Multi-Faceted Set Expansion

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Neel Kabra
(111901057)

Parichita Das
(111901039)

INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

CERTIFICATE

*This is to certify that the work contained in the project entitled "**Multi-Faceted Set Expansion**" is a bonafide work of **Neel Kabra** (**Roll No. 111901057**) and **Parichita Das** (**Roll No. 111901039**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Palakkad under my guidance and that it has not been submitted elsewhere for a degree.*

Dr. Koninika Pal

Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Palakkad

Acknowledgements

This project was envisioned by Professor Koninika Pal, without whose help we would not even be doing it right now. She guided us through lots of brainstorming sessions, in which we discussed and understood the necessities and requirements for this project. She helped us throughout the project and enabled us to learn more about the field this project is based on.

I also would like to thank my teammate, who has given an equal contribution and effort into the project as well as the document. Only our combined workforce has been able to bring fruition to this project.

Contents

Chapter 1

Introduction

In exploratory search and comparative data analysis, calculating related entities for a given seed object is a crucial task. Calculation of related entities is a key task for search and even recommendation systems. For example, if a user express interest in a celebrity or a company, search engines do not just return information about the entity of interest, but also highly related entities. For example, if a user searches {*Sundar Pichai*}, an algorithm to search for related entities could return the set {*Jeff Bezos, Elon Musk, Satya Nadella*}. This is the well known **set expansion problem**.

Current studies only provide single faceted set expansion, i.e., only expanding a set on one particular trait. But in real life, this does not actually happen. Words in real life often contain multiple semantic meanings.

For many use cases, it is not sufficient to have a simple set expansion. It may also be required to know **why** a particular set was expanded to the output. For instance, in the previous example, {*Jeff Bezos, Elon Musk, Satya Nadella*} were expanded on trying to expand {*Sundar Pichai*} because they are also CEOs of technological companies. It is also important to note that expansion based on one particular trait is often insufficient. In real life, this does not actually happen. Words in real life often contain multiple semantic meanings. In fact, the set expansion of the initial set {*Sundar Pichai*} could return different sets based on different aspects. Here is an example of two sets with two entities in each set:

- {*Eric Schmidt, Larry Page*} - Past Google CEOs
- {*Elon Musk, John F. Kennedy*} - Stanford Alumni

This idea could also be applied in recommendation systems for movies. For instance, if an animated Disney movie was given as an input, we could expand the intial to find expansions for both Disney movies and animated movies, and let the user decide what appeals to them more.

This problem can be modelled as a **Multi-Faceted Set Expansion Problem**, where we want to expand an initial set of entities based on different characteristics. The initial set of entities is also called the **seed set**. The label for the different sets are also produced, and they are termed as **facets**.

In this project we want to tackle the problem of **Multi-Faceted Set Expansion**. We try to compute multiple groups of entities that convey different aspects in an explainable manner. These different groups must have high similarity within the group, and high diversity across different groups. Our aim is to understand the need, the requirements, the past approaches and try to see what else we can add to the table. We intend to try to find a new solution or make optimizations in previous approaches.

1.1 The Problem Statement

According to the formal problem statement, we are given:

1. Universe of entities E = $\{e_1, e_2, \ldots, e_n\}$
2. Set of labeled facets S = $\{S_1, S_2, \ldots, S_m\}$ $where \, \forall i \in \{1, 2, ...m\}, \; S_i \subset E$
3. Query q, Number of output groups l and size of groups k.

The objective is compute l sets of entities each of size k such that, each group can inherit its label from some input facet.

$$\text{G} = \{G_1, G_2, \ldots, G_l\}$$

such that, for each $G_j \subset S_i$ for some input facet S_i from which G_j can inherit its label. The following conditions must be satisfied: The following conditions must be satisfied:

1. Pairwise similarity between q and faceted groups G_i i.e., *Sum of similarity over all $e_i \in G_i$ and q is maximized*

2. Pairwise similarity between entities in each group G_i i.e., *Similarity summed up over all entity pairs $e_i, e_j \in G_i$, is maximized to reflect coherence inside the group*

3. Pairwise similarity across groups G_i, G_j summed up over all entity pairs (e_i, e_j) with $e_i \in G_i$ and $e_j \in G_j$ is minimized to preserve diversity.

The problem boils down to finding faceted groups G that maximize the following function f(G):

$$\alpha \sum_{\forall i, y \in G_i} rel(q, y) + \beta \sum_{\forall i, y, z \in G_i} rel(y, z) - \gamma \sum_{\forall ij, i \neq j, m \in G_i, n \in G_j} rel(m, n)$$

where q is the Query set, α, β, γ are tunable parameters and *rel* denotes the similarity between pairs of entities, subject to:

- $\forall G_i, |G_i| = k$ (bounded size of each group)
- $|G| = l$ (bounded number of groups)
- $\exists S_p, G_i \subseteq S_p$ (Selecting groups from input facets)

The solution we are looking for should be able to provide these multiple facets with high similarity within a facet, but also high diversity among different facets, since two different facets like (*CEOs, Indian CEOs*) would not be very interesting. There are a few publications on multi-faceted set expansion as well, which will be discussed in detail in the upcoming chapter.

While this chapter discussed the problem we are trying to tackle and the current solution which is used by recommendation systems, chapter 2 discusses the existing works related to the set expansion problem i.e., *State-of-The-Art* Solutions. chapter 3 presents the Baseline Implementation where we discuss in detail one of the set expansion

models **FUSE**. Here we see how we adapt the FUSE code for our dataset and the results thus received.

In chapter 4 we shall be discussing a greedy approach for the Multi-Faceted Set Expansion. In the next chapter, chapter 5, we see how our algorithm performs against baselines. Finally in chapter 6 we discuss our plans of future work.

For a detailed explanation of our algorithm and a step-by-step output, please refer to chapter 7.

Chapter 2

Prior Works

We will be discussing the State of the Art Algorithms to tackle the problem of **Multi-Faceted Set Expansion**. We will look at the unique approach each of these algorithms take and what we can do better.

2.1 FUSE

FUSE [1] is an unsupervised framework developed for multi-faceted set expansion. FUSE consists of three major components(modules) to tackle this problem.

1. Facet Discovery Module
2. Facet Fusion Module
3. Entity Expansion Module

FUSE uses a large text corpus to extract skip-grams that words belong to, and then try to cluster skip-grams extracted for all different seeds. After obtaining multiple skip-gram clusters for each seed, we then need to determine the coherent semantic facets among all seeds and generate the coherent skip-gram clusters.

For example if we start with two seed words *"Sundar_Pichai"* and *"Elon_Musk"*, facets for the former might be *Google_CEOs* and *Stanford_Alumni* while that for the latter may be *Stanford_Alumni* and *Tesla_CEOs*. Clearly the coherent semantic facet is

Stanford_Alumni.

Once the fusion is done, and fused clusters are obtained, FUSE expands the obtained clusters using the BERT expansion model.

In the cluster fusion stage of FUSE, there are several key things to be noted. Firstly, the number of clusters, i.e. the number of facets, that will be produced at the end of the algorithm is not known. Secondly, the clustering is done using the word embedding, and dimensions of the embedding vectors can be very high. To tackle the problem of unknown number of clusters and high dimension space, FUSE uses affinity propagation for clustering the skip-grams.

2.2 FaSets

This algorithm works based on random walks over the candidate network and computes faceted groups of limited size. The candidate entities and labeled facets (categories) are collected from a large pool of information from Wikipedias and KGs (Knowledge Graphs) using Information Extraction.

Given an input entity and a large set of potential facets each in the form of a labeled entity set, task is to compute a compact group of facets with a small set of salient features such that:

- Each group is highly related to the seed entity
- Entities per group are highly related to each other
- Selected groups diversify overall, by being pairwise dissimilar

The objective is compute l sets of entities called **faceted groups** each of size k such that, each group can inherit its label from some input facet.

Faceted Set Expansion **(FSX)** boils down to the problem of finding faceted groups G that maximize the following function f(G):

$$\alpha \sum_{\forall i, y \in G_i} rel(q, y) + \beta \sum_{\forall i, y, z \in G_i} rel(y, z) - \gamma \sum_{\forall ij, i \neq j, m \in G_i, n \in G_j} rel(m, n)$$

where q is the Query set, α, β, γ are tunable parameters and rel denotes the similarity between pairs of entities, subject to:

- $\forall G_i, |G_i| = k$ (bounded size of each group)
- $|G| = l$ (bounded number of groups)
- $\exists S_p, G_i \subseteq S_p$ (Selecting groups from input facets)

FSX is proven to be **NP Hard** and **Submodular**.

2.3 SEISA

In this paper [2], the issue of finding additional entities that also pertain to the same concept set by extending a collection of given seed entities into a more complete set is investigated. The use of "Godrej" and "Philips" as seed entities to generate additional entities (such as "Samsung") within the same idea group of electronics companies is a typical illustration. A variety of online data sources are used, such as lists taken directly from web sites and user inquiries from a web search engine, to find these pertinent organizations. Web data tend to be very noisy despite being highly varied and abundant in information that typically span a broad variety of interest areas. on these erratic info sources. In light of this, a brand-new general framework is suggested based on iterative similarity aggregation and in-depth experimental findings are presented to demonstrate that this strategy beats earlier methods when using general-purpose web data for set enlargement in terms of both accuracy and recall. Specifically, lists that have been taken out of HTML web sites (the Web List data) and online search query logs (the Query Log data) are examined. Each time, the data is represented as bipartite networks, with the contexts of the potential terms as nodes

on one side and candidate entities on the other. There are various ways in which we can model edges in the graph for query log data.

Given a seed set S and an expanded seed set R, the quality of R with respect to S is given by the function:

$$Q(R, S) = \alpha * S_{rel}(R, S) + (1 - \alpha) * S_{coh}(R)$$

where, α is a constant weight that balances the emphasis between *relevance* and *coherence*.

The problem statement thus becomes, given a set of candidate terms **U**, a subset seed set **S** and a function $\boldsymbol{Sim : UXU \rightarrow [0, 1]}$ that gives similarity between any two elements of this set, the task is to identify the expanded seed set **R** of size k such that the **Quality function** is maximized.

There are two greedy algorithms for the above problem statement, the **Static Thresholding Algorithm** and the **Dynamic Thresholding Algorithm** [3], that iteratively refine a candidate R of size k to maximize the above defined function.

2.4 EgoSet

By fusing pre-existing user-generated ontologies with a word-similarity metric based on skip-grams, EgoSet [4] provides a novel approach to handling set expansion. EgoSet generates sparse word ego-networks that are centred on the seed terms and can capture semantic equivalence between words by combining the two resources. In order to reflect the various semantic classes of the seeds, EgoSet shows that the resulting networks have internally coherent clusters that may be used to produce non-overlapping expansions. Empirical testing of EgoSet against state-of-the-art baselines demonstrates that it is not only capable of capturing numerous facets in the input query but also generates expansions for each facet with greater accuracy.

Chapter 3

Adaptation of FUSE for our dataset

FUSE being the State of the Art Algorithm, we tried to modify the implementation of FUSE to suit our needs. In this chapter we will discuss the model of FUSE and how we are adapting it for our dataset.

First, let us discuss what our dataset is and then we will step by step discuss the algorithm and the changes we are making.

3.1 Dataset

Our dataset consists of many entities mapped to a list of categories/facets they belong to. For example, $Cristiano\_Ronaldo : [LaLigaFootballers, PortugeseFootballers]$. The dataset contained entities from three different categories :

- People : 50,000 popular people associated with 65,000 facets.
- Movies : 50,000 popular movies associated with 54,000 facets.
- Companies : 5,000 popular companies associated with 33,000 facets.

All entities in the dataset were collected from Wikipedia and also exist in YAGO.

3.1.1 Cleaning Dataset

We must now see the format in which our dataset is given to us.
The text corpus is given to us as a **tsv** (Tab separated Values). For example a line from the input file is given below:

Elizabeth_II wordnet_military_officer_109816771

Now as we can see here, the first word *"Elizabeth_II"* is an entity and the second word *"wordnet_military_officer_109816771"* which after cleaning becomes *military_officer* is a category/facet for the entity. So we can use the second word directly as the skip gram and we just need to modify it according to our use.

So we read all the datasets line by line and first split them by tabs. Now we add the entity as it is just in all lower case with all the underscores. Now after adding a space we add the facet (all lower case) by replacing the underscores by spaces. We also remove the first word of the facet which is just the source of the facet, and the last word if its a number greater than 5 digits. So the above example will become

elizabeth_ii military officer

Now all these strings are put in a list (**all_text**) and we can now proceed to the next step of the process.

3.2 Facet Discovery Module

The main aim of this module is to recognise the different semantic facets that the seed set represents. FUSE does this by identifying skip-grams of the seeds in the large text corpus D. Skip-grams refer to the words that surround the seed word. For example, the skip-grams of the word "apple" may be "_____ tree" and "CEO of _____ Inc.". The distributional hypothesis [5] says that the neighboring skip-grams can reflect the semantic meaning of the word.

For our dataset, we create the skip-grams by following the given steps:

1. For each of the seeds, we iterate over **all_text** and check if the first word matches the seed, otherwise we skip the string altogether.

2. In case we do not skip the string, we replace the first word (entity) by the identifier "[**MASK**]" and add the rest of the string as it is to the skip gram.

So if one of the seeds is "**elizabeth_ii**", an example string of the skip-gram would look like:

[MASK] military officer

Once the skip-grams has been extracted, FUSE uses the word embeddings of the skip-grams to discover the semantic meaning. The embedding of a skip-gram is calculated as the average of the component words. Next, FUSE aims to cluster the facets using their embeddings. However, here, the number of clusters cannot be specified. Hence, FUSE uses affinity propagation to find the best skip-gram clusters. This clustering is done for each seed word to discover the different semantic facets each seed represents. Example of Facet discovery would be to discover that the word "apple" may have facets "fruit" and "technology company".

3.2.1 Clustering Skip-grams for our Dataset

We try to form clusters by fusing word embeddings of all skip grams. Individual words are represented as real-valued vectors in a preset vector space in a method known as a **Word embedding**. The method is frequently referred to as deep learning because each phrase is assigned to a single vector, and the vector values are learned in a manner resembling a neural network. The concept of using a dense dispersed representation for each word is essential to the method.

For getting the sentence embedding we must search for the embeddings of the different words in each skipgram from the Glove embeddings [6] or the Wikipedia embeddings [7]. We add all such embeddings of the words in a sentence and take a normalized average to get sentence embedding of each sentence in the skipgram. We use different methods and compare the results for all of them.

Method 1

In this method, we split the skipgram strings by spaces and search for each individual word's embedding to take it into consideration. For example, let there be a skipgram such as:

[MASK] people from palm beach, florida

We clean each individual word by stripping it off of commas and full-stops. Among these, we must ignore **Stopwords of English** like *'of'*, *'the'* etc. Now if we search for individual words like *'people'* and *'palm'*, the embeddings we will get will be very distant from what we originally intended. Something belongs to the category *'people'* is not a very **interesting** fact and the category is not very informational. On the other hand, *'palm'* is not even a category for the entity here. So we cannot expect good results from this method.

Method 2

In this method we try to solve the problem we saw in the previous method. We want to look for meaningful, more descriptive categories. For example, we could search for the following word embeddings:

- people_from_palm_beach
- palm_beach
- palm_beach_florida

So what we do is make different combinations of words in the same order and join them by underscores and then search for their embeddings and finally take the average of these embeddings. We can do this in two different ways which are discussed below.

- **Method 2.a**

  In this method, we start with all the words and join them by underscore. So in

the given example, our first search would be for *people_from_palm_beach_florida*. If we are able to find an embedding for this, we remove all these words and the algorithm will automatically end here. Otherwise we now skip one word from the left or right. So we will search and take embedding of either *people_from_palm_beach* or and *from_palm_beach_florida*. Once again if we do not find any embedding for either of them, we keep repeating this process of searching for smaller and smaller subsets of the sentence and if we find the embeddings, we remove the groups of words. Like this we end up finding more meaningful facets and add their embeddings to the sentence embedding.

- **Method 2.b**

  This is just a small modification to the method above where if we find the embedding for any group of words, we **do not remove them from the sentence**. So for example, if we find the embedding for *people_from_palm_beach*, we will still look for embeddings of *people* and *palm_beach* both. This is something that we were not doing in **2.a**. This may add some unnecessary embeddings to the sentence but in some cases it will increase weightage of categories which have more importance, by adding similar categories twice.

We shall discuss all the results received from above in coming sections. Now that we have the skip gram clusters, we may proceed to the next step.

3.3 Facet Fusion Module

After obtaining different semantic facets of each word, FUSE now tries to combine the the clusters of the different words to get coherent facets. For example, consider seed words "apple" and "orange". For apple, the semantic facets discovered would be "fruit" and "technology company", but for "orange" the facets discovered would be "fruit" and "color". Here, the coherent facet would be "fruit".

FUSE tries to determine if a facet of seed word A matches any facet of the seed word B. Suppose seed word A has r skip-gram clusters $S_A = \{S_A^{(1)}, S_A^{(2)}, .., S_A^{(r)}\}$, and seed word B has t skip-gram clusters $S_B = \{S_B^{(1)}, S_B^{(2)}, .., S_B^{(t)}\}$. If they share k facets, then we can represent the *coherent/fused clusters* as $S_{A,B} = \{S_A^{(i_1)} \cup S_B^{(j_1)}, ..., S_A^{(i_k)} \cup S_B^{(j_k)}\}$.

3.3.1 Calculation of correlation between two skip-gram clusters

Suppose that facet A_x, a facet of the seed word A corresponds to a skip-gram cluster $X = [x_1; ...; x_m]$, where X has m skip-gram vectors, and facet B_y, a facet of seed word B corresponds to a skip-gram cluster $Y = [Y_1; ...; Y_n]$, where Y has n skip-gram vectors. We want to calculate correlation between any two skip-gram clusters, X and Y to see if they correspond to the same semantic facet.

Let X and Y be two skip-gram clusters. Their sense vectors are considered as a linear combination of their skip-gram vectors of each skip-gram cluster. Let u and v be the sense vectors for X and Y respectively. Now, the correlation between two skip-gram clusters is given by solving the following optimization problem.

$$\begin{aligned} &\max_{a,b} \frac{\mathbf{u}^T\mathbf{v}}{||\mathbf{u}||.||\mathbf{v}||} \\ &\text{s.t. } \mathbf{u} = \mathbf{Xa},\ \mathbf{v} = \mathbf{Yb} \end{aligned}$$

Upon solving this problem, we get the common sense vectors u^* and v^*, and the correlation between X and Y is defined as

$$corr(\mathbf{X}, \mathbf{Y}) = \mathbf{u}^{*T}\mathbf{v}^* \tag{3.1}$$

After correlation between any two facets is defined, we now need to cast a binary decision on whether the cluster X matches semantically with any facet of the word B. It can be noted that if any facet of seed A (ex. A_x) matches with any facet of seed B (ex. B_y), then $corr(A_x, B_y)$ is higher than correlation between A_x and any other facet of seed B. Based on the above information, we can find the best matching facet B^*

in word B, and generate one coherent skip-gram cluster A_1B^*. We can continue this process for all facets of word A.

For this section, we do not make any modifications to the actual code and follow the same implementation as the baseline. The sentence embeddings for each of the sentence of each skip gram is considered as their **Skip-Gram Vectors** and now for each skip-gram clusters we take a linear combination of all the embeddings to get the **Sense Vector** of each skip-grams. We can find the coherent skip-gram(s) by **eq. 3.1**.

3.4 Entity Expansion

The coherent skip-gram clusters obtained from the facet fusion model are used to expand the seed set. Iterative approaches, or graph based algorithms are commonly used in to complete the set expansion. FUSE models the expansion problem as a Masked Language Model (MLM) which leverages the power of BERT and contextual representations.

For our data, we have done entity expansion by implementing one of these two methods.

3.4.1 BERT Model

For each skip-gram denoted as sg, FUSE computes the MLM probability $h_{c,sg}$ for each word candidate c in the vocabulary by a pre-trained BERT model. Therefore, given a set P of skip-grams, the weight w_c of a word candidate c is calculated as:

$$w_c = \sum_{sg'} h_{c,sg'}$$

The final entity expansion process simply ranks all word candidates by their weights.

3.4.2 Using Wikipedia embeddings

For this we once again get the sentence embeddings of the fused cluster(s) by using *Method 2.b* of **section 3.2.1**. Now after adding all the embeddings we also add the embeddings of the seed words with some added weightage, which is basically the count

of embeddings found divided by the affinity propagation value. This is to ensure that the seed words get as much importance as the cluster. Now we use the api provided by **Wikipedia2Vec** [8] to get the most similar words based on the combined embedding we give.

3.5 Results

Let us now look at the results we received for the **FUSE** code against our dataset. The colab notebook for the same can be found here. [9]

1. We use **Method 1** from section **3.2** for Clustering Skip-grams (**preference = -5**) and then use the **BERT Model** for expansion. Here are the results.

S.no.	**Query**	**Top Skip-grams after fusion**	**Expanded Entities**	**Ground Truth: Expanded Entities**
1	Toyota	[MASK] multinational companies headquartered in japan	Facet 1: [japan, toyota, china, mitsubishi, bmw]	**Motor Vehicle Manufacturers of Japan**: [Honda, Subaru Corporation, Mitsubishi] **Car Manufacturers**: [Kia Motors, General Motors, Tesla_Inc.]
2	Nokia	[MASK] companies based in silicon valley	Facet 1: [sony, ibm, microsoft, china, boeing, california]	**Mobile Phone Manufacturers**: [Samsung, Qualcomm, Huawei] **Companies based in Silicon Valley**: [Hewlett-Packard, Oracle_Corporation, LinkedIn]

2. We use **Method 2.a** from section **3.2** for Clustering Skip-grams (**preference = -5**) and then use the **BERT Model** for expansion. Here are the results.

S.no.	Query	Top Skip-grams after fusion	Expanded Entities	Ground Truth: Expanded Entities
1	Toyota	[MASK] multinational companies headquartered in japan, [MASK] companies listed on the london stock exchange	Facet 1: [japan, mitsubishi, china, toyota, tokyo]	**Motor Vehicle Manufacturers of Japan**: [Honda, Subaru Corporation, Mitsubishi] **Car Manufacturers**: [Kia Motors, General Motors, Tesla_Inc.]
		[MASK] car manufacturers, [MASK] engine manufacturers	Facet 2: [toyota, nissan, bmw, renault, mazda]	
2	Nokia	[MASK] companies based in silicon valley	Facet 1: [ibm, sony, nokia, boeing, finland, apple]	**Mobile Phone Manufacturers**: [Samsung, Qualcomm, Huawei] **Companies based in Silicon Valley**: [Hewlett-Packard, Oracle_Corporation, LinkedIn]

3. We use **Method 2.b** from section **3.2** for Clustering Skip-grams (**preference = -1**) and then use the **BERT Model** for expansion. Here are the results.

S.no.	Query	Top Skip-grams after fusion	Expanded Entities	Ground Truth: Expanded Entities
1	Toyota	[MASK] multinational companies headquartered in japan, [MASK] companies of japan	Facet 1: [japan, mitsubishi, toyota, honda, nissan]	**Motor Vehicle Manufacturers of Japan**: [Honda, Subaru Corporation, Mitsubishi] **Car Manufacturers**: [Kia Motors, General Motors, Tesla_Inc.]
		[MASK] bus manufacturers, [MASK] car manufacturers	Facet 2: [bmw, toyota, mazda, nissan, volkswagen]	
2	Nokia	[MASK] companies based in silicon valley	Facet 1: [sony, ibm, microsoft, china, finland, japan, nokia]	**Mobile Phone Manufacturers**: [Samsung, Qualcomm, Huawei] **Companies based in Silicon Valley**: [Hewlett-Packard, Oracle_Corporation, LinkedIn]

4. We use **Method 2.b** from section **3.2** for Clustering Skip-grams (**preference = -3**) and then use the **Wiki2Vec Model** for expansion. Here are the results.

S.no.	Query	Top Skip-grams after fusion	Expanded Entities	Ground Truth: Expanded Entities
1	Toyota	[MASK] multinational companies headquartered in japan, [MASK] companies listed on the london stock exchange	Facet 1: [arvinmeritor, pamcor, dongvo, autocomp, gmipl]	**Motor Vehicle Manufacturers of Japan**: [Honda, Subaru Corporation, Mitsubishi] **Car Manufacturers**: [Kia Motors, General Motors, Tesla_Inc.]
		[MASK] bus manufacturers, [MASK] car manufacturers	Facet 2: [toyota, nissan, lexus, mazda, honda, hyundai]	
2	Nokia	[MASK] mobile phone manufacturers, [MASK] telecommunications companies	Facet 1: [nokia, motorola, symbian, samsung, pantech, benq]	**Mobile Phone Manufacturers**: [Samsung, Qualcomm, Huawei] **Companies based in Silicon Valley**: [Hewlett-Packard, Oracle_Corporation, LinkedIn]

Chapter 4

A Greedy Approach to Implement Multi-Faceted Set Expansion

We will now look at a more greedy and iterative approach to the **Multi Faceted Set Expansion**. Before looking at our algorithm, we will discuss the problem statement and go over some definitions from the **FaSets Algorithms** that we will need for our own algorithm.

4.1 Formal Problem Statement

According to the formal problem statement of FaSets, we are given:

1. Universe of entities E = $\{e_1, e_2, \ldots, e_n\}$
2. Set of labeled facets S = $\{S_1, S_2, \ldots, S_m\}$ $where\, \forall i \in \{1, 2, ...m\},\ S_i \subset E$
3. Query q, Number of output groups l and size of groups k.

The objective is compute l sets of entities called **faceted groups** each of size k such that, each group can inherit its label from some input facet.

$$\mathrm{G} = \{G_1, G_2, \ldots, G_l\}$$

such that, for each $G_j \subset S_i$ for some input facet S_i from which G_j can inherit its label. The following conditions must be satisfied: The following conditions must be satisfied:

1. Pairwise similarity between q and faceted groups G_i i.e., *Sum of similarity over all $e_i \in G_i$ and q is maximized*

2. Pairwise similarity between entities in each group G_i i.e., *Similarity summed up over all entity pairs $e_i, e_j \in G_i$, is maximized to reflect coherence inside the group*

3. Pairwise similarity across groups G_i, G_j summed up over all entity pairs (e_i, e_j) with $e_i \in G_i$ and $e_j \in G_j$ is minimized to preserve diversity.

Faceted Set Expansion **(FSX)** boils down to the problem of finding faceted groups G that maximize the following function f(G):

$$\alpha \sum_{\forall i, y \in G_i} rel(q, y) + \beta \sum_{\forall i, y, z \in G_i} rel(y, z) - \gamma \sum_{\forall ij, i \neq j, m \in G_i, n \in G_j} rel(m, n)$$

where q is the Query set, α, β, γ are tunable parameters and *rel* denotes the similarity between pairs of entities, subject to:

- $\forall G_i, |G_i| = k$ (bounded size of each group)
- $|G| = l$ (bounded number of groups)
- $\exists S_p, G_i \subseteq S_p$ (Selecting groups from input facets)

FSX is proven to be **NP Hard** and **Submodular**.

4.2 Saliency Score

There are two types of saliency score given to the two different types of nodes - **Entities** and **Facets**.

- **Saliency score of Entity Nodes ($score_e$):** Page views of wikipedia page for an entity as its saliency score. For each page, extract total page views over a period of month.

- **Saliency score of Facet Nodes ($score_f$):** Number of existing multilingual Wikipedia editions for category pages as a measure of facet Saliency. We use multilingual Wikipedia editions for the Wikipedia page of the object in POs as their saliency score, for facets from KBs or infoboxes.

Both scores are dampened by log values and normalized between 0 and 1

4.3 Relatedness Score

For an entity e_x, how much they are related to a group is called **Group Membership**. It is represented as a vector of size m given by,

$$\hat{e}_x = [\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3, \ldots, \mathrm{v}_m]$$

where v_i is the saliency score of the ith facet, i.e.,

$$v_i = score_f(S_i) \text{ , if } e_x \in S_i$$
$$\mathrm{v}_i = 0 \text{ , otherwise}$$

Now the **Relatedness score** is calculated by the *Weighted average of the weighted-Jaccard similarity* between their weighted-group memberships and the proximity based on their page-views. This is given by,

$$rel(e_x, e_y) = w_1 \frac{\sum_i min(\hat{e}_{xi}, \hat{e}_{yi})}{\sum_i max(\hat{e}_{xi}, \hat{e}_{yi})} + w_2(1 - |score_e(e_x) - score_e(e_y)|)$$

The parameters w_1 and w_2 control two components of the relatedness measure.

4.4 Popularity Score

For our algorithm, we need to also use a factor to score the facet. More specifically, we like to emphasize the importance of a facet and the label it has.

For example, if a facet label is *'Living_People'*, it is a very general facet and is not providing any important information about the entities it is connected to. On the

other hand facet label like *'Presidents_Of_India'* is specific to only a small group of people compared to the thousands of entities in the entire dataset. We also use a **popFactor** which is an experimental value that decides how much importance the popularity score will have.

So in order to give more value to a more informational facet we use a function **PopScore** where:

$$PopScore(S_i) = popFactor * \frac{(\log_2 a_i)^2}{(\log_2 b_i)^{0.7}}$$

where,

$S_i \in S$,

a_i = Number of entities $e \in S_i$,

b_i = Number of *interesting* entities $e \in S_i$, (Interesting entities are the ones that are selected in each of the facets during the BFS on the candidate graph, which will be discussed in the next section)

4.5 Our Algorithm

Now that we have discussed some necessary definitions, we will now look at a greedy approach for the problem statement.

4.5.1 Input and Output

- m : Number of Input Facets
- q : Input Query Set
- U : Entity Set (Universe of entities)
- k : Number of Facets to produce
- l : Number of Entities required per output Facet

4.5.2 Iterative Algorithm to find Faceted Groups

Now we will discuss the iterative algorithm. We are initially going to work with some θ facets each with p entities, where $\theta >> k$ and $p >> l$.

Finding Potential Candidate Facets:

We build a candidate graph for the query by selecting potential entities and facets that can form the faceted groups and provide their explainable labels. For this purpose, we explore the bipartite graph, starting from the each of the query nodes and alternating between entity nodes and facet nodes using breadth-first search until we gather θ facets. These facets become our **Candidate Facets** and these are our initial sets, with all the entities that came up during the BFS. We shall put all the candidate facets in a set **V**. Let us now start with our initialization of peer groups.

Algorithm 1 Initializing the sets and peer groups

1: We create a score for each entity in each facet and call it $Score(e_f)$ where e is the entity and f is the facet it belongs to.
2: **for** all $V_i \in V$ **do**
3: **for** all $e \in V_i$ **do**
4: $Score(e_i) = \alpha . \sum_{q_i \in q} rel(q_i, e)$
5: **end for**
6: **end for**
7: **for** all $V_i \in V$ **do**
8: Create a peer group G_i for every V_i and take top x entities ($x < 5$) based on $Score(e_i)$
9: **end for**
10: Create a set G containing all G_i
11: We create a group score **Gs** for each peer group where $\forall\ G_i \in G$, $Gs(G_i) = \sum_{e \in G_i} Score(e_i)$
12: $Gs* = \max_{G_i \in G} Gs(G_i)$

13: We **remove** all V_i from V (and G_i from G) for which $Gs(G_i) < threshold.Gs*$

14: **for** all $G_i \in G$ **do**

15: **for** all $e \in G_i$ **do**

16: $Gs(G_i)$ += $\beta . \sum_{e_x \in G_i, e_x \neq e} rel(e_x, e)$

17: **end for**

18: **end for**

19: $Gs* = \max_{G_i \in G} Gs(G_i)$

20: We **remove** all V_i from V (and G_i from G) for which $Gs(G_i) < threshold.Gs*$

The *threshold* is an experimental value and takes dynamic values based on the highest score obtained or the number of groups to be removed.

Now we will create a ranking and make the **top k peer groups** each having x entities, $(x \leq 5)$.

Algorithm 2 Ranking the top k peer groups

1: Create a set **T**

2: **while** $|T| < k$ **do**

3: Create a temporary $Gs_t = Gs \; \forall \; G_i \in G$

4: **for** all $G_i \in G$ **do**

5: **for** all $e \in G_i$ **do**

6: $Gs_t(G_i)$ −= $\gamma . \sum_{e_y \in G_j \forall G_j \in T} rel(e_y, e)$

7: **end for**

8: **end for**

9: Add G_a to T $\ni G_a = \arg\max_{G_i \in G} Gs_t(G_i)$

10: $Gs(G_a) = Gs_t(G_a)$

11: **end while**

Now that we have the top k groups T with x elements each, we can start the iterative algorithm. We will call the initial T as T^0 and in the t_{th} iteration we will work with T^t. Every iteration we will create the next T^{t+1}.

Algorithm 3 Iterative Algorithm

$t \leftarrow 0$

$count \leftarrow x$

while True **do**

  while True **do**

    for $G_i \in G$ **do**

      $e_r = \arg\min_{e \in G_i}(\alpha. \sum_{q_i \in q} rel(q_i, e) + \beta. \sum_{e_x \in G_i} rel(e_x, e))$

      $s_{min} = \min_{e \in G_i}(\alpha. \sum_{q_i \in q} rel(q_i, e) + \beta. \sum_{e_x \in G_i} rel(e_x, e))$

      for $e \in V_i$ **do**

        $s = (\alpha. \sum_{q_i \in q} rel(q_i, e) + \beta. \sum_{e_x \in G_i} rel(e_x, e))$

        if $s > s_{min}$ **then**

          $s_{min} = s$

          $e_t = e$

        end if

      end for

      Replace e_r with e_t

    end for

    Repeat **Algorithm 2** to make set T^{t+1}

    if $T^{t+1} = T^t$ **then**

      break

    end if

    $T = T^{t+1}$

  end while

  $count = count + x$

  for $T_i \in T$ **do**

    $e_{max} = \arg\max_{e \in V_i, e \notin G_i}(\alpha. \sum_{q_i \in q} rel(q_i, e) + \beta. \sum_{e_x \in G_i} rel(e_x, e) - \gamma. \sum_{e_y \in G_j \forall G_j \in T, i \neq j} rel(e_y, e))$

    Add $e_{max} to T_i$

    $Gs(G_i) \mathrel{+}= \max_{e \in V_i, e \notin G_i}(\alpha. \sum_{q_i \in q} rel(q_i, e) + \beta. \sum_{e_x \in G_i} rel(e_x, e) - \gamma. \sum_{e_y \in G_j \forall G_j \in T, i \neq j} rel(e_y, e))$

  end for

for $G_i \in G, G_i \notin T$ **do**

$e_{max} = \arg\max_{e \in V_i, e \notin G_i} (\alpha . \sum_{q_i \in q} rel(q_i, e) + \beta . \sum_{e_x \in G_i} rel(e_x, e) - \gamma . \sum_{e_y \in G_j \forall G_j \in T, i \neq j} rel(e_y, e))$

Add e_{max} to T_i

$Gs(G_i) \mathrel{+}= \max_{e \in V_i, e \notin G_i} (\alpha . \sum_{q_i \in q} rel(q_i, e) + \beta . \sum_{e_x \in G_i} rel(e_x, e) - \gamma . \sum_{e_y \in G_j \forall G_j \in T, i \neq j} rel(e_y, e))$

$S_{min} = \min_{T_j \in T} Gs(G_j)$

if $S_{min} < Gs(G_i)$ **then**

Replace the minimum score peer group with G_i in T

end if

end for

Repeat **Algorithm 1** to recalculate all scores

Repeat **Algorithm 2** to re-rank top k peer groups and finalize it as T^{t+1}

if *count* $\geq \nu.x$ **then**

break while loop

end if

end while

After the iterative algorithm is over we should have k faceted groups with $\nu.x$ entities each. If $\nu.x > l$, then we are done as we just need to take the top l entities from each group based on the $Score(e)$. Otherwise, we need to expand each of the k groups in T based on the entities present. This will leave us with **k** Faceted Groups and **l** entities each. l

Algorithm 4 Expanding Final Groups

for all $T_i \in T$ **do**

for all $e \in V_i$ s.t. $T_i \subset V_i$ **do**

$Score(e) = \alpha . \sum_{q_i \in q} rel(q_i, e) + \beta . \sum_{e_x \in T_i^t} rel(e_x, e)$

end for

Add $l - \nu.x$ top entities to T_i based on $Score(e)$

end for

Chapter 5

Evaluation

5.1 Dataset

The dataset is the same one we used in FUSE, the details of which are listed below. It contained many entities mapped to a list of categories/facets they belong to. For example, $Cristiano\_Ronaldo : [LaLigaFootballers, PortugeseFootballers]$. The dataset contained entities from the following category:

- People : 50,000 popular people associated with 65,000 facets.

All entities in the dataset were collected from Wikipedia and also exist in YAGO. On top of that, there are a chosen 40 queries for popular people, which were used as benchmark queries.

5.2 Ground Truth

As mentioned in the FaSets paper, a-priori gold-standard groups were obtained through crowd sourcing. For each query, 10 diverse and informative facets were selected. Then, top-20 candidate facets based on four simple scoring functions:

1. The size of a facet
2. $max(score_e(e))$ for the entities from a facet
3. $avg(score_e(e))$ the entites of from a facet

4. $score_f$ for a facet

Annotators were then asked to choose five diverse and informative facets. The facets are then aggregated and top-10 of them were selected as the gold-standard groups.

Table 5.1: **Gold-Standard groups for Nelson Mandela**

Socialists	Presidents of South Africa	Revolutionists
Karl Marx	Thabo Mbeki	Che Guevara
Noam Chomsky	P. W. Botha	Vladimir Lenin
Leon Trotsky	Jacob Zuma	Malcolm X
George Galloway	F. W. de Klerk	Mahatma Gandhi
George Orwell	Kgalema Motlanthe	Leon Trotsky

5.3 Evaluation Metrics

Two evaluation metrics were used to evaluate our results.

5.3.1 Quality@l.k

Suppose an algorithm generates output $\mathcal{G}$ with l groups with k entities each, and we have ground truth $\mathcal{GT}$: m groups with n entities each. We define the quality of the algorithm against the ground truth, $Quality@l.k$, as follows.

Conceptually, we consider all mappings between the output groups $\mathcal{G}$ and the ground-truth groups $\mathcal{GT}$.

For each mapping $\mathcal{G} \to \mathcal{GT}$, we calculate the *Quality* by averaging the precision for each group in $\mathcal{G}$ against its mapped group in $\mathcal{GT}$

$$Quality@l.k = \frac{\sum_{g_i \to gt_j} \frac{|g_i \cap gt_j|}{k}}{l} \quad i \in l, j \in m$$

From all possible mappings, we select the one where the Quality metric is maximal, and we present $Quality@l.k$ for this mapping.

5.3.2 B-cubed

We also use the B-cubed measure [10], a standard metric for evaluating co-reference resolution by clustering. B-cubed computes the weighted average of the per-entity precision scores over all output groups, with precision defined as the fraction of correct elements in an output group containing the entity.

$$\text{B-cubed}(\mathcal{G}, \mathcal{GT}) = \frac{\sum_{i=1}^{|\mathcal{GT}_i|} \sum_{j=1}^{|\mathcal{G}_i|} \frac{|g_j \bigcap gt_i|^2}{|g_i|}}{\sum_{j=1}^{|\mathcal{G}_i|} |g_i|}$$

5.4 Parameter tuning

In our algorithm we have four hyper-parameters α, β, γ and $popFactor$. These were tuned to obtain the best results based on the evaluation metrics mentioned above. Our set expansion algorithm was run on a set of 40 people varying from different domains, and we tried to maximise our scores of the above metrics.

We first fix a value for $popFactor$ (0.5), and then try to vary values of α, β, γ to get a local maxima. After the maximal values of α, β, γ are obtained, we then vary $popFactor$ for these values. The algorithm is run on 40 seed queries separately and we produce 5 groups of 5 entities each for every query, i.e. $k = 5$, $l = 5$.

Table 5.2: **Variation of α, β, γ with fixed popFactor**

popFactor	α	β	γ	B-cubed	Quality@l.k
0.5	0.1	0.1	0.8	0.15	0.14
		0.4	0.5	0.10	0.11
	0.2	0.1	0.7	0.18	0.17
		0.4	0.4	0.17	0.12
	0.3	0.1	0.6	0.18	0.17
		0.3	0.4	0.16	0.134
	0.4	**0.1**	**0.5**	**0.19**	**0.18**
		0.2	0.4	0.16	0.16
	0.5	0.1	0.4	0.19	0.17
		0.3	0.2	0.15	0.14

We can see that a maximum value is obtained for $\alpha = 0.4, \beta = 0.1, \gamma = 0.5$. Now, we try to vary popFactor to get the optimal value of popFactor for the maximal

obtained triplet.

Table 5.3: **Variation of score with popFactor with fixed α, β, γ**

α	β	γ	popFactor	B-cubed	Quality@l.k
0.4	0.1	0.5	0.5	0.19	0.178
			0.7	0.19	0.175
			0.9	0.20	0.178
			1.1	0.20	0.177
			1.3	**0.21**	**0.178**
			1.5	0.20	0.178

We finally obtained the highest score with popFactor value of 1.3. Hence, we finalized these values for the parameters, $\alpha = \mathbf{0.4}, \beta = \mathbf{0.1}, \gamma = \mathbf{0.5}$, and $popFactor = \mathbf{1.3}$.

5.5 Evaluation using gold-standard groups

We used the above discussed metrics to score the set expansions of other existing state-of-the-art methods against ours.

Table 5.4: **Comparison of scores obtained with existing methods**

		B-cubed			Quality@l.k		
l	k	Our Algorithm	FaSets	SEISA	Our Algorithm	FaSets	SEISA
3	5	**0.21**	0.18	0.05	0.23	**0.25**	0.11
5	5	**0.21**	0.18	0.06	**0.19**	**0.19**	0.08
3	10	**0.06**	**0.06**	0.01	**0.19**	**0.19**	0.01
5	10	**0.06**	**0.06**	0.01	**0.13**	**0.13**	0.01

5.6 Analysis and Discussion

5.6.1 Run-time Analysis

One of the main criteria on which we must test our algorithm is its speed of computation. It should be able to achieve results at least as fast as **FaSets Algorithm**, if not faster. One of the major differences between FaSets and our algorithm is that the latter

produces one group in every iteration and therefore its runtime was observed to increase linearly with number of groups. On the other hand, our algorithm adds one new element to every peer group in every iteration, so it is not linearly dependent on the number of peer groups to be produced.

We will try to compare the run-times of these two algorithms by generating time to output one group. We will run both the FaSets algorithm and our algorithm and try to output $k = 5$ groups (we will normalize this to get runtime per group). We vary number of entities l per group and number of candidate entities θ.

Before we look at the results, there is one thing to note. For every query, in both algorithms, we must compute the **Relationship Matrix** which has similarity score between any two entities of the *Candidate Graph*. The time to compute this would be same irrespective of the number of groups we are trying to compute. We have denoted this time as **t'** and separated it out from the run-time of the rest of the algorithm, since it is a common time that will only depend on θ (The number of Candidate Entities). Let us look at the results below:

Table 5.5: **Runtime Performance Comparison (in sec.)**

		Our Algorithm						FaSets					
θ		500	1000	1500	2000	2500	3000	500	1000	1500	2000	2500	3000
t'		1.53	6.28	14.13	25.25	39.26	55.07	1.61	6.67	14.95	26.83	40.79	60.23
l	5	0.19	0.48	0.91	1.36	2.07	2.55	0.26	0.59	1.21	2.03	2.71	4.09
	10	0.37	0.88	1.47	2.05	2.69	3.38	0.58	0.9	1.72	2.36	3.46	4.91
	20	1.13	2.38	3.74	4.86	6.15	6.96	3.05	3.16	3.84	5.47	6.18	18.3
	40	4.21	8.65	12.41	15.8	18.65	22.65	7.74	11.28	19.19	18.43	23.68	40.04

As we can see in the table, the run-time for our algorithm significantly decreases with increase in both $\boldsymbol{l}$ and $\boldsymbol{\theta}$.

All our codes can be accessed from our github page [11] and all the required files and folders, including our results, can be downloaded from our drive link [12].

5.6.2 Qualitative Analysis

The following examples demonstrate how our algorithm performs on two single seeded queries.[1]. All examples have been run for $k = 3$ faceted groups and $l = 5$ entites per group.

Table 5.6: Max Plank with 3 faceted groups, 5 elements in each

Methods	Faceted Groups	Explanation
Our Algorithm	Max Born, Erwin Schrödinger, Arnold Sommerfeld, Niels Bohr, Werner Heisenberg	Quantum Physicists
	Max von Laue, Max Born, Wilhelm Röntgen, Werner Heisenberg, Philipp Lenard	German Nobel Laureates
	Gustav Kirchhoff, Michael Faraday, Antonie van Leeuwenhoek, Niels Bohr, C.V. Raman	Opticians
FaSets	Max von Laue, Philipp Lenard, Arnold Sommerfeld, Max Born, Werner Heisenberg	German physicists
	C. V. Raman, Erwin Schrödinger, C. V. Raman, Erwin Schrödinger, Niels Bohr, Paul Dirac	Nobel laureates in physics
	Max Born, Werner Heisenberg, Wolfgang Pauli, Niels Bohr, Arnold Sommerfeld	Quantum physicists
SEISA	Max Born, Niels Bohr, Abdus Salam , Max von Laue, Arnold Sommerfeld	Physicists
	Philipp Lenard, Carl Bosch, Robert Bunsen, Hans Geiger, Rudolf Clausius	Ethnic German people
	Arnold Sommerfeld, Max von Laue, Max Born, Abdus Salam, Niels Bohr	Physicists
FUSE	Robert Luxton, Arthur McMaster, Pratson, Cudaback, Gritson	-
	Vikkor, Merkwelt, Debacksy, John Stuart, Kalamik	-

[1]Note that FUSE does not produce labels for the output sets and may not produce required number of peer groups

Table 5.7: **Barack Obama with 3 faceted groups, 5 elements in each**

Methods	Faceted Groups	Explanation
Our Algorithm	Franklin D. Roosevelt, Abraham Lincoln, Jimmy Carter, George W. Bush, Theodore Roosevelt	Presidents of the United States
	Al Gore, Woodrow Wilson, Jimmy Carter, Henry Kissinger, Theodore Roosevelt	Nobel Peace Prize laureates
	John Kerry, John F. Kennedy, Ted Kennedy, Joe Biden, Robert F. Kennedy	Democratic Party United States Senators
FaSets	Abraham Lincoln, Robert F. Kennedy, Franklin D. Roosevelt, Jimmy Carter, Theodore Roosevelt	American people
	Abraham Lincoln, Franklin D. Roosevelt, Jimmy Carter, Theodore Roosevelt, John F. Kennedy	Presidents of the United States
	Theodore Roosevelt, Barack Obama, Franklin D. Roosevelt, Abraham Lincoln, Jimmy Carter, Eminem	American people of English descent
SEISA	Keith Ellison, Rob Portman, George McGovern, Joe Biden, Elihu Root	American people
	Al Gore, Theodore Roosevelt, Jimmy Carter, Woodrow Wilson, Óscar Arias	Communicator
	Jamie Raskin, Rob Portman, Keith Ellison, Michelle Obama, Joe Biden, Barack Obama	Living people
FUSE	William Armstrong, Oscar Magoni, Douglas Hodkin, Arthur McMaster, Piereson	-
	Dean Mcgraw, John Cook, William Armstrong, Oscar Magoni, Milkovisch	-
	Micheal S. Berman, Gerald Malloy, Christina Hale, Neal Zimmers, Eldon Nygaard	-

As we can notice, our results are comparable to *FaSets* Algorithm because just like it, we also incorporate coherence and dissimilarity scores. Therefore, we are also able to ensure diversity among different groups while maintaining similarity within the same peer group.

Chapter 6

Conclusion and Future Work

Until now, we have seen and understood the **Multi-Faceted Set Expansion** and looked at different algorithms which provide a solution to this problem. We thoroughly studied the *State-Of-The-Art*, **FUSE** and saw how it worked on our dataset using different methods.
We then saw a greedy iterative approach to the problem statement, **FaSets**. The major difference between the two methods is that FaSets give more information on the **Labels/Categories** based on which the clusters are being formed and it also has a definitive algorithm to produce exact number of output groups with exact number of entities.

The next part of our work comprises of coming up with a new greedy algorithm which can produce results comparable to FaSets and try to produce the output in *lesser computation time*. On studying the algorithm and the working of FaSets, we found out that it needs to create the *Similarity Matrix* (or the *Relationship Matrix*) in every iteration. It must be noted that this is a very costly operation as there are about *2000-3000* entities in the beginning of the algorithm and finding the similarity between any two entities is highly time consuming. *FaSets* also creates one group per iteration.
Our aim was to make the new algorithm minimize computation time by having to

create the Similarity Matrix only once. We also tried to make our algorithm more efficient by adding new elements to every group in every iteration instead of the previous approaches taken. After coming up with different versions of the new algorithm [11] and coding it out, we finally started the evaluation and comparison process.
Our algorithm was able to produce faceted groups with quite informative labels which can be comparable to FaSets' algorithm. We were also able to achieve slightly faster results and all of these can be seen in **chapter 5**.

In Future Work, we hope to assess output groups from our algorithm using crowdsourcing workers, who would rate the coherence within each group and diversity across groups. To carry out the assessment, we would display the output groups for a particular query and ask some evaluators to rate each outcome using three labels: *bad*, *moderate*, and *good*. We would then calculate the average score for our algorithm using some metric and see how it performs against other baselines.

Our code can be accessed from our github page [11] and all the required files and folders, including our results, can be downloaded from our drive link [12].

Chapter 7

Appendix A

In this section, we attempt to show a closer working of our algorithm by explaining the steps in more detail and then giving some example seed query and its result.

7.1 Detailed working of the algorithm

For ease of understanding, we have divided our algorithm into the following steps. We will try to explain each step with maximum details and clarity.

7.1.1 Candidate Graph Creation

We find the candidate facets and entities with which we compute our initial peer groups. In order to do this, just like Faceted Set Expansion (FSX), we shall consider a huge pool of labeled facets as input. We start out with a Bipartite graph which has two different types of nodes. The first type are all *Entities* and the second are *Facets/Labeled Groups*. Any two nodes of different types have an edge between them *iff* the entity node belongs to the facet node. Given any query set, our target is to find all the facets and entities belonging to the former which are the most related to the seed.

Starting from each of the query nodes, we do a **Breadth First Search** on the graph. We attempt to find θ candidate entities and also consider all facets which have an edge with these entities. Once we have collected our **Candidate Graph**, we use

this to calculate the similarity between any two entities. We have already discussed how to find similarity between any two entities and we use the same to create our **Relationship Matrix**.

7.1.2 Initial Peer Groups Computation

Before the creation of initial sets iteration, we find the relatedness score between each of the entities of each facet, with the seed entity. Our ultimate aim is to create peer groups which has the maximum **Quality Score**.

The Quality Score comprises of three values:

- **Alpha Contribution**: Relation of an entity with the seed query(s).
- **Beta Contribution**: Relation of entities within a peer group with other members of the same peer group.
- **Gamma Contribution**: Relation of entities of a peer group with members of other peer groups. We want to keep the groups as unique as possible and therefore, for this we take the negation of similarity score as we want to decrease similarity between entities from two different groups.

Having found out the alpha contribution of all entities of every facet, we create the peer group for each of the facets and include the x entities from that facet which shall contribute the maximum to the overall Quality Score of the group.
Now we compute the beta contribution of each of the entities that we considered in the peer groups for each facet and add it to overall quality score of the groups.
Then, we sort these groups before beginning our gamma calculation. The contribution by gamma is basically the penalty for having similar groups. We create the set for the top peer groups in which we will *finally* have the **k peer groups** with **l entities each**. The group ranked highest, based on alpha and beta contributions, is added to the top group's set, considering its the **best group**. The following groups' scores are calculated by giving them a penalty based on the gamma contribution of the entities of their peer groups with the entities of the groups which have already been selected in

the top groups' set. After the gamma calculation is completed, we now have K facets ranked on α, β and γ scores.

7.1.3 Replacement Algorithm

After we have ranked our initial peer groups, it is possible to get some better overall quality score by replacing one or more entities in the peer groups which might increase the alpha, beta contributions and be less penalized due to similarity with the entities of the current top peer groups.

For replacement, we simply see if there exists an element in a facet which could increase the group score if included in the respective peer group instead of entity with the least contribution to that group. To do this, we simply try to calculate the element which is contributing the least in that peer group, and try too see if any other element would contribute more, based on the alpha and beta scores. We would then again rank the peer groups based on their group scores like how we did earlier.

It is possible that the top K groups change after one iteration of replacement is done, so if a replacement is done, the recalculation of the top K groups is done completely again. This happens because it is possible that some entity which was earlier not present in the peer group, because of its low alpha contribution, actually has a better coherence with the other entities of that peer group and also lesser similarity with the entities of the other peer groups. The replacement and reordering algorithm is done until there are no more changes in the top k groups.

7.1.4 Adding Entities to Peer Groups

After we have established the best possible top k groups with x entities per peer group, we can no move on to adding new elements to each of the peer groups. For adding a new elements, we calculate how much an element could add to the score of the group based on its alpha, beta, and gamma contributions and the element which adds the maximum value was chosen to be added to the peer group. It should be noted that here, we if an element is not available to be added, then the group is penalized for it.

Here, addition is done considering a penalty with the highest k-1 top groups. If it is seen that the new element has resulted in it having a better score than the k^{th} top group, we replace this group in the top groups' set. Since, it is possible that the top K Groups' set changes after an element is added, we need to do a fresh ranking of the groups to make sure the all subsequent additions are done considering the current top k-1 groups.

7.1.5 Iterative Algorithm

The main iterative algorithm is done after creating the initial sets. We then run the replacement algorithm and add a new element until the number of required elements have been added. This iterative algorithm ensures the best possible top k groups with any number of entities.

7.1.6 Expansion Algorithm

Ideally we run the iterative algorithm only for 6 or 7 times. By the time we have our top k groups with 6 entities each, we have already made the groups pretty distinct from each other by alternate replacement and addition of entities and penalizing the groups every time based on high similarity between each other. It can be safely said that each group has enough number of peers and further expansion of these groups can be done solely based on the alpha and beta contributions of the entities only. If l is less than 6, we can just take the best top l entities per group, otherwise, expand the groups by taking the entities with highest contributions based on alpha, beta scores.

7.2 Example Output

Here, we try to demonstrate the working of the algorithm by showing the formation of sets after every step. We try to compute $k = 5$ peer groups for seed query **Max Planck**, with each peer group having $l = 5$ elements.

```
wikicat Lorentz_Medal_winners
Gerard 't Hooft
Wolfgang Pauli
Arnold_Sommerfeld

wikicat Quantum_physicists
Max Born
Werner Heisenberg
Arnold_Sommerfeld

wikicat Opticians
Niels Bohr
Gustav Kirchhoff
C._V._Raman
```

(a) **Figure 1**

```
wikicat Lorentz_Medal_winners
Gerard 't Hooft
Wolfgang Pauli
Arnold_Sommerfeld

wikicat Quantum_physicists
Niels Bohr
Max Born
Werner_Heisenberg

wikicat Opticians
Niels Bohr
Michael Faraday
C._V._Raman
```

(b) **Figure 2**

Fig. 7.1: Initialization of peer groups

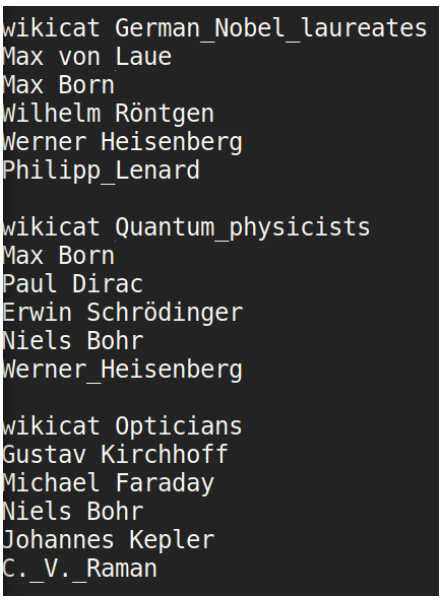
```
wikicat Quantum_physicists
Max Born
Erwin Schrödinger
Niels Bohr
Werner_Heisenberg

wikicat Lorentz_Medal_winners
Gerard 't Hooft
Wolfgang Pauli
Frank Wilczek
Arnold_Sommerfeld

wikicat Opticians
Niels Bohr
Johannes Kepler
Michael Faraday
C._V._Raman
```

(a) **Figure 3**

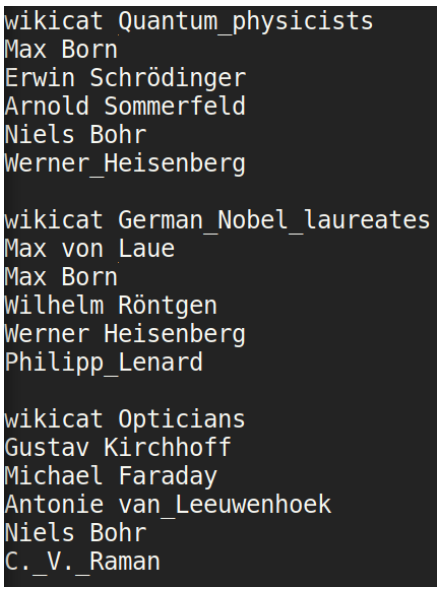
```
wikicat Lorentz_Medal_winners
Gerard 't Hooft
Wolfgang Pauli
Edward Witten
Arnold_Sommerfeld

wikicat Quantum_physicists
Max Born
Erwin Schrödinger
Niels Bohr
Werner_Heisenberg

wordnet optician_110380000
Gustav Kirchhoff
Michael Faraday
Niels Bohr
C._V._Raman
```

(b) **Figure 4**

Fig. 7.2: Addition of new element



(a) **Figure 5**



(b) **Figure 6**

Fig. 7.3: Final Results

Figure 1 7.1 shows the result after the initial iteration.

Figure 2 7.1 shows the result after the replacement algorithm has run. It can be seen that in the facet Quantum Physicists, *Arnold Sommerfield* has been replaced by *Neils Bohr*. Similarly, there are changes in the facet Opticians as well.

Figure 3 7.2 shows the result after the an element has been added to each facet. We can see that some groups have been re-ordered due to the score of the element that was added. The group scores change with the addition of an element, so it is possible that groups are re-ordered.

In figure 4 , another replacement algorithm has been run, and some elements have been replaced in a few facets. We can also see the top 3 groups have changed after replacement of some elements.

Another addition has been performed in figure 5 7.3. The top 3 groups have changed again, and German Nobel Laureates is now in the top 3 groups.

The final result after replacement of the last addition is displaced in figure 6 7.3. The group Quantum Physicists has moved up with the replacement of *Paul Dirac* by *Arnold Sommerfeld*.

References

[1] W. Zhu, H. Gong, J. Shen, C. Zhang, J. Shang, S. Bhat, and J. Han, *FUSE: Multi-faceted Set Expansion by Coherent Clustering of Skip-Grams*, 02 2021, pp. 617–632.

[2] Y. He and D. Xin, "Seisa: Set expansion by iterative similarity aggregation," in *Proceedings of the 20th International Conference on World Wide Web*, ser. WWW '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 427–436. [Online]. Available: https://doi.org/10.1145/1963405.1963467

[3] "Static vs Dynamic Thresholds," https://last9.io/blog/static-threshold-vs-dynamic-threshold-alerting/.

[4] X. Rong, Z. Chen, Q. Mei, and E. Adar, "Egoset: Exploiting word ego-networks and user-generated ontology for multifaceted set expansion," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, ser. WSDM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 645–654. [Online]. Available: https://doi.org/10.1145/2835776.2835808

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[6] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[7] I. Yamada, A. Asai, J. Sakuma, H. Shindo, H. Takeda, Y. Takefuji, and Y. Matsumoto, "Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020, pp. 23–30.

[8] "Wikipedia2Vec API Usage," https://wikipedia2vec.github.io/wikipedia2vec/usage/.

[9] "FUSE Adaptation Colab Notebook," https://colab.research.google.com/drive/1TnkWqbXSYfZAU_qbefnlW41EEMyEDgoG#scrollTo=-1SPGA5T1_dC.

[10] A. Bagga and B. Baldwin, "Algorithms for scoring coreference chains," 1998.

[11] "GitHub Link To Our Code," https://github.com/PerryAgent/Multi-Faceted-Set-Expansion.

[12] "Google Drive Link To Project Files," https://drive.google.com/drive/u/2/folders/1YMmG1ImS2JeSYJvd7PGmxn8FNDwttg_r.