Why?'	'Yes.'
'Uh... Yes.'	'Me either,'	'Uh... Yes.'	'...Why?'	'Uh... Yes.'
'Yes.'	'For me?'	'Yes.'		'Yes.'
'Yes?'	'Ay, me.'	'Yes?'		'Yes?'
'Yes, ma'am.'	'With me?'	'Yes, ma'am.'		'Yes, ma'am.'

Table 5.2: Representation of global pattern through phrase classes

The global pattern is very generic. It is made up of phrase classes which highly align with almost any other phrase class due to the nature of the English language and the structure of the embeddings. These phrase classes also form the largest communities comprising the largest number of underlying phrases. We can see that it is too generic to convey specific information about the movie conversations the algorithm was run on. On physical inspection, the motifs found align on meaning across positions as shown in Table 5.3.

Movie ID	**Motif Classes**	**Motif Phrases**
m532	[21,245,328,103,37]	['Hi!', 'What's the...', 'Where have you been...?', 'I can't', 'Why?']
m4	[36,21,83,248,37]	['Hello.', 'Hi, it's me...', 'F* you', 'That was quick.', 'Why?']
m102	[224,242,639,164,3084]	['Who's that?', 'Jesus.', 'Is this a lover?', 'Please!', 'What is it Miles?']
m327	[768,4,6,986,419]	['Who's Principal Collins?', 'The principal.','Wow.', 'Oh, is this your special place?', 'H A R R Y...']

Table 5.3: Table showing detected motifs in quarter 2

These phrases are difficult to interpret as abstractions of pieces of conversations due to

the conversational gaps present. The conversational cues beginning to form are apparent - for example: ['Greeting', 'Implication', 'Statement', 'Statement', 'Why'] and ['Who', 'Person Entity', 'Statement', 'Question', 'Person Reference']. However, these do not appear in enough sequences to form an abstraction as opposed to a commonality between a handful of sequences. In the next section, we run the motif detection algorithm on the full movie scripts. This will mitigate the sparsity effect and provide longer sequences to work with.

5.4.2 Full Script Comedy Movie Analysis

In this section we describe the results of the comedy movie script analysis and pay more attention to the alignments and motifs found. We structure the input sequences differently. Firstly, we take the full length of the movie script without splitting it into quarters. Secondly, we have acknowledged the fact that our results are worsened by existing gaps in the conversational mappings. This is due to phrases that span multiple sentences or phrases that do not match any others. Due to time constraints at the time of writing, we do not redo the parsing of the script data and rather perform our analysis with mitigating steps in mind.

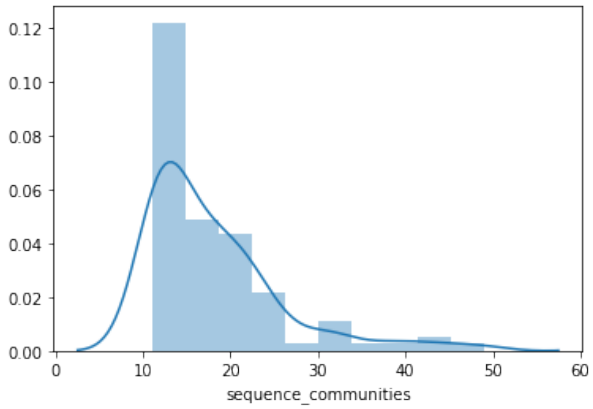

Figure 5.9: Distribution of sequence lengths

This entails finding the longest subsequence in each sequence with no more than two consecutive missing phrases. Taking any less than two consecutive gaps produces sequences which are too short to be useful for the algorithm. In order to keep a meaningful sequence length, we choose to work with sequences of greater than length 10. We work with 97 sequences. The distribution of their lengths is right skewed as shown in Figure 5.9.

We run the algorithm with similar parameters: a window size of 5, over 400 iterations and with 841 phrase classes or sequence elements. The final score and global pattern extracted include are 0.84433245 and [0, 0, 23, 0, 0] respectively. Figure 5.10 illustrates the positional variation of local alignments.

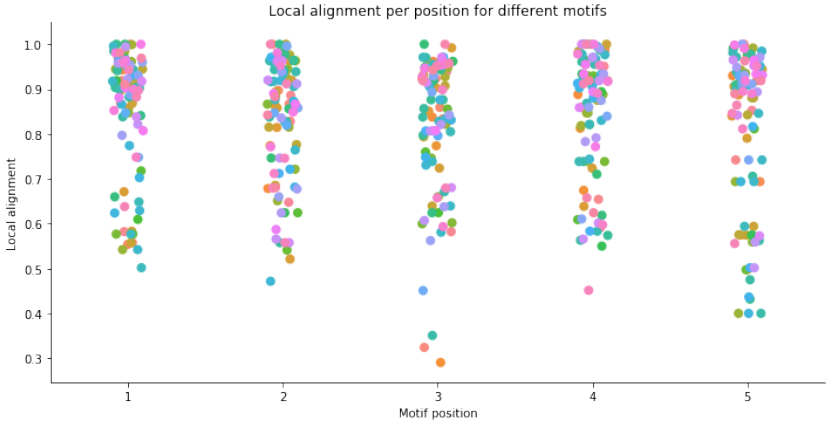

Figure 5.10: Variation of alignment across motif positions for sequences

The graph in Figure 5.10 shows an improvement in the alignment of different sequences across motif positions as compared with Figure 5.6. This is true for all positions except for the middle position which is phrase class 23. This class is very specific and only contains five phrases. It is therefore expected that while the algorithm is finding more specific movie script motifs, there will be a lot of variation where sequences are not aligning to this motif. The five phrases contained in the phrase class are as follows:

- "George! What are you doing here?"
- "Why're you doing this?"
- "Why are you doing that?"
- "Why are you doing this?"
- "What are you doing here?"

We consider the 'uncertainty' - or range of alignment scores across motifs. There is an improvement over the quarter two experiments with more sequences have a smaller range.

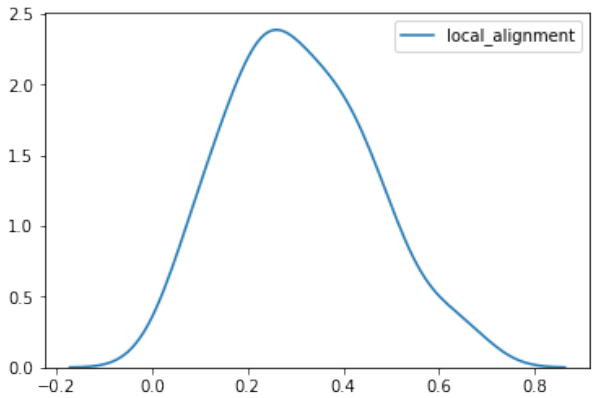

Figure 5.11: Range of alignments across motifs

These sequences contain the motif that is picked out. There are sequences that do not align as well. Since we can see which ones these are we have the flexibility of removing them. For ease of readability, we do not include the box-plot across sequences that shows this. The kernel density plot of the ranges is shown in Figure 5.11. The peak occurs just after 0.2, whereas the peak in quarter 2 analysis Figure 5.8 lies close to 0.6.

We find a very strong motif by extracting the exchanges containing variations of the very specific 23rd class. We post the results below and explain this. There appears to be a human emotional undertone that is extracted in each of these passages which is what the motif detection picked up on.

Movie ID	Motif Classes	Motif Phrases
m35	[401,7,23,461,126]	[‘Take it. Damn it!’, ‘Okay’, ‘Why are you doing that?’, ‘I haven’t brushed yet.’, ‘Wait, Eve, Please! Wait’]
m331	[383,0,23,1,2369]	[‘So Tracy?’, ‘Yes?’, ‘Why are you doing this?’, ‘Doing what?’, ‘[...] we might have been, well, friends.’]
m458	[110,700,23,280,4]	[‘Better than I...’, ‘That’s the spirit!’, ‘George! What are you doing here?’, ‘What is this?’, ‘Wings?’]
m140	[25,152,335,382,0]	[‘I say we don’t want to appear greedy.’, ‘Oh. That.‘ ,‘What do you think you’re doing?’, ‘Why, I’m assisting you, sir.’, ‘Yes, sir.’]
m54	[194,1079,151,1079,22]	[‘But you LOVED her all the same...’, ‘We never went to bed together.’, ‘Why do you go on about that? What does it matter?’, ‘You’ve been to bed with somebody else, haven’t you?’, ‘I’ve never LOVED anybody!’]
m380	[126,654,1170,173,19]	[‘So how are you?’, ‘I’m okay’, ‘You didn’t answer my question. What are you doing here?’, ‘Oh, nothing much. You know...’,‘Well...’]

Table 5.4: Table showing detected motifs in the full movie scripts

"The motive is the smallest structural unit possessing thematic identity" [46]. We see this expressing itself here in our findings. We pick up a very particular conversational queue. This is the theme of exasperation that we pick up on. "Why are you doing this?", "What are you doing?" carries a large emotional weight in the motif. This is surrounded by generic classes which can be thought of as wildcards since they can align with most other classes. In this instance this is class 0. Our motif window is not long enough to capture the whole complexity of the conversation; if we were to extend the window, we might begin to pick up on more queues. There is a second scenario where this motif holds; in syntax, grammar and words it is similar; however, in nuance it is not. This is the case of the straight forward factual question: "Why are you doing that"? in context of brushing, for example in "m35". Algorithmically, changing the initialization will pick out a different motif with a different conversational cue and "essence".

We perform one more demonstration. We run the algorithm with a different random initialization on the same data and tuning. We find encouraging results. The global pattern: [100, 0, 66, 1, 185] has an alignment score of 0.84346919. The underlying classes are interesting and fit a comedy type of script. This looks as follows:

phrase class	**example phrases**	**cue**
100	'Huh?'	confusion
0	'Yeah', 'Yes, 'Oh','Yep', 'yes?','yes!'	wildcard class
66	'Tell me', 'tell me again', 'that's what you tell me'	statement or further prompt for information
1	'What?','Doing what?', 'What happened?', 'The reason?', 'what!'	disbelief exclamation or question for further information
185	'You okay?'	concern or display of consideration

Table 5.5: Global Motif and associated cues

Table 5.6 contains a sample of the motifs detected in the second experiment of the analysis. There are slight deviations in individual motifs found across the sequences. The overall tone and message remain quite clear. The motif detected is of a conversation between characters - some inquiry and an explanation. Individual motifs are expressions of a higher representation of a conversational cue. Since it is typically difficult to separate conversations from emotions, we detect variations on an emotional theme. We observe that the detected motif depends on the chosen initialization. The same set of sequences may have many different motifs present in them. In order to start thinking about applications such as scene detection using this set of techniques, we would have to understand the effect of initialization slightly better. We may be able to start introducing concepts such as scene detection by introducing an example of the desired scene into the set of global motifs. This could be done by increasing the probability weighting for the desired motif in order for the Gibbs sampling search to find similar patterns in other sequences.

Movie ID	Motif Classes	Motif Phrases
m140	[100,2,5,152,335,3]	[‘Huh?’, ‘I say we don’t want to appear’, ‘Oh. That.’, ‘What do you think you’re doing?’, ‘Why, I’m assisting you, sir.’]
m82	[1312,1,2419,94]	[‘No, no–s*t no. Excuse me, Penelope’, ‘For what?’,‘For saying “s*t” Or is that now okay’, ‘Who’s dead?’,‘What happened?’]
m152	[820,555,28,9,20,6]	[‘Are you kidding me?’, ‘Seems like a nice place.’, ‘It is, if you like idiots...’, ‘What do you mean?’,‘Really?’]
m178	[157,1,1818,1,98]	[‘What’s wrong?’,‘What the f*!’, ‘You said it looked like a car accident’, ‘What the f*!’,‘I’ll pay for it.’]
m59	[44,745,234,223,631]	[‘Do you know what you want?’, ‘Excellent.’, ‘Are you alright?’, ‘Oh yeah.’, ‘Do you mind if I excuse myself for a moment?’]
m516	[1,117,366,1,1203]	[‘The reason?’, ‘I see.’, ‘Sit down.’, ‘What?’, ‘Miss Can Cartier.’]
m235	[342,1088,66,340]	[‘I would have said so.’, ‘Aye, but you know him well.’, ‘That’s what you tell me.’, ‘What have you heard?’, ‘Is it? Is it really?’]
m380	[220,220,700,125,654]	[‘Well, I hope you’ve changed.’, ‘Yeah, and for yours. I’m sure you’ve changed’, ‘...namely a personality.’, ‘So how are you?’, ‘I’m okay.’]

Table 5.6: Excerpt of detected motifs with varied random initialization

5.4.3 Results Summary

We began this analysis with a few questions. Based on what we have observed, we can say that we are able to successfully extract patterns in conversations using motif mining. We can detect conversational build up not just in sequence patterns but in meaning and tone. This ability can be used as a base for further applications such as scene detection based on script text.

In this Chapter, we address the two high level components of the conversational motif mining pipeline. In order to demonstrate how the motif mining algorithm works, we perform multiple simple demonstrations using dummy generated data for which we know the results. We see that in the one-dimensional case with 50 elements, our algorithm discovers all of the planted motifs. In the case where we generate sequences with motifs containing variations, the algorithm selects all of the motifs and manages to distinguish the most correct motif in a sequence with two close motifs. In the case where we generate data which mimics our conversational classes, we see that the algorithm detects every instance of a planted motif except one. We have visual demonstrations of how the motifs look in a higher-dimensional instance.

We then examine the application of the entire conversational motif mining pipeline on the movie script data. Without placing emphasis, we describe the considerations necessary to take for application-specific sequence preprocessing. We find that we are able to construct generic phrase classes that form the basis elements of our sequences. During this process, we consider high-dimensional phrase embedding representations of the communities found using community detection. We simplify this representation by using non-linear dimensionality reduction preserving the relationships between the phrase classes. We ignore phrases that are in communities that are too small and thereby reduce the number of classes we are dealing with. We see that we lose information in doing this and view this as an important step to address in future work.

We perform two analyses. The first is a more specific analysis in which we extract the second quarter of comedy movies and run it through the pipeline. There are promising

results; however, there appears to be information loss due to the reason mentioned above. We perform an analysis on the full script of comedy movies using the longest subsequences with no more than two consecutive missing phrases. We observe that the motif alignments are closer in this analysis. Motifs appear to vary depending on the chosen initialization. We are able to understand how to analyze the global motif pattern. We observe very generic phrase types that do not capture very specific meaning to be in multiple positions of the global motif, and very emotively specific motifs in some other positions. This forms the emotional tone of the extracted motifs.

The multitude of questions we can ask of an open source dataset demonstrate the diversity of ways in which the pipeline can be applied. In a business setting, much more directed questions can be addressed with this approach than in a cross-sectional manner. For example, in financial institutions using free-form chat platforms, it is difficult to parse portions of conversations relating to pricing. This ability would give these businesses ability to perform a whole suite of analytics on top of such a pricing feature. The simplest example of this being win loss ratios and the like. All of these applications begin with being able to detect motifs in sequences which we have managed to do. In the following section, we explain our conclusions as well as improvements and future work that we could consider doing from a technical perspective.

Chapter 6

Conclusion

We set out to answer a two-part research question in our work. Firstly, we wanted to know whether it is possible to characterize a conversation as a meaningful sequence. Secondly, we wanted to test the feasibility of applying the Bioinformatics tool of motif mining to a set of conversational sequences. In this Chapter we address these questions successively.

6.1 Sequences

We conclude that it is possible to characterize a conversation as a meaningful sequence. These sequences can be used as an input for motif detection or potentially other downstream models such as clustering. In our work, we are able to leverage pre-trained embeddings because the text data used to demonstrate our methodology has clean and standard language. These embeddings are advantageous due to their high quality; however, the price for this is high dimensionality. Performing motif detection directly on top of these phrase embeddings becomes computationally infeasible on a standard machine. In addition, using the most granular phrase embeddings does not allow for the discovery of the abstract classes which we capture in "phrase types". These are beneficial as they capture generic information about the types of phrases. In order to construct the phrase classes, we group phrases together using a graph structure. This is done in a way that avoids clustering. Clustering is avoided due to the high cardinality and dimensionality of

the phrase embeddings. It also avoids the need to make assumptions about the space our data lies in as a standard k-means typically would. Community detection on top of the phrase graph yields the phrase classes. Each phrase class is represented by calculating the centroid of the phrase embeddings in the community. We apply manifold learning on top of the phrase classes and reduce the dimensionality while conserving the relationships between the centroids.

The aim of our work is to explore new methods expose new possibilities. As such the results we derive are preliminary. These look promising; however, further work is needed in order to enhance and benchmark them. An example of this is the addition of a pre-processing step in which longer phrases with multiple sentences could be split up into multiple phrases. This would make it easier to capture a single idea per phrase and map this phrase to a phrase class. This may reduce some ambiguity around longer phrases with many ideas. Another enhancement worth exploring is finding a more structured approach to community detection. The number of communities found in the data number in the thousands. There are also many communities that are very small and thus discarded. All phrases mapping to these communities are dropped. This leaves 'holes' in our sequences. Addressing this will likely improve the results of the motif detection. This can already be seen in the application Chapter with two different analyses performed. The first analysis considers only the second quarter of movie films. Here, we permit any number of consecutive missing phrases within the sequence. The second analysis we perform over the entire film. We select the longest subsequence with no more than two missing consecutive phrases. This performs better. It is observed that the results are more coherent in meaning when looking at the detected global pattern and local alignments.

The advantages of having constructed the phrase classes are as follows:

1. We can represent the abstract meaning of a conversational phrase.
2. We can express these representations in a continuous space as opposed to categorical structure.

3. We preserve the nature of the class similarities. i.e. phrase classes that are more similar are represented closer together.

We have shown that it is possible to create low dimensional, meaningful conversational sequences. These are interpretable because they are made up of abstract representations - we have sequences of phrase classes.

6.2 Motif Detection

For the second part of the research question, we conclude that it is possible to run genetics-inspired motif detection on conversational sequences in order to extract meaningful abstract patterns. We can extract a representative subsequence best describing a group of sequences and interpret them by their phrase classes. These we call global patterns. The specific instances of these subsequences have a local alignment. We can demonstrate these concepts using the comedy genre in film scripts. We observe that the global motif calculated from our data contains emotional information. This pertains to conversational turns that characters may have in movie scripts.

As we have mentioned, our work aims to focus on exploration of the collection of methods. As such, the application of motif detection on real data was used as a demonstrative tool. It is possible to perform a deeper analysis on comedy movies and extract more insights using conversational motif mining. Future directions of this could include:

1. Performing an analysis on different quarters of the comedy films. This would highlight the differences in generic patterns in the introductory scene setting, the plot and character development, the dramatic crisis and the happy or unhappy ending.

2. Clustering entire movies based on motifs detected within them. This might result in a more precise grouping.

3. Indexing movies based on motifs found within them. This can be used for information retrieval purposes or as features for a much more insightful movie recommendation algorithm. This may add to a content-based filtering approach.

4. Using motif detection as a search function. For example, the algorithm could be given a query phrase in order to find the most generically similar phrases across all given sequences. This may be useful in a commercial setting such as finding particular client phrases in communications.

In order to take conversational motif mining from an interesting idea to a useful tool, it is critical to begin developing a benchmark or other means of quantitative evaluation of algorithm performance. Some techniques may be adapted from the Bioinformatics field.

As mentioned in Chapter 2, the motif detection algorithm we have made use of is based on a more traditional method. This is done with the intention of testing the idea and not obtaining state of the art performance. A next step may be to consider comparing various motif detection algorithms to find the best one to use. Our current algorithm may be used as a benchmark. There are multiple clear directions we can take in terms of algorithmic extension. The largest of this is an adaptive motif length size. Because motif detection is an unsupervised task, we might not have a clear idea of the ideal length of motif to work with. A further step is to remove the restriction of a fixed length across motifs. This will likely improve the types of patterns that are extracted. Humans often have a fixed intent which can be expressed in a sequence of thoughts; however, there are often embellishments, interruptions or sidetracks. Catering for such variability will provide higher quality conversational patterns.

We have shown that the extensions we add to the original algorithm perform well (Chapter 5). We have removed the restriction of fixed sequence length for the motif detection algorithm. We have also added an ability to see which sequences are more likely to contain the motif and which sequences are not well aligned. It is possible to introduce further modifications to our algorithm in order to make this an algorithmic attribute as opposed to a post-hoc analysis. This would discard sequences that are not adding good alignment information throughout the progression of the algorithm. The result would likely be a clearer motif with the relevant sequences that it comes from. Another extension we have made is the addition of an update scheme for element alignment calculations. Elements in our sequences are no longer categorical, and we have

catered for this. In future work, we may want to compare this functionality to existing multivariate motif detection schemes such as in [38] or more recent work. We mention that it is possible to extract different motifs depending on the type of initialization we choose. There is a need to monitor the stability in the system as well as to avoid extracting patterns based on local minima. There is scope for future work to introduce techniques to combat this.

The preliminary results we achieve show that the ideas we present are possible to implement and have potential value. There are many opportunities for improvement and development of novel applications and methods stemming from this starting point. We have shown that it is possible to mine conversations for motif patterns using the methods described in this work. We are able to detect conversational cues that appear to be linked with emotional cues. A generalizable pipeline would assist with domain transferability. We are able to build such a tool using this work as a base.

Bibliography

[1] Arunkumar Bagavathi, Pedram Bashiri, Shannon Reid, Matthew Phillips, and Siddharth Krishnan. Examining untempered social media: analyzing cascades of polarized conversations. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 625–632, 2019.

[2] Timothy L Bailey and Charles Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine learning*, 21(1-2):51–80, 1995.

[3] Amir Bakarov. A survey of word embeddings evaluation methods. *arXiv preprint arXiv:1801.09536*, 2018.

[4] Franco Borsini. *Pharmacology of 5-HT6 receptors, Part I.* Academic Press, 2010.

[5] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[6] Leonid Boytsov and Bilegsaikhan Naidan. Engineering efficient and effective non-metric space library. In Nieves R. Brisaboa, Oscar Pedreira, and Pavel Zezula, editors, *Similarity Search and Applications - 6th International Conference, SISAP 2013, A Coruña, Spain, October 2-4, 2013, Proceedings*, volume 8199 of *Lecture Notes in Computer Science*, pages 280–293. Springer, 2013.

[7] E. Cambria and B. White. Jumping nlp curves: A review of natural language processing research [review article]. *IEEE Computational Intelligence Magazine*, 9(2):48–57, 2014.

[8] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35, 2017.

[9] Hongshen Chen, Zhaochun Ren, Jiliang Tang, Yihong Eric Zhao, and Dawei Yin. Hierarchical variational memory network for dialogue generation. In *Proceedings of the 2018 World Wide Web Conference*, pages 1653–1662, 2018.

[10] X. Cheng, X. Yan, Y. Lan, and J. Guo. Btm: Topic modeling over short texts. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):2928–2941, 2014.

[11] Giovanni Ciriello and Concettina Guerra. A review on models and algorithms for motif discovery in protein–protein interaction networks. *Briefings in Functional Genomics and Proteomics*, 7(2):147–156, 2008.

[12] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[13] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364, 2017.

[14] Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

[15] Modan K Das and Ho-Kwok Dai. A survey of dna motif finding algorithms. In *BMC bioinformatics*, volume 8, page S21. Springer, 2007.

[16] Thomas A Down and Tim JP Hubbard. Nestedmica: sensitive inference of over-represented motifs in nucleic acid sequence. *Nucleic acids research*, 33(5):1445–1453, 2005.

[17] S. A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327, 1976.

[18] Milica Gašić, Dilek Hakkani-Tür, and Asli Celikyilmaz. Spoken language understanding and interaction: machine learning for human-like conversational systems, 2017.

[19] Gerald Z Hertz, George W Hartzell III, and Gary D Stormo. Identification of consensus patterns in unaligned dna sequences known to be functionally related. *Bioinformatics*, 6(2):81–92, 1990.

[20] Jianjun Hu, Bin Li, and Daisuke Kihara. Limitations and potentials of current motif discovery algorithms. *Nucleic acids research*, 33(15):4899–4913, 2005.

[21] Elke Kirschbaum, Manuel Haußmann, Steffen Wolf, Hannah Sonntag, Justus Schneider, Shehabeldin Elzoheiry, Oliver Kann, Daniel Durstewitz, and Fred A Hamprecht. Lemonade: Learned motif and neuronal assembly detection in calcium imaging videos. *arXiv preprint arXiv:1806.09963*, 2018.

[22] Sven Köhler, Phillip Seitzer, Marc T Facciotti, and Bertram Ludäscher. Improved motif detection in large sequence sets with random sampling in a kepler workflow. *Procedia Computer Science*, 9:1999, 2012.

[23] Hoang Thanh Lam, Ninh Dang Pham, and Toon Calders. Online discovery of top-k similar motifs in time series data. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 1004–1015. SIAM, 2011.

[24] Charles E Lawrence, Stephen F Altschul, Mark S Boguski, Jun S Liu, Andrew F Neuwald, and John C Wootton. Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. *science*, 262(5131):208–214, 1993.

[25] Charles E Lawrence and Andrew A Reilly. An expectation maximization (em) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins: Structure, Function, and Bioinformatics*, 7(1):41–51, 1990.

[26] Nung Kion Lee, Xi Li, and Dianhui Wang. A comprehensive survey on genetic algorithms for dna motif prediction. *Information Sciences*, 466:25–43, 2018.

[27] Yongrui Li, Jun Yu, and Zengfu Wang. Dense semantic matching network for multi-turn conversation. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1186–1191. IEEE, 2019.

[28] Carlo Lipizzi, Luca Iandoli, and José Emmanuel Ramirez Marquez. Extracting and evaluating conversational patterns in social media: A socio-semantic analysis of customers' reactions to the launch of new products using twitter streams. *International Journal of Information Management*, 35(4):490–503, 2015.

[29] Chia-Wei Liu, Ryan Lowe, Iulian V. Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation, 2016.

[30] JLEKS Lonardi and Pranav Patel. Finding motifs in time series. In *Proc. of the 2nd Workshop on Temporal Data Mining*, pages 53–68, 2002.

[31] Tamar Jeffers McDonald. *Romantic comedy: Boy meets girl meets genre*. Columbia University Press, 2007.

[32] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018.

[33] Luis AA Meira, Vinıcius R Máximo, Alvaro L Fazenda, and Arlindo F da Conceiçao. An improved network motif detection tool. *arXiv preprint arXiv:1804.09741*, 2018.

[34] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*, 2017.

[35] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[36] Sebastian Nagel, Jun 2017.

[37] Xiaoyong Pan, Yong-Xian Fan, Jue Jia, and Hong-Bin Shen. Identifying rna-binding proteins using multi-label deep learning. *Science China Information Sciences*, 62(1):19103, 2019.

[38] Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, and Umeshwar Dayal. Multi-dimensional sequential pattern mining. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 81–88, 2001.

[39] Pedro Ribeiro, Fernando Silva, and Marcus Kaiser. Strategies for network motifs discovery. In *2009 Fifth IEEE International Conference on e-Science*, pages 80–87. IEEE, 2009.

[40] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, Jan 2008.

[41] Majed Sahli, Essam Mansour, and Panos Kalnis. Parallel motif extraction from very long sequences. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 549–558, 2013.

[42] Scott Sczopek. DNA Motif Finding via Gibbs Sampler, April 2017.

[43] Idan Szpektor, Deborah Cohen, Gal Elidan, Michael Fink, Avinatan Hassidim, Orgad Keller, Sayali Kulkarni, Eran Ofek, Sagie Pudinsky, Asaf Revach, et al. Dynamic composition for conversational domain exploration. In *Proceedings of The Web Conference 2020*, pages 872–883, 2020.

[44] Martin Tompa. An exact method for finding short motifs in sequences, with application to the ribosome binding site problem. In *ISMB*, volume 99, pages 262–271, 1999.

[45] Ngoc Tam L Tran and Chun-Hsi Huang. Modside: a motif discovery pipeline and similarity detector. *BMC genomics*, 19(1):755, 2018.

[46] J.D. White and J. White. *The Analysis of Music*. G - Reference, Information and Interdisciplinary Subjects Series. Scarecrow Press, 1984.

[47] Elisabeth Wong, Brittany Baur, Saad Quader, and Chun-Hsi Huang. Biological network motif detection: principles and practice. *Briefings in bioinformatics*, 13(2):202–215, 2012.

[48] Han Yuan, Meghana Kshirsagar, Lee Zamparo, Yuheng Lu, and Christina S Leslie. Bindspace decodes transcription factor binding signals by large-scale sequence embedding. *Nature methods*, 16(9):858–861, 2019.

[49] Huan Zhao, Yingqi Zhou, Yangqiu Song, and Dik Lun Lee. Motif enhanced recommendation over heterogeneous information network. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2189–2192, 2019.