MINING LEGO DATA SETS TO SUPPORT LEGO DESIGN

BY

XIAODAN ZHANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Advisor:

Professor ChengXiang Zhai

Abstract

The Lego Group invented LEGO bricks. These bricks could be put up together to build creative LEGO sets of different themes, such as a Volkswagen T1 Camper Van model (in "Sculpture" theme) or The Simpsons House model (in "Town" theme). LEGO accompanies nearly everyone from youth to adulthood. The age groups of fans range from kids in pre-school to elder people who have grandchildren. With such a strong and huge fan base, there appear a lot of websites that provide LEGO data of sets, parts, minifigures as well as online communities for fans to share their experience about LEGO sets. However, there barely exists any research in this rich data source to discover knowledge and insights about how each LEGO part plays a role in a LEGO set, in its own part category and in a LEGO theme; how each LEGO set is different from the other one in the aspects of theme and part components. There are a lot of interesting questions we can address from the datasets that will not only help better LEGO designs, but will also help LEGO fans or potential customers make efficient purchasing decisions when they get more familiar with LEGO sets and parts.

To address these needs, in this thesis, we propose a systematic method of mining LEGO datasets of sets and parts to support LEGO design. Treating each LEGO set as a document and each part in it as a word, we are able to apply data mining techniques, such as Topic Model and K-Means Clustering to find statistics of sets and parts. The preliminary experiment results show that the proposed methods can automatically construct a LEGO Brick Lexicon that shows a part's relationship with other parts, sets and themes, discover knowledge about typical LEGO construction patterns and create hybrid theme recommendations. We believe this is a step forward to helping LEGO designers create more attractive sets with pragmatic parts as well as improving the building experience of LEGO fans/builders.

To My Parents and My Friends, for their love and support.

Acknowledgments

This thesis would not be possible without all of the support from a group of people who helped me in finding the data, developing the system and providing constructive feedbacks. I would like to express greatly appreciation to my advisor, Professor ChengXiang Zhai, for always being supportive and encouraging. He has given me the freedom to explore data that are interesting and motivated me to solve novel and challenging research problems. He is so patient, respects to my idea and trusts my ability. Thank you, Professor Zhai.

Also, I would like to thank University of Illinois Urbana-Champaign Department of Computer Science for providing me with great experience and resources in both undergraduate and graduate studies.

Finally, I would also like to thank my parents for their support and encouragement throughout my study, who were always there for me during my most difficult times.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

LEGO is the world renowned line of construction toys - interlocking plastic bricks that are manufactured by The Lego Group[1], a privately held company based in Billund, Denmark. LEGO bricks can be assembled and connected in so many ways. They provide you the flexibility to work with bricks of various sizes, shapes and colors. People combine them to create something fun, unique or even valuable, including constructing objects such as vehicles, buildings and robots, and even part of famous movie scenes. The Lego Group creates new sets with new themes every year as well as invents new shapes of bricks with cool decorations.

A great amount of people in the world grow up with LEGO. The LEGO fans share opinions about LEGO sets, exchange minifigures and discuss about investing in collectible toys of movies or limited seasonal editions, etc. They also collaboratively make recommendations to each other about sets to purchase and share ideas about sorting and storing bricks efficiently. They even talk about which part LEGO should make. However, LEGO players need support to help them make interesting creations (MOCs, term mentioned in Section 2.2) that go beyond the existing models. When they face a huge pile of parts, if they know what pieces are used commonly in a specific theme, they can relate to some ideas from sets in that theme. When they find some small subsets of parts, if they know where these parts are used and what companioned parts are, it would be easier for them to make purchasing decisions and buy companioned parts that help them build a bigger model. And when they want to learn what traditional logic of using some parts, if they know how these parts contribute to sets in different themes, they will get a bigger idea about what these parts can construct. In order to meet these needs, we propose to use data mining to help them and address some of these problems. While many different kinds of data sets have been mined, to the best of our knowledge, no work has been done on mining LEGO sets.

As it is said by Albert Einstein, “Play is the highest form of research”, communications between LEGO fans generate a lot of useful information and create several interesting research problems. We live in a data-driven world and are surrounded by data everyday and everywhere. But not many people have realized

that LEGO sets contain valuable information and in fact behave as a great data source that wait for people to explore. The information we get from each LEGO set is not just limited to the price, age group, number of pieces, set id and theme. We also know what parts it contains, how many times they appear and what categories they belong to.

The main idea we propose in this thesis is to view a LEGO set as a"document" with parts as "words". A word has a part-of-speech tag and can appear more than one time in a document; a part has a category it belongs to and can be used in several sub-models of a set. The only difference between a document and a LEGO set is that the order of parts used is uncertainty. For example, the number to combine 6 two-by-four studded LEGO bricks of the same color is 915,103,765[2]. Though there are rare research studies in mining LEGO datasets, there are data mining algorithm studies in texts datasets, which we can use for reference.

In this thesis, we conduct multiple data mining tasks in finding knowledge about LEGO sets in the aspects of the relationship between sets and parts, sets and themes, parts and themes as well as construction patterns, in hopes of providing useful insights for LEGO design and improving the user experience of LEGO customers. Our work will open up a new application direction in data mining of mapping knowledge discovery in toy patterns to designs.

1.2 Contributions

In order to solve the problems stated above, we address several challenges.
For LEGO Designers (D), we help them answer the following questions:
1. When designing and creating a new piece in a piece category, will it contribute to some specific themes?
2. When creating a new set, is it necessary to create all categories of bricks or just a smaller subset of them?
3. When creating a new theme, will it be very unique with unpopular bricks or will it have some similarities with existed theme so that people follow similar instructions and use similar sets of bricks?

For LEGO Builders/Fans (B), our research provides answers to the following questions:
1. What are the parts that are used to support most or least sets?
2. What are the parts that are used frequently in a specific theme?
3. What are the parts that are frequently appear together with a specific part, i.e. the companioned parts?
4. What themes are similar with each other and what are unique? For unique themes, the pieces in them tend to be not sharable among many sets.
5. What sets are of different themes but are in fact similar based on the part types they contain?

To better understand how each part contributes to a set and a theme, we propose to automatically construct a LEGO Part Lexicon by doing statistical analysis of each part including what category the part is; what themes and sets this part appears in and how frequent(D1, B1, B2); what frequent companioned parts a specific part possesses (B3). This lexicon helps find the most frequent and least frequent parts among all sets as well as among sets with a specific theme.

Next, we propose a novel approach by building topic models on part categories for all sets data. From there, we get a few topics and for each set and each theme, it has a distribution over these topics. By doing this, we can see that in order to build sets in a theme, we probably will only need parts of a subset of all part categories (D2) instead of using all of them. And it is the subset that shows unique properties of a theme. Additionally, we are also able to know the similarities between all sets by computing the similarity measures of topic coverage vectors of these sets, which will not only help cluster similar themes (B4) by applying K-Means Clustering algorithm, but will also show similar sets from different themes (D3, B5). Eventually, we have data visualizations of theme clusters and for a specific set, we can create hybrid theme recommendations of sets from different themes but use similar categories of parts.

The rest of the thesis is organized as follows. In Chapter 2, we discuss about all related works including the existing LEGO online databases and communities, as well as some existing text mining research. Then in Chapter 3, we provide the background information including basic concepts in LEGO and all tools we use for experiments we conduct. Next, in Chapter 4, we discuss about problem formulation, which covers both the LEGO datasets we explore and the formal definition of the three data mining tasks finished in this thesis. In Chapter 5, we illustrate the first data mining task, the construction of LEGO Part Lexicon in detail. In Chapter 6, we talk about the second data mining task, patter discovery of LEGO parts. Next, in Chapter 7, we discuss about the third data mining task, hybrid LEGO theme and set recommendations. Finally, we conclude and discuss future works in Chapter 8.

Chapter 2

Background

2.1 Existing Online LEGO Sets Databases and Communities

There exist several websites and online communities for people to purchase LEGO sets, parts, minifigures and discuss ideas of building with LEGO bricks. Lego.com[1] is the official website of the Lego Group and Lego brand toys that was created in 1996. The website serves as an online shop as well as a product catalog. However, if a set is too old or is sold out, we will not find that set. The advantage of the Lego.com is that it has the most up-to-date set information and the disadvantage of it is that there are no statistics or insights about each set. Brickset.com[3] was founded in 1997, which serves as both a LEGO information database with high-quality scans instruction covers and an online community for LEGO fans to write reviews, record details of their collection and make wish-lists. The advantage of the website is that it has nearly complete data of LEGO sets and parts with a good interface for people to search. It also assigns some tags to each set, which indicates more features about the set. However, the disadvantage is that it does not look into a specific part and when we see a strange-shaped part, we can barely infer its functionality or which themes/sets it serves for.

Fortunately, Rebrickable.com[4], a website created by LEGO fans and designed to show alternate models that can be built from LEGO sets besides the stock model, started to look into parts and sets in a deeper angle. This website not only has an advanced search engine for also up-to-date LEGO sets data, shown in Figure 2.1, but it also offers more useful information and statistics about a part. In Figure 2.2, we know how common this part among all parts and among all sets, as well as a visualized graph showing how many times this specific part is created and how many sets it serves. Rebrickable.com has already stepped forward in discovering knowledge and insights that support LEGO design. However, this existing work does not provide enough insightful information for helping future LEGO designs or get LEGO a builder a deeper understanding about how parts work with other parts, contribute to a theme or a set, etc. In this thesis, our goal is to mine more knowledge from LEGO data set, in hoping to discover LEGO construction patterns and provide a deeper analysis about relationships between LEGO parts, sets and themes.

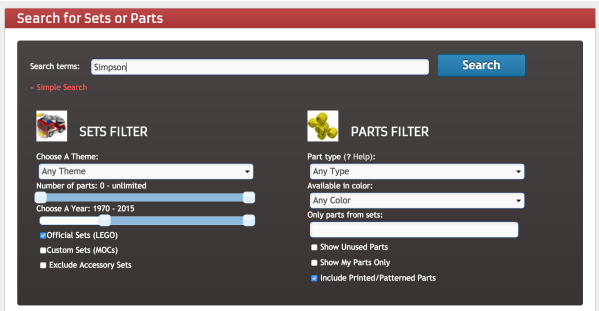


Figure 2.1: LEGO Sets Searching Page of Rebrickable.com

Figure 2.2: A Part Detail Page of Rebrickable.com

2.2 Basic Concepts in LEGO

In this thesis, **LEGO** refers to the brand building bricks as well as the brick itself. **MOC** means My Own Creation, which is any LEGO creation designed and built by a LEGO fan without instructions.

Brick refers to the heart-and-soul of LEGO, the basic brick. They come in many different sizes and colors. They can be attached to other parts with studs on both the top and the bottom of a brick.

Minifigures are the figures most often found in LEGO sets today that have interchangeable hands, hair, legs, heads, torsos, and accessories.

Parts, also called pieces, refer to the small components of a LEGO set, including plates, minifigures, bricks, etc. Parts come with different colors and each color of a part is called an **Element**. Here, we consider parts without caring about colors.

2.3 Tools

In the pursuit of a solution to our problem, we made use of following tools and technology:

2.3.1 Gensim

Gensim[5] is an open-source vector space modeling and topic modeling toolkit, implemented in the Python programming language, using NumPy, SciPy and optionally Cython for performance. It is specifically intended for handling large text collections, using efficient online algorithms. It includes implementations of TF-IDF, latent semantic analysis (LSA) and latent Dirichlet allocation (LDA), including distributed parallel versions. We use LDA to build topic models on part categories of all 2612 LEGO sets. We also use its utility library to compute cosine similarities between the topic distribution of each set.

2.3.2 scikit-learn

scikit-learn[6] is an open source machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. We use its K-Means Clustering function to cluster similar sets in order to understand their themes relationships.

2.3.3 D3.js

D3.js[7] (Data-Driven Documents) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. It makes use of the widely implemented SVG, HTML5, and CSS standards. It allows great control over the final visual result. We use the Circle Packing[8] example to visualize new theme clusters after applying the K-Means algorithm.

2.4 Topic Model

Topic models have been applied to many text mining problems[11][12]. A topic model is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. Each topic is a distribution over a set of words and each document has a distribution over a set of topics. For example, topic "fruit" may be represented as {"apple" 0.06, "banana" 0.04, "watermelon" 0.02, ... } and the document "Apple Macbook Pro On Sale" will have a topic distribution of {"computer" 0.08, "shopping" 0.05, "fruit" 0.001, ... }.

There are many algorithms for building a topic model. We focus on applying the most commonly used algorithm, Latent Dirichlet Allocation (LDA)[11]. LDA is a generative probabilistic model for collections of discrete data, such as text corpora. It is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. In return, each topic is modeled as an infinite mixture over an underlying set of topic probabilities. The topic probabilities provide an explicit representation of a document. The graphical model representation of LDA is shown in Figure 2.3. The outer plate represents documents and the inner plate represents the repeated choice of topics and words within a document.

LDA Algorithm:

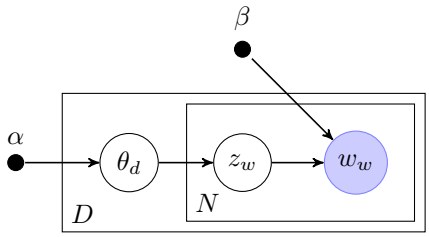


Figure 2.3: Plate Diagram of LDA.

1: **for** document d_d in corpus D **do**
2: Choose $\theta_d \sim$ Dirichlet(α)
3: **for** position w in d_d **do**
4: Choose a topic $z_w \sim$ Multinomial(θ_d)
5: Choose a word w_w from $p(w_w|z_w, \beta)$, a multinomial distribution over words conditioned on the topic and the prior β.
6: **end for**
7: **end for**

This thesis adds to the existing pool of applications a new type of application of topic models, which has not been explored before. That is, it is a novel approach to discover "topics" in LEGO parts. Though LEGO parts do not have meanings as words, they have specific shapes and functionality to support part of a set. In this thesis, we will have each topic as a distribution over a set of part categories and each set has a distribution over a set of topics. We will apply LDA to Section 5.2.2 when building "topics" of part categories.

2.5 K-Means Clustering

After performing Topic Models to the categories of parts of all LEGO sets, we are able to know the topic probability distribution for a set. The topic distributions can be used to compute the similarity matrix among all LEGO sets because when documents are represented as term vectors, the similarity of two documents corresponds to the correlation between vectors.

A similarity matrix is a matrix of scores that represent the similarity between a number of data points. Each element of the similarity matrix contains a measure of similarity between two of the data points. In this thesis, we choose to use the cosine similarity[10] measure. It is one of the most popular similarity measure applied to text documents, such as in information retrieval applications and clustering. An important

property of cosine similarity is its independence of document length. The cosine similarity is defined by:

$$\cos(x, y) = \frac{x \cdot y}{\| x \| \| y \|} \tag{2.1}$$

Now that we have generated a similarity matrix of all LEGO sets, we will apply K-means clustering algorithm[12] to them in order to find sets that are similar to each other. K-means uses distance-based criterion assigning individuals to their nearest population which is located at the population mean. In order to determine K, the algorithm compares the variance within clusters to that between clusters. The goal is to have greater variances between clusters and smaller variances within a cluster.

K-means Clustering Algorithm:

0. Start with initial guesses for cluster centers (centroids)

1. For each data point, find closest cluster center (partitioning step)

2. Replace each centroid by average of data points in its partition

3. Iterate step1 and step2 until convergence

Write $x_i = (x_{i1}, ...x_{ip})$:

If centroids are $m_1, m_2, ...m_k$, and partitions are

$c_1, c_2, ...c_k$, then one can show that K-means converges to a *local* minimum of

$$\sum_{k=1}^{K} \sum_{i \in c_k} ||x_i - m_k||^2 \qquad \text{Euclidean distance}$$

(within cluster sum of squares)

After clustering sets into clusters, we can analyze the relationships between each set's theme.

Chapter 3

Problem Formulation

3.1 Data Set

The LEGO datasets were downloaded from Rebrickable.com[4]. The original datasets were in the form of CSV files. The following shows what information we have in each file.

Sets: [set id, year, num pieces, theme, description] Originally, 10,564 entries.

Set Parts: [set id, part id, quantity, ldraw color id, type (1=normal, 2=spare)] Originally, 513,772 entries.

Parts: [part id, description, category] Originally 22,467 entries.

For the data processing step, we take the three data files as the input and will output a JSON file, "all_sets.json", containing subsets of all LEGO sets. The file is also available in the Github repository for this thesis[9]. To meet the experiment purpose, we select sets created between **2004** and **2014** with more than **10** parts including spare parts. This results in a total of **2612** LEGO sets with **9991** unique parts without considering color and **20,855** parts if regarding each element individually. There are **802,076** total pieces in these LEGO sets. Among these sets, there are **43** unique themes (shown in Section A.2) and **52** unique part categories (shown in Section A.1).

The following contains an example JSON object from the LEGO set data output. This set, named "Atakus", with 8972-1 as its id number has a theme "Bionicle" and was created in 2009. Including the spare parts, there are total 13 parts in this set and the "parts" dictionary shows the distribution of parts (part ids).


```
{"num_pieces": 13,"num_pieces_plus_spare": 13,"parts_with_color": {"64262+57": 1,"60
    896+0": 2,"62386+72": 2,"64263pat0002+0": 2,"60900+0": 2,"60895+72": 1,"64319+0"
    : 1,"64251+72": 2},"theme": "Bionicle", "parts": {"62386": 2,"60900": 2,"64262":
     1,"60895": 1,"60896": 2,"64263pat0002": 2,"64319": 1,"64251": 2},"year": 2009,"
    descrption": "Atakus", "set_id": "8972-1"}
```

3.2 Definition of Data Mining Tasks

As stated in Chapter 1, we focus on the following three tasks.

3.2.1 LEGO Lexicon Construction

The motivation for this task is to help LEGO builders get better understanding of how each part contributes to a set and a theme in order to create ideas about MOCs they can build and make effective purchasing decisions. Additionally, for LEGO designers, they can get insights about how all historical parts they have designed play a role in all sets and themes, in order to create better and powerful parts in the future.

This task takes as input the data of all 2612 LEGO sets, including the theme of a set, what parts appear in the set and the category of each part.

It produces as output a list of dictionaries, which is called the LEGO Part Lexicon. Each dictionary is for each part containing additional information including the number of sets it serves, all themes it serves, the most frequent theme it serves, and the top 10 frequent companioned parts associated with it.

3.2.2 LEGO Construction Patterns Discovery

The motivation for this task is to help LEGO designers when they create a new set or a new theme, what new categories of parts they should design and how they work with historical parts. This task also helps LEGO builders understand LEGO construction patterns by showing what sets of part categories each theme and each set tend to have. They will also see how similar LEGO themes are for considering LEGO sets that share the same types of parts or for finding substitutes of a part in a different set.

This task takes as input the data of all 2612 LEGO sets, including all part categories.

It also produces intermediate results of several "part topics" with the part category and its probability in contribution to the topic, the topic coverage vector of each set and a similarity matrix showing how similar each set is to all other sets based on the topic distribution.

This task finally outputs theme clusters that show what similar themes are in the form of packed circles data visualization.

3.2.3 LEGO Hybrid Theme Recommendation

The motivation for this task is to provide LEGO set recommendations to LEGO builders. For a given set, we recommend sets in another theme that are similar to it based on the categories of parts, which will also help them generate ideas about how to build a new object with the same set of LEGO parts. This task

takes as input the similarity matrix generated in Section 3.2.2 and a similarity threshold (0 to 1) that shows what degree of similarity between two sets. It produces as output a list of pairs of sets that are similar to each other with their similarity score and some possible "hybrid sets" from other themes that could be built. From there, we may able to tell that given a specific set id, what similar sets in other themes are.

Chapter 4

Construction of LEGO Part Lexicon

A lexicon is the vocabulary of a person, language, or branch of knowledge that serve both a catalogue and a system of rules which allow for the combination of small components in the vocabulary into bigger chunks of contexts. Relating this concept to LEGO data sets, our goal is to construct a LEGO Part Lexicon. How does this idea help LEGO designers and builders? On one hand, this helps LEGO builders get better understanding about how each part contributes to a set and a theme in order to create ideas about MOCs they can build and make effective purchasing decisions. Additionally, for LEGO builders, they can get insights about how all historical parts they have designed play a role in all sets and themes, in order to create better and powerful parts in the future. On the other hand, for LEGO designers, they can get insights about how all historical parts they have designed play a role in all sets and themes, in order to create better and powerful parts in the future.

In this section, after formally defining the LEGO Part Lexicon, we focus on two tasks. The first task is to generate a list of pieces pairs that appear in these LEGO sets and is sorted by the frequencies of each pair. Then, the second task is to integrate the piece companions list with all sets data and the piece category dictionary to actually build the part lexicon.

4.1 What is a LEGO Part Lexicon?

In Natural Language Processing, a lexicon[13] is defined as a collection of information about the words of a language about the lexical categories to which they belong. A lexicon is usually structured as a collection of lexical entries, like (“pig” N V ADJ). “pig” is familiar as an N, but also occurs as a verb (“Jane pigged herself on pizza”) and an adjective, in the phrase “pig iron”. In practice, a lexical entry will include further information about the roles the word plays, such as feature information.

Similarly, in order to infer what role each LEGO part plays, we can build a lexicon for LEGO parts that is based on the statistics of each brick. We generate a lexical entry for each piece including the following information:

1. **piece_id**: the id of the part
2. **category**: the category of the part
3. **total**: the total number of sets this part appears
4. **in_sets_percentage**: the total percentage of sets this part appears, which is (total / # of all sets)
5. **in_themes**: a list of themes this piece appears with number of sets in each theme
6. **in_themes_num**: the total number of themes this piece appears
7. **frequent_in_theme**: the most frequent theme(s) this piece appears
8. **10_frequent_companions**[1]: a list of companioned piece_ids with frequency of this pair of parts

This functionality will effectively help LEGO fans and builders get familiar with various pieces. They may get ideas about how popular or rare a part is; what functionality this part would support based on a theme; what companioned parts they need to find or purchase when they design an MOC set.

4.2 Build a LEGO Part Lexicon

We use data mining techniques to generate an entry for each LEGO part. In this section, we focus on two tasks. The first task is to generate a list of pieces pairs that appear in these LEGO sets sorted by the frequencies of each pair. The second task is to integrate the piece companions list with all sets data and the piece category dictionary to actually build the part lexicon.

4.2.1 Generate Piece Companions List

INPUT: A list of JSON objects of all sets

OUTPUT: A sorted list of piece companions pairs

First, we initialize a dictionary D for storing the frequency of pairs of part ids. Then, we read in all LEGO sets as dictionaries. For each set, we get the "parts" dictionary. Since the keys of this dictionary are the part ids. We generate combinations of 2 keys among these keys. For example, set 8972-1 (shown in 3.1) has 13 parts in total, which will generate $\binom{13}{2} = 78$ pairs. Then, we check whether D contains the pair or not. If it does not contain the pair, we sort the pair lexically and store in D with count 1; otherwise, we add one to the count associated with the pair. Finally, after finishing processing all sets, we get a list of all

[1]Only for pieces that appear more than two sets among all sets.

items in the dictionary D. Each item is a tuple with the first element as a tuple containing a sorted pair of two part ids and the second element as a frequency of the pair. For example, (('3023', '4073'), 1334) is part of the list, which indicates that parts with id 3023 and parts with id 4073 appear together in 1334 sets in total.

Now, we are ready to serve this list as an input for the next step.

4.2.2 Build the LEGO Part Lexicon

INPUT: A list of JSON objects of all sets, a dictionary C for referring a piece's category by its id, a sorted list of piece companions pairs

OUTPUT: A list of JSON objects of all parts

First, we initialize a dictionary P for storing each part with its statistics mentioned in Section 4.1. Then, we read in all LEGO sets as dictionaries. For each set, we record its theme and get the "parts" dictionary. For each part id in the "parts" dictionary, we get its category in C and check whether the part is in P or not. If not, we put the part id as a key in P and initialize it value to a dictionary as following:

```
result[p] = {'piece_id': p, 'total': 1,'category': category, 'in_themes': {},'
    in_themes_num':0, '10_frequent_companions': [],'frequent_in_theme': [],'
    in_sets_percentage': "" }
```

p is the part id, category is the part category returned from C. Otherwise, we update the dictionary above to increase the value of 'total' by 1. We also calculate the 'in_sets_percentage' using the value of 'total' divide by the total number of sets. Next, we check whether the theme is in 'in_themes' dictionary or not. If not, we put the theme with count 1 in to the dictionary; otherwise, we increase the value by 1. In each iteration, we also update the 'in_themes_num' by assigning length of the 'in_themes' list to it. As for the 'frequent_in_theme' field, we find the maximum value among all values of the 'in_themes' dictionary. Then, we get all keys that associate with this value in it. Finally, we assign all keys to the 'frequent_in_theme' list.

Once we finish all the other 7 fields besides the '10_frequent_companions' field, we can now read in the sorted list generated in the last step. For each piece object in P, we first check whether it appears more than 2 LEGO sets because this makes more sense, i.e. if its 'total' is less than 3, we skip to the next piece. If it appears in more than 2 LEGO sets, we read in the sorted list of piece companions pairs. For each tuple, we get its first element as the pair and the second element as the frequency. Then, only when the '10_frequent_companions' has less than 10 elements do we check whether the current piece is in the pair. If it exists in the pair, we generate a new tuple with its companion and the frequency. And we append this new tuple to the '10_frequent_companions' of the current part.

Now, we have finished building our novel LEGO Part Lexicon. In the next section, we will show some good and bad sample results from the lexicon.

4.3 Sample Results

Good Results:

The following contains an entry returned from our LEGO Part Lexicon for part id 55976, which is in Figure 4.1a. It appears most frequently in "Technic" theme and appears in 0.8422% sets (a pretty rare part). Based on the piece ids of its 10 frequent companion parts, we find their images and information through Rebrickable.com[4]. Here, we only display the 7 frequent companion parts in Figure 4.1. This is a high quality result because this part is frequently used in car models or technic objects and all top companioned parts are very frequently used with this part.


```
{"category": "Wheels and Tyres", "frequent_in_theme": ["Technic"], "
    in_sets_percentage": "0.842266462481%", "10_frequent_companions": [["56145", 22]
    ,["3713", 21],["2780", 21],["43093", 19],["3705", 18],["32123b", 18],["32523", 1
    8],["32062", 17],["4519", 17],["4073", 17]],"in_themes_num": 8,"total": 22,"
    piece_id": "55976", "in_themes": {"Town": 3,"Legends of Chima": 1,"Creator": 2,"
    Racers": 1,"Ninjago": 1,"Technic": 11,"Educational and Dacta": 1,"Super Heroes":
     2}}
```


The following contains an entry returned from our LEGO Part Lexicon for part id 3961, which is in Figure 4.2a. It appears most frequently in "Star Wars" theme and appears in 1.0337% sets. Based on the piece ids of its 10 frequent companion parts, we find their images and information through Rebrickable.com[4]. Here, we only display the 3 frequent companion parts in Figure 4.2. Though the top companioned parts are all frequent parts, i.e. "stop parts" (mentioned in Section 5.1), this result still makes sense because part 4073 can be put on top of it and it is always put on top of part 3020 and 3022.


```
{"category": "Plates Round and Dishes", "frequent_in_theme": ["Star Wars"], "
    in_sets_percentage": "1.0336906585%", "10_frequent_companions": [["4073", 27],["
    3020", 26],["3022", 26],["4032a", 26],["3666", 25],["3023", 25],["3710", 24],["3
    021", 24],["3068b", 23],["3062b", 22]],"in_themes_num": 10,"total": 27,"piece_id
    ": "3961", "in_themes": {"Star Wars": 12,"Town": 2,"Legends of Chima": 1,"
    Sculptures": 3,"Creator": 1,"Cars": 1,"Toy Story": 1,"SpongeBob SquarePants": 4,
    "Ninjago": 1,"Educational and Dacta": 1}}
```

These good entries in general can help a LEGO beginner builder understand why a part is shaped in its way based on the places where they are used and other pieces that are used together, which will also help them decide which parts to buy.

Bad Results:

The following contains an entry returned from our LEGO Part Lexicon for part id 60169, which is in Figure 4.3a. It appears most frequently in "Town" theme and appears in 2.7565% sets. Based on the piece ids of its 10 frequent companion parts, we find their images and information through Rebrickable.com[4]. Here, we only display the 3 frequent companion parts in Figure 4.3. This result is bad because we could barely infer how the part 60169 would work together with the other three frequent companioned parts. In fact, this part, "Minifig Chain" is used mostly on minifigures inside a LEGO set with a lot of pieces. In the sets that have a lot of parts, we tend to see more of the "stop parts" and a small part such as this one plays an insignificant role in being associated with other insignificant parts.


```
{"category": "String, Bands and Reels", "frequent_in_theme": ["Town"], "
    in_sets_percentage": "2.75650842266", "10_frequent_companions": [["3020", 64],["
    4073", 63],["3023", 61],["3795", 57],["3710", 56],["54200", 54],["3003", 53],["3
    022", 53],["3001", 53],["3004", 51]],"in_themes_num": 24,"total": 72,"piece_id":
     "60169", "in_themes": {"Legends of Chima": 2,"Space": 3,"Indiana Jones": 1,"
    Racers": 2,"Teenage Mutant Ninja Turtles": 2,"Cars": 1,"The LEGO Movie": 2,"
    Technic": 1,"Educational and Dacta": 3,"Town": 12,"Hero Factory": 2,"Monster
    Fighters": 3,"Ninjago": 7,"The Hobbit and Lord of the Rings": 3,"Bionicle": 2,"
    Castle": 7,"Super Heroes": 2,"Star Wars": 2,"Power Miners": 3,"Pirates": 2,"
    Creator": 3,"Pirates of the Caribbean": 3,"Agents": 2,"Master Building Academy":
     2}}
```

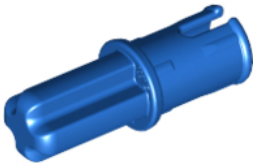

In conclusion, the above good and bad results are useful in general when a LEGO fan wants to sort and store frequent used similar parts together as well as when they build an MOC. But the "stop parts" that appear extremely frequently among all sets are really annoying because they do not display unique explicit functionality and serve as a "generalist" instead of a "specialist". This may efface the functionality of a rare part when generating its frequent companioned part list. We may need to build the lexicon based on the parts category and utilize that information when filtering out "stop parts" in their companioned parts list. We should also consider separately for the parts in minifigures.

(a) 55976. Tyre 56 x 26 Balloon

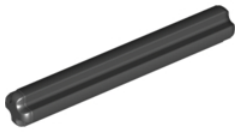
(b) 56145. Wheel 30.4 x 20 without Pinholes, with Reinforced Rim

(c) 3713. Technic Bush

(d) 2780. Technic Pin with Friction Ridges Lengthwise and Center Slots



(e) 43093.Technic Axle Pin with Friction Ridges Lengthwise

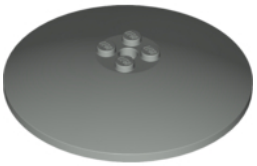


(f) 3705. Technic Axle 4

(g) 32123b. Technic Bush 1/2 Smooth with Axle Hole Semi-Reduced

(h) 32523. Technic Beam 1 x 3 Thick

Figure 4.1: Pictures of Part 55976 and Its Top Seven Frequent Companioned Parts



(a) 3961. Dish 8 x 8 Inverted [Radar]

(b) 4073. Plate Round 1 x 1 with Solid Stud

(c) 3020. Plate 2 x 4

(d) 3022. Plate 2 x 2

Figure 4.2: Pictures of Part 3961 and Its Top Three Frequent Companioned Parts

(a) 60169. Minifig Chain

(b) 3020. Plate 2 x 4

(c) 4073. Plate Round 1 x 1 with Solid Stud

(d) 3023. Plate 1 x 2

Figure 4.3: Pictures of Part 60169 and Its Top Three Frequent Companioned Parts

Chapter 5

LEGO Construction Patterns Discovery

Our next goal is to discover patterns and clusters in LEGO constructions. Why might we be interested in discovering such clusters? This will benefit both LEGO designers and LEGO builders/fans. On one hand, this helps LEGO designers to decide what new categories of parts they should create, what new parts of in old categories should design and how these potential parts work with historical parts. Not only does this benefit for the process of designing a new LEGO part, but also a new LEGO set and a new theme. On the other hand, this helps LEGO builders understand construction patterns by showing what sets of part categories each theme and each set tend to have. They will also see how similar LEGO themes are for considering LEGO sets that share the same types of parts or for finding substitutes of a part in a different set. So, this pattern discovery study is both novel and influential.

In this section, we will look into categories of LEGO parts (shown in Section A.1) and based on all LEGO sets, what "topics" these part categories generate. Then, we utilize the distribution of each LEGO set over the topics to compute the "part topic" of each theme. We will also cluster LEGO sets based on their "part topic" distribution, which further provides insights of how the existing themes are similar to each other. We focus on three tasks. The first task is to do some data preparation, removing "stop parts" in a LEGO set, like "stop words" in a word document. The second task is to build topic models using LDA (discussed in Section 2.4) for part categories. We can then generate both the topic coverage distribution of a LEGO set and a LEGO theme. The third task is to use K-Means Clustering Algorithm (discussed in Section 2.5) to cluster LEGO themes and visualize the similarities between them.

5.1 Find and Generate "Stop Parts" List

In Natural Language Processing, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called **Stop Words**. For example, for the phrase "Chairman of the company", it contains 2 stop words, "of" and "the". Stop words are usually filtered out before processing of natural language data. A stop

(a) 4073. Plate Round 1 x 1 with Solid Stud

(b) 3710. Plate 1 x 4

(c) 3023. Plate 1 x 2

Figure 5.1: Pictures of Top "Stop Parts"

(word) list significantly reduces the number of postings that a system has to store. And the general way for determining a stop list is to sort the terms by collection frequency (the total number of times each term appears in the document collection). Then, we manually filter out words based on whether their semantic content relative to the domain of the documents being indexed, to form the stop list.

Similarly, we treat LEGO parts as words and treat extremely frequent parts according to some threshold as **"Stop Parts"**. The **Threshold** is a float number between 0 and 1 indicating the percentage a part appears in all sets. For example, if we have 0.4 as the threshold, all parts that appear more than $2612 * 0.4 = 1044.8$ sets will be added to the list. The procedure of generating such list is discussed in the following.

In Section 4.2, we have already built a LEGO Part Lexicon. It gives statistics about each part, including how many sets each part appears in. So, we simply go over the lexicon and select all parts that have the 'in_sets_percentage' field that is greater than the *threshold*.

In this thesis, we have tried different threshold values. When it is set to 0.6, we only get 2 parts in the list, parts with id 3023 and 4073. So, we adjust it to 0.5 and get 5 parts in the list. However, it does not seem to include all the top frequent parts. Then, we adjust it to 0.3 and get a much bigger list of 23 frequent parts, of which some already shows some design patterns that we do not want to exclude. Finally, we set the threshold to **0.4** and the following shows the "stop parts" list, which is a good set of 10 frequent parts that do not infer much potential construction patterns. We display 3 pieces that belong to the list inn Figure 5.1. This heuristic makes sense because all these 3 parts are basic small parts that can be used almost in any LEGO set and do not infer much information about a theme or a design of a set.

{'4073': 1628,'2412b': 1046,'3710': 1347,'3795': 1160,'3023': 1579,'3022': 1332,'54
200': 1139,'3020': 1393,'3004': 1136,'3021': 1155}

5.2 Build Topic Models for LEGO Part Categories

Discussed in Section 2.4, applying Topic Models to LEGO parts data is an innovative research study. In this section, we aim to build topic models using LDA for part categories, which will help generate both the topic coverage distribution of a LEGO set and a LEGO theme. We utilize the Gensim topic modeling toolkit mentioned in Section 2.3.1. The detailed procedures are illustrated in the subsections below.

5.2.1 Construct a Document-term Matrix

To generate an LDA model, we need to understand how frequently each term occurs within each document. To do that, we need to construct a document-term matrix. In fact, we construct a "LEGO set-Part category" matrix with a dimension of $2612 * 52$ according to our dataset. 2612 is the total number of LEGO sets and 52 is the total number of part categories.

INPUT: A list of JSON objects of all sets, a list of "stop parts", a dictionary C for referring a piece's category by its id

OUTPUT: corpus, in the form of bag-of-words

Our first task is to know what categories of parts cover each LEGO set. For example, if we have a LEGO set containing 5 pieces, 2 of them belong to 'Beams' category, 2 of them belong to 'Gears' category and 1 of them belongs to 'Pins' category, we will return ['Beams', 'Beams', 'Gears', 'Gears', 'Pins'] for this specific set.

First, we initialize an empty list L that will be filled with the part category list of each LEGO set. Then, we read in all LEGO sets as dictionaries. For each set, we initialize its only empty list l for storing part categories and get the "parts" dictionary. For each part in the "parts" dictionary, only if it does not a "stop part" should we retrieve its category from C and append to l. After processing all parts in the set, we append l to the global list L. So eventually, L has 2612 small lists containing repeated part categories.

Next, we traverse L to construct a dictionary, assigning a unique integer id to each unique part category while also collecting frequency counts and relevant statistics. Our dictionary is a mapping between part categories and their integer ids. After this step, we convert our dictionary to a bag-of-words. The result, corpus, is a list of vectors equal to the number of documents. In each document vector is a series of tuples. The following two lines of codes will do the work.

```
dictionary = corpora.Dictionary(L)
corpus = [dictionary.doc2bow(l) for l in L]
```

For example, if we have a set whose part category list is ['Beams', 'Gears', 'Pins', 'Gears'] and another set

as ['Beams', 'Brick']. We may have a dictionary of 'Beams': 0, 'Gears':1, 'Brick': 2, 'Pins': 3. And for first set, the resulted bag-of-words is [(0, 1), (1, 2), (3, 1)].

5.2.2 Build the LDA Model

Once we have the corpus ready, we could generate an LDA model. We use the following two lines to build the model and check the topics. The 'num_topics' parameter is how many topics should be generated. In this thesis, I have tried 5, 6 and 7 topics. But the topics returned with **6** as the parameter appear to make most sense. The 'passes' parameter is the number of laps the model will take through corpus. The greater the number of passes, the more accurate the model will be. However, too many passes can be slow on a very large corpus. So, I have chosen to run **20** passes, which also help generate reasonable results.

```
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics=6, id2word = dictionary,
    passes=20)
topics = ldamodel.print_topics(num_topics=6)
```

After building the LDA model, we can generate both the topic coverage distribution of a LEGO set and a LEGO theme. In the next section, we will show some sample results.

5.2.3 Sample Results

Based on the parameters discussed in last section, we get the following 6 randomly selected topics. Within each topic, there are 10 most contributing part categories with their contribution probability (served as weight) appearing in that topic. The **higher** the probability is, the **more** the part category contributes to the topic.

Topic #0: 0.825*Bricks + 0.129*Bricks Sloped + 0.012*Windows and Doors + 0.008*Bricks Printed + 0.005*Bricks Round and Cones + 0.004*Wheels and Tyres + 0.003*Plates + 0.003*Bricks Curved + 0.003*Baseplates + 0.002*Bricks Special

Topic #1: 0.919*Duplo, Quatro and Primo + 0.056*Tubes and Hoses + 0.009*Other + 0.006*Pneumatics + 0.003*Windows and Doors + 0.002*Baseplates + 0.002*Supports, Girders and Cranes + 0.002*Non-LEGO + 0.001*Plants and Animals + 0.000*Flags, Signs, Plastics and Cloth

Topic #2: 0.140*Plates Special + 0.104*Bricks Sloped + 0.077*Technic Bricks + 0.068*Plates Angled + 0.066*Hinges, Arms and Turntables + 0.064*Pins + 0.063*Plates + 0.052*Bricks Curved + 0.045*Plates Round and Dishes + 0.045*Tiles

Topic #3: 0.303*Pins + 0.157*Beams + 0.153*Technic Connectors + 0.137*Axles + 0.082*Bionicle,

Set id	Theme	Description	Topic Coverage Vectors
42026-1	Technic	Black Champion Racer	[(3, 0.994)]

Table 5.1: A Good Result of LEGO Set with Topic Coverage Distribution

Hero Factory and Constraction + 0.055*Bushes + 0.027*Gears + 0.017*Technic Special + 0.017*Wheels and Tyres + 0.012*Technic Steering, Suspension and Engine

Topic #4: 0.152*Minifig Accessories + 0.141*Minifigs + 0.096*Bricks Round and Cones + 0.092*Plants and Animals + 0.077*Bricks Sloped + 0.067*Bricks Special + 0.054*Bars, Ladders and Fences + 0.032*Plates Special + 0.031*Bricks Curved + 0.030*Bricks

Topic #5: 0.236*Plates + 0.202*Tiles + 0.119*Plates Special + 0.076*Bricks Special + 0.065*Wheels and Tyres + 0.040*Bricks + 0.038*Bricks Curved + 0.031*Bricks Sloped + 0.030*Panels + 0.029*Windows and Doors

We find the model to be reasonable due to the following findings and insights:

1) The most meaningful topic is **Topic #3**. Based on categories like "Technic Steering, Suspension and Engine", "Wheels and Tyres" and "Beams", we can easily infer that car models and models with technical designs will fall into this topic.

2) Another meaningful topic is **Topic #5**. In this topic, a lot of types of Bricks and Plates have been mentioned, which both contribute to build a house or a landmark. These buildings need a lot of tiles to fill the floors and have windows and doors.

3) Some of the sets that have houses as the main model usually come with several minifigures and vehicles. **Topic #0** helps cover this information.

4) **Topic #2** best describe a model that is a combination of building component and vehicle component. But the vehicle component weighs more than the other one.

5) **Topic #1** covers unique sets that are old, in rare themes or in small pieces.

6) **Topic #4** covers Minifigure sets and and building sets that has detailed designs.

Good results:

A good sample result is shown in Table 5.1. The set is with id 42026-1 (shown in Figure 5.2a) and is described as "Black Champion Racer". This LEGO set belongs to the "Technic" theme. According to its topic coverage distribution, the **Topics #3** makes perfect sense for a car model.

Set id	Theme	Description	Topic Coverage Vectors
7598-1	Toy Story	Pizza Planet Truck Rescue	[(5, 0.456), (2, 0.235), (4, 0.170), (3, 0.073), (0, 0.065)]

Table 5.2: A Bad Result of LEGO Set with Topic Coverage Distribution

(a) 42026-1.Technic & Black Champion Racer (b) 7598-1.Pizza Planet Truck Rescue

Figure 5.2: Pictures of Two LEGO Sets in Sample Results.

Bad results:

A bad sample result is shown in Table 5.2. The set is with id 7598-1 (shown in Figure 5.2b) and is described as "Pizza Planet Truck Rescue". This LEGO set belongs to the "Toy Story" theme. According to its topic coverage distribution, **Topics #5**, **Topics #2** and **Topics #4** all cover this set, which shows ambiguity of the categories of parts that contribute to the set. As we can see from its cover image, both minifigures and a car model play important role in expressing this set. "Toy Story" is also a movie that LEGO creates peripheral sets of a movie to attract fans of that movie. The peripheral sets could construct a movie scene with minifigures as the main characters or a landmark in a specific scene. As a theme, "Toy Story" is more volatile than characteristic themes such as "Technic" and "Duplo" (designed with twice bigger bricks for children aged 1 to 5 years). This explains why multiple topics all contribute to this single set and why this bad result is generated.

Additionally, we also map themes to the "topics" of part categories. We first define some notations below before illustrating the algorithm of how to compute the "topic coverage vector" for each of the LEGO themes.

Notations

1. s_i: a LEGO set
2. $S = \{s_1, s_2, ..., s_n\}$: a list of LEGO sets
3. $P = \{p_1, p_2, ..., p_k\}$: a list of topics of part categorites

4. t_i: the theme of a LEGO set

5. θ_i: the topic distribution for a LEGO set

6. θ_{ik}: the probability of topic p_k that covers a LEGO set

7. $T = \{t_1 :\{i, ..., j\},\ t_2 :\{m, ...n\},\ ...,\ t_t :\{x, ..., y\}\}$: a theme dictionary with each theme as each key and a list of indices of LEGO sets in S that belong to the theme as the corresponding value

Theme-based Topic Coverage Computing Algorithm:

for theme t_i in dictionary T **do**
  Get a list of set indices I
  Create an empty list D for appending all distributions of the current theme t_i
  for index i in I **do**
    Generate the topic distribution θ_i from the LDA model for s_i
    Append θ_i to list D
  end for
  Convert D to a dictionary X with p_k as the key and sum of θ_{ik} as the value
  for p_i in X **do**
    Divide $X[p_i]$ by the length of the index list I, $X[p_i] = X[p_i]$ / $|I|$
  end for
  Convert X back to a list D with tuples of the topic and its probability
  Print D for the current theme t_i
end for

According to the above algorithm, we basically take a sum of the topic coverage vectors of all the sets in a theme and compute an average topic coverage vector for the theme. The results shown in Table 5.3 look reasonable in general. Some characteristic themes are covered thoroughly by one "expressive" topic of part categories. We then use this result to help further re-cluster similar LEGO themes.

5.3 Cluster and Visualize Similar Themes

In this section, we use K-Means Clustering Algorithm (discussed in Section 2.3.2) to cluster LEGO themes and visualize the similarities between them. To achieve this goal, we focus on three tasks. The first task is to generate the similarity matrix of LEGO sets. The second task is to run K-Means algorithm to get clusters of sets. Then, we map each theme to the clusters. The third task is to visualize the results.

Theme	Topic Coverage Vectors
Creator	[(0, 0.399), (5, 0.29), (2, 0.225), (4, 0.057), (3, 0.022), (1, 0.001)]
Duplo	[(1, 0.973), (5, 0.003), (4, 0.003), (2, 0.001), (0, 0.001), (3, 0.001)]
Mixels	[(2, 0.803), (4, 0.157), (5, 0.013), (0, 0.011)]
Agents	[(2, 0.4528), (3, 0.174), (5, 0.167), (4, 0.167), (0, 0.034)]
Master Building Academy	[(2, 0.343), (5, 0.228), (4, 0.214), (0, 0.117), (3, 0.070), (1, 0.021)]
SpongeBob SquarePants	[(2, 0.32), (4, 0.287), (5, 0.252), (0, 0.119), (3, 0.021)]
Technic	[(3, 0.976), (2, 0.012), (5, 0.007), (4, 0.0)]
Bionicle	[(3, 0.934), (4, 0.026), (2, 0.014), (0, 0.006), (5, 0.003), (1, 0.001)]
4 Juniors	[(4, 0.726), (2, 0.089), (0, 0.084), (5, 0.082), (3, 0.009), (1, 0.002)]
Pirates of the Caribbean	[(4, 0.557), (2, 0.288), (5, 0.067), (0, 0.046), (3, 0.034)]
Architecture	[(5, 0.742), (0, 0.129), (4, 0.05), (2, 0.05), (3, 0.025), (1, 0.001)]
Town	[(5, 0.458), (4, 0.279), (2, 0.175), (0, 0.049), (3, 0.031), (1, 0.0)]
Sculptures	[(5, 0.691), (2, 0.174), (0, 0.077), (3, 0.034), (4, 0.024)]

Table 5.3: LEGO Themes with Topic Coverage Distribution

5.3.1 Generate the Similarity Matrix

In Section 5.2.1, we build our corpus and our LDA topic model of part categories. Now, we will combine them to convert the corpus into LDA space. Then, we can use the transformed corpus to compute the similarity matrix of LEGO sets. The similarity measure used is cosine between two vectors, i.e. the cosine similarity. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1]. The more closer the value is to 1, the more similar the two vectors are. The diagonal has 1's because each set is absolutely 100% similar to itself.

For performance benefits, we store the matrix as a dense numpy array. The following codes work efficiently for generating the matrix.

```
corpus_lda = ldamodel[corpus]
index = similarities.MatrixSimilarity(corpus_lda)
l = []
for s in index:
    l.append(s)
    new_dense = np.asarray(l)
```

Based on the 2612 LEGO sets and the 6 topics as features, we get the a dimension of $2612 * 2612$ similarity matrix below:

```
[[ 1.  0.58829916 0.01353808 ..., 0.01376874 0.9829101
   0.01353808]
 [ 0.58829916 1.         0.13813192 ..., 0.14794694 0.4338336
```

```
  0.13813192]
 [ 0.01353808 0.13813192 1.       ..., 0.         0.         1.       ]
 ...,
 [ 0.01376874 0.14794694 0.       ..., 1.         0.         0.       ]
 [ 0.9829101  0.4338336  0.       ..., 0.         1.         0.       ]
 [ 0.01353808 0.13813192 1.       ..., 0.         0.         1.       ]]
```

5.3.2 K-Means Clustering for LEGO sets

Now that we have the dense similarity matrix ready, we can utilize the scikit-learn library mentioned in Section 2.3.2 for applying the K-means clustering algorithm. According to the codes below, the 'n_clusters' parameter is the number of clusters to form as well as the number of centroids to generate. Since there are **43** themes among our dataset, I have chosen to generate the same number of clusters among all LEGO sets. The 'init' parameter is for setting the method for initialization, defaults to 'k-means++', which selects initial cluster centers for K-means clustering in a smart way to speed up convergence. The 'max_iter' parameter is the maximum number of iternations of the K-means algorithm for a single run. Typically, the algorithm converges in less than 30 iterations for our LEGO dataset. The 'fit' function computes the K-means clustering and assign a label to each LEGO set indicating which cluster it belongs to.

```
km = KMeans(n_clusters=43, init='k-means++', max_iter=100, n_init=1, verbose=1)
labels = km.fit(new_dense).labels_
cluster_list = list(labels)
```

After separating the 2612 LEGO sets into 43 clusters, we map the theme of each set to each cluster in order to see which themes fall into the same cluster. The following contains the sample results:

```
{0: ['Exo-Force', 'Castle', 'Toy Story', 'Agents'], 1:['Friends'], 2:['Sports', '
    Technic', 'Educational and Dacta', 'Hero Factory', 'Bionicle'], 4:['Duplo'], 5:[
    'Space', 'Super Heroes', 'Master Building Academy'], 7:['Bulk Bricks', 'Creator'
    ], 8:['Legends of Chima'], 9:['Sculptures'], 10:['Harry Potter', 'Belville'], 11
    :['Harry Potter', 'Indiana Jones', 'SpongeBob SquarePants'], 13:['Modular
    Buildings', 'Racers', 'Cars', 'Architecture', 'Minecraft'], 15:['Agents'], 16:['
    X-Pod', 'Mixels', 'Star Wars', 'Designer Sets'], 19:['Juniors'], 20:['Ninjago',
    'Pirates', '4 Juniors'], 21:['Monster Fighters', 'The Hobbit and Lord of the
    Rings'], 24:['Teenage Mutant Ninja Turtles'], 25:['Town'], 27:['The LEGO Movie',
```

'Master Building Academy'], 28:['Toy Story'], 31:['Pirates of the Caribbean'], 32:['Indiana Jones'], 37:['The Hobbit and Lord of the Rings', 'Pirates of the Caribbean'], 40:['Power Miners', 'Master Building Academy'], 41:['Atlantis'], 42 :['Agents']}

5.3.3 Visualization of LEGO Theme Clusters

In this section, we utilize the d3.js library mentioned in Section 2.3.3 for visualizing the clusters of LEGO themes based on the LDA topic model of part categories and the K-Means clustering of LEGO sets. Based on the sample results we get from the last section, we can draw packed circles in Figure 5.3 to show the similarities between different LEGO themes.

This visualization is useful because we get an idea about how unique a theme is and what themes can be potentially combined together to collaboratively use parts together. Especially for sets that belong to some movie themes, since typically they use more unique parts to display movie details, it is interesting to infer that how they use similar parts to express two different movie scenes and whether it is needed to create new types of parts every time when a new movie theme is introduced. The result also makes sense. For example, it is shown that in Table 6.1, "Racers" theme and "Architecture" themes fall into the same circle (cluster), we find similar sets from them.

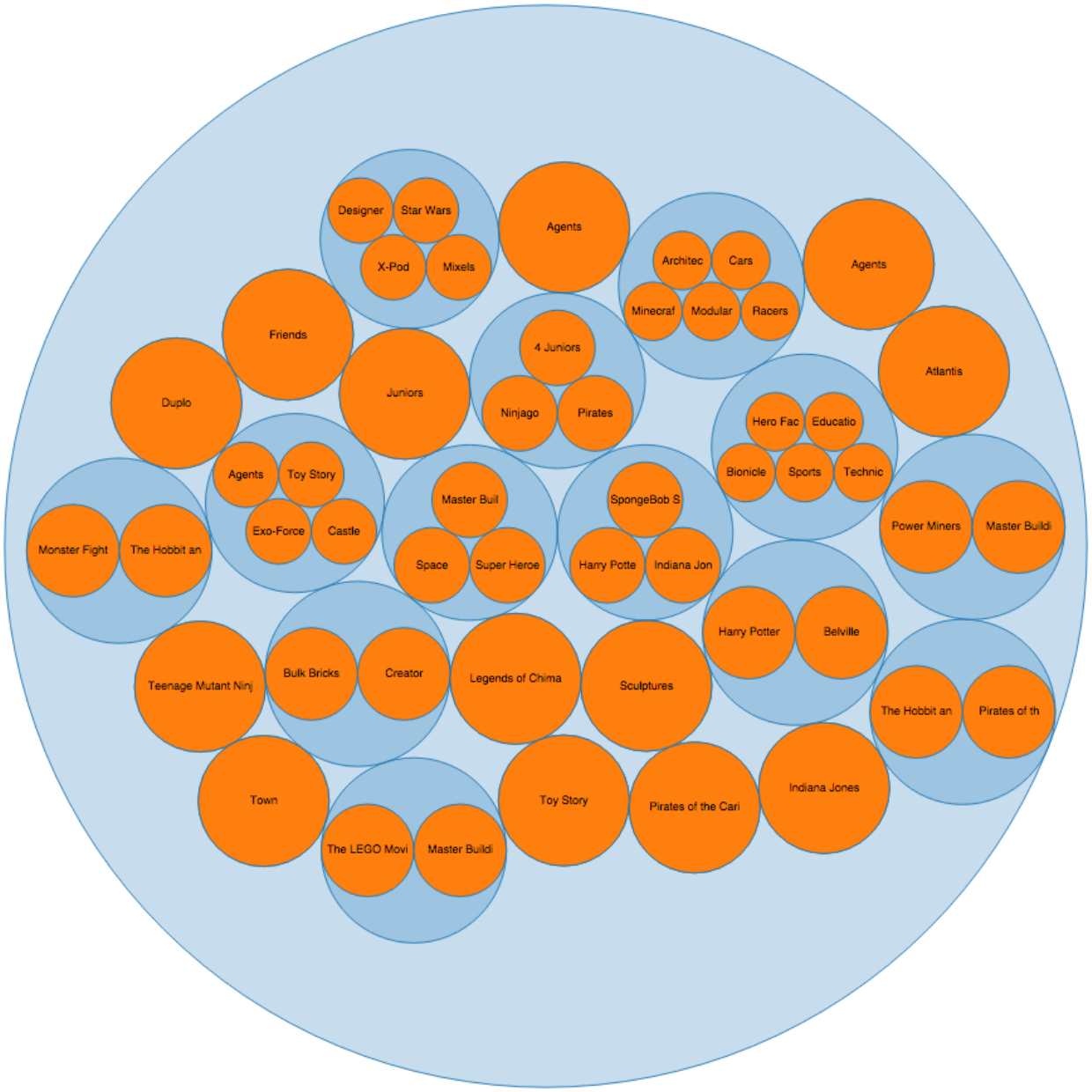


Figure 5.3: Data Visualization of the 43 LEGO Themes Clusters Generated by LDA Topic Model and K-Means Algorithm

Chapter 6

Hybrid Theme Recommendation

In this section, we explicitly look into the similarity matrix generated in Section 5.3.1 in order to find similar themes that fall into different LEGO themes. Following this direction, we may create hybrid theme and set recommendations to LEGO fans and potential LEGO customers. Our goal is towards improving their LEGO purchasing decision making experience as well as MOC ideas generating processes.

6.1 Select Extreme Similar LEGO Sets

According to the similarity matrix, we set a threshold of **0.98** and generate all sets that are similar to each other in the aspect of distributions over the "part category topics". That is, due to the symmetric property of this matrix, we traverse the upper triangular matrix and check whether each cell has a value that is greater or equal to 0.98. If so, we record the indices of the two LEGO sets and find their information based on their positions in the list of LEGO set JSON objects. This threshold produces 105,423 similar pairs of LEGO sets.

6.2 Sample Results

For the LEGO set 10220-1, Volkswagen T1 Camper Van, we find the following similar sets from different themes. This set is from the "Sculptures" theme and has 1355 pieces in total including spare parts. Table

Set id	Theme	Description	Number of Pieces	Similarity
8169-1	Racers	Lamborghini Gallardo LP 560-4	751	0.99769
21006-1	Architecture	The White House	567	0.992161
6743-1	Creator	Street Speeder	174	0.990474
30240-1	Star Wars	Z-95 Headhunter	61	0.989679
10242-1	Creator	Mini Cooper	1136	0.98612
7597-1	Toy Story	Western Train Chase	579	0.980877
7208-1	Town	Fire Station	673	0.980849

Table 6.1: Seven Similar LEGO Sets to 10220-1 From Different LEGO Themes

(a) 10220-1.Volkswagen T1 Camper Van

(b) 8169-1.Lamborghini Gallardo LP 560-4

(c) 21006-1.The White House

(d) 6743-1.Street Speeder

(e) 30240-1.Z-95 Headhunter

(f) 10242-1.Mini Cooper

(g) 7597-1.Western Train Chase

(h) 7208-1.Fire Station

Figure 6.1: Pictures of Set 10220-1 and Its Top Seven Similar Sets From Different Themes

6.1 shows the top 7 LEGO sets that are most similar to the set 10220-1. Figure 6.1 shows all eight LEGO sets cover photos.

We define a **"Bridge set"** to be the similar set we discover for a set in another LEGO theme. If we identify multiple bridge sets between themes, they should help a player create a hybrid theme combining the two. According to our sample results, set 6743-1 (shown in 6.1d) and set 10242-1 (shown in 6.1f) are two "Bridge sets" in the "Creator" theme of this current set , 10220-1. We may create recommendations about combining set 10220-1 and 6743-1 or set 10220-1 and 10242-1 to potentially build set 21006-1 that belongs to the "Architecture" theme. We may also recommend combining 21006-1 with 7597-1 to build 7208-1 in "Town" theme.

The idea is that for sets in similar themes, they share some overlapping characteristics of parts as well as their own unique parts that express individual theme. We may do further experiments and research in how to combine those unique theme-based parts in a friendly way in the future.

Chapter 7

Conclusion and Future Works

The LEGO Group designs LEGO parts and sets. LEGO fans and builders build sets follow the instructions or purely start from scratch (a pile of LEGO parts) to create MOCs. LEGO designers need to create more interesting parts and sets in order to attract more customers; LEGO fans would like to have a systematic understanding about LEGO parts in order to create more MOC design ideas. However, there is barely research in discovering knowledge or insights from the LEGO data sets. Our goal is to build a LEGO design support system that both benefit to the designers and fans and open up a new application direction in data mining of mapping knowledge discovery in toy patterns to future designs.

In this thesis, we proposed a systematic method of doing data mining of LEGO sets and parts data. It is also a completely novel way to treat each LEGO set as a document and each part in it as a word. We focused on three main data mining tasks: 1) automatically constructing a LEGO Part Lexicon that contains statistics of a LEGO part, showing its relationship with other parts, sets, and themes; 2) discovering LEGO construction patterns in order to find a subset of part categories ("part topics") and group similar themes into clusters based on the part categories each set covers; 3) building a hybrid theme recommendation framework based on similarities of sets in different themes.

For task 1), we first defined our LEGO Part Lexicon to be a list of dictionaries, of each is a dictionary for a specific part containing additional information including the number of sets it serves, all themes it serves, the most frequent theme it serves, and the top 10 frequent companioned parts associated with it. Then, we built the lexicon integrated with the piece companions list, which is a sorted list of piece companions pairs. Our algorithm worked well for unique parts that appeared occasionally in all sets. But for frequent parts of basic brick types, i.e. "stop part", the most frequent companioned part list did not provide useful insights. So, for future works, we may build the lexicon based on the parts category and utilize that information when filtering out "stop parts" in their companioned parts list. We should also consider separately for the parts in minifigures. Another possible research findings is to include the information of when the part was created and how it has contributed to sets and themes through years. In this direction, we could predict that when a new movie coming out, which parts would be put in a new set that builds a scene of that movie.

For task 2), we started with finding and creating the "stop parts" list that contains extremely frequent parts based on some threshold. Then, we proposed a novel approach by building topic models on part categories for all sets data. We started with constructing a "LEGO set-Part category" matrix that shows how frequently each part occurs in each set. Then, we built LDA model on part categories for all sets data. From there, we generated a few topics and for each set and each theme, we got a distribution over these topics. We could infer which subset of part categories used frequently to build sets in a theme. Next, we clustered similar themes by applying K-Means Clustering algorithm to the similarity matrix of sets and followed by mapping those clusters to the themes of these LEGO sets. Finally, we visualized the clustering result in the form of packed circles. The "stop parts" list helped just as the same way a stop words list served. But for future work, we could try varying the set of parts in the "stop parts". That is, one possible direction is to study the actual size and dimension of a part and see whether removing all small parts also makes sense. Additionally, we may also look into the years they create and see how clusters of themes change along years. If we include the price of each set with their age group, we could probably come up with some more innovative clustering algorithms, such as clustering sets and themes that have similar "values (worth of the price)" or "educational purposes (based on age groups)".

For task 3), we utilized the similarity matrix generated in task 2) and selected extreme similar LEGO set pairs based on some threshold. We manually evaluated the sample results and gave did some possible combinations of sets with different themes that could probably generate another similar set. For future works, since for sets that are in similar themes, they tend to share at least overlapping characteristics of parts (categories) but still have a set of unique theme-based parts. we may do further experiments and research in how to combine those parts in a systematic way in order to know what possible themes or ideas the "hybrid parts set" would generate. Moreover, we could also study if the similarities between different themes or different sets relate to the price of sets or the aimed age group of these sets.

With the methods discussed in this thesis, we take an effective first step forward towards opening a new application direction in data mining for toys data, particularly LEGO data sets that will help with LEGO future design for parts, sets and themes. We strongly believe that these applications will improve LEGO fans brick building experience, help them make purchasing decisions effectively and eventually make them happy, which is the most important.

Appendix A: Data

A.1 Part Categories

'Axles', 'Bars, Ladders and Fences', 'Baseplates', 'Beams', 'Beams Special', 'Belville, Scala and Fabuland', 'Bionicle, Hero Factory and Constraction', 'Bricks', 'Bricks Curved', 'Bricks Printed', 'Bricks Round and Cones', 'Bricks Sloped', 'Bricks Special', 'Bricks Wedged', 'Bushes', 'Containers', 'Duplo, Quatro and Primo', 'Flags, Signs, Plastics and Cloth', 'Gears', 'Hinges, Arms and Turntables', 'Magnets and Holders', 'Mechanical', 'Minifig Accessories', 'Minifigs', 'Non-LEGO', 'Other', 'Panels', 'Pins', 'Plants and Animals', 'Plates', 'Plates Angled', 'Plates Round and Dishes', 'Plates Special', 'Pneumatics', 'Power Functions, Mindstorms and Electric', 'Rock', 'String, Bands and Reels', 'Supports, Girders and Cranes', 'Technic Bricks', 'Technic Connectors', 'Technic Panels', 'Technic Special', 'Technic Steering, Suspension and Engine', 'Tiles', 'Tiles Printed', 'Tiles Special', 'Transportation - Land', 'Transportation - Sea and Air', 'Tubes and Hoses', 'Wheels and Tyres', 'Windows and Doors', 'Windscreens and Fuselage'

A.2 Themes

'4 Juniors', 'Agents', 'Architecture', 'Atlantis', 'Belville', 'Bionicle', 'Bulk Bricks', 'Cars', 'Castle', 'Creator', 'Designer Sets', 'Duplo', 'Educational and Dacta', 'Exo-Force', 'Friends', 'Harry Potter', 'Hero Factory', 'Indiana Jones', 'Juniors', 'Legends of Chima', 'Master Building Academy', 'Minecraft', 'Mixels', 'Modular Buildings', 'Monster Fighters', 'Ninjago', 'Pirates', 'Pirates of the Caribbean', 'Power Miners', 'Racers', 'Sculptures', 'Space', 'SpongeBob SquarePants', 'Sports', 'Star Wars', 'Super Heroes',

References

[1] The Lego Group. (n.d.). Retrieved November 22, 2015, from http://www.lego.com

[2] A LEGO Counting problem. (n.d.) Retrieved November 29, 2015, from http://www.math.ku.dk/ eilers/lego.html#whywrong

[3] Brickset.com. (n.d.). Retrieved November 29, 2015, from http://brickset.com/

[4] Rebrickable.com. (n.d.) Retrieved November 29, 2015, from https://rebrickable.com/

[5] Gensim, topic modelling for humans. (n.d.) Retrieved November 29, 2015, from https://radimrehurek.com/gensim/

[6] scikit-learn. (n.d.) Retrieved November 29, 2015, from http://scikit-learn.org/stable/

[7] d3.js Gallery. (n.d.) Retrieved November 29, 2015, from https://github.com/mbostock/d3/wiki/Gallery

[8] Weixin Wang, Hui Wang, Guozhong Dai, and Hongan Wang. (2006). Visualization of large hierarchical data by circle packing. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06), Rebecca Grinter, Thomas Rodden, Paul Aoki, Ed Cutrell, Robin Jeffries, and Gary Olson (Eds.). ACM, New York, NY, USA, 517-520. DOI=http://dx.doi.org/10.1145/1124772.1124851

[9] mining-lego-data, Github Repository for this thesis. (2015). Retrieved December 3, 2015, https://github.com/Xiaodan/mining-lego-data/tree/master/data

[10] Huang, A. (2008, April). Similarity measures for text document clustering. In Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand (pp. 49-56).

[11] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. (2003). Latent dirichlet allocation. J. Mach. Learn. Res. 3 (March 2003), 993-1022.

[12] Lawson, D. J., and Falush, D. (2012). Similarity matrices and clustering algorithms for population identification using genetic data.

[13] The Natural Language Processing Dictionary. (2012). Retrieved November 29, 2015, from http://www.cse.unsw.edu.au/ billw/nlpdict.html#lexicon