Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Towards More Natural Explanations of User Preferences

Master's Thesis in Computer Science
by

Renny Octavia Tan

Internal Supervisor

Krisztian Balog

July 15, 2020

"Believe you can and you're halfway there."

Theodore Roosevelt

Abstract

Explainable recommendations refer to algorithms or methods that enable recommender systems to provide recommendations to the users, as well as to explain the reason why the items or products are being recommended. Recently, a concept of explainability in terms of user preferences is introduced. It provides a mechanism for recommender systems to explain their understanding of the user's preferences by generating user preference statements in the form of text.

In this thesis, we explore different approaches to making the user preference statements to sounding more natural through paraphrasing, while at the same time still preserving relevancy of the sentence, with correct grammar. Two main approaches are: (1) the template-based approach which includes enhancing the template with various sentence patterns and mining more colorful expressions from movie reviews; (2) employing neural language generation techniques by experimenting on state-of-the-art neural network models explicitly built for paraphrase generation, and on transfer learning method by fine-tuning pre-trained neural models. The objective of this work is to discover which of these approaches can be devised in generating paraphrases for user preference statements, that is sounding relevant, grammatically correct, and sounding natural.

We found that some methods or architectures did not work as expected during the experiment, but we also managed to develop a better alternative solution to one of the methods. The experiment results show that both approaches have potential, with their strength and challenges.

Acknowledgements

I want to express my sincere gratitude to my supervisor, **Prof. Krisztian Balog**, for his thorough guidance, dedication, and valuable input in this thesis work.

I would also like to thank my family for their continuous support, especially my husband **David**, and son **Jonathan** who have been cheering me up along the way.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

The use of recommender systems has grown significantly in line with the rapid growth of e-commerce based businesses such as Amazon, Ali Express, and E-bay, as well as online-based content providers such as Netflix and Spotify. It has become something we are dependent on in our daily life, i.e., when doing online shopping, picking movies to watch on Netflix, or picking songs to listen to in Spotify. The use of recommender systems makes decision making and product filtering easier for users when dealing with a wide range of products or services available online.

The explainability of recommendations has gained research highlights in the branch of recommender systems [1]. Explainable recommendations refer to algorithms that enable the recommender systems to not only provide recommendations but also able to explain why they recommend such items or products to the users. It will, in general, improve many aspects of the recommender systems, such as being more transparent and gaining more trust from the users [2, 3].

In recent work, Balog et al. [1] introduced a concept of explainability to explaining user preferences. Instead of explaining why items are being recommended, the system generates textual descriptions summarizing their understanding of the user's preferences in natural language. The system also allowed users to give feedback based on the given summary for adjustments. Therefore, it will make recommender systems more robust to the change of user preferences, which will most likely happen along the time.

Table 1.1 shows summaries of user's preferences, which we call user preference statements. Essentially, user preference statements can be generated by combining a set of template-based sentences with the user-tags information, that will be elaborated more in the next

- You like 'romantic' movies, especially if they are 'quirky'.
- You like 'plot twist' movies, if they are 'tought-provoking'.
- You don't like 'superhero' movies, unless they are 'action'.
- You like 'melodrama' movies, unless they are 'scary'.

Table 1.1: Example user preference statements.

.

chapters. In this project, we want to make these user preference statements sound more natural through paraphrasing.

Paraphrasing is the task of generating a sentence with different style or expression given an input sentence, while still preserving the meaning. Paraphrase generation is one of the important tasks in Natural Language Processing (NLP) [4–6]. It can be adapted in many NLP applications, such as question-answering, recommendation systems, and dialog-based AI. However, due to the complexity of natural language, this task is still considered challenging [4].

1.2 Objectives

This thesis aims to explore and experiment on different approaches for paraphrasing, to find methods or frameworks which can be devised for generating paraphrases of user preference statements that fulfill the following criteria:

- "Relevant," the paraphrases should retain the intended meaning of the original user preference summary.
- "Grammatical," the paraphrase is having correct grammar.
- "Sounding natural," the paraphrase to sounding more natural.

These will make the generated user preference statements to be more informative and accurate in describing the user's preferences, which will potentially enable the user to give more accurate feedback to the system for improvements.

1.3 Approach and Contributions

Different approaches will be explored to identify which of the approaches can be effectively utilized to generate paraphrase with the criteria mentioned earlier. The two main lines of approaches experimented for paraphrasing are:

- Enhancing the template-based approach, which is based on [1], by creating richer sentence patterns and mining movie reviews for more colorful expressions. This including performing linguistic analysis and perform techniques in natural language processing (NLP) to make a more robust template based solution.
- Employing neural (deep learning) language generation techniques, where we experiment on: (1) utilizing neural network models specifically built for paraphrase generation (neural paraphrasing), such as Deep Generative Framework (VAE-LSTM) [5] and Stacked Residual LSTM [7]; (2) employing transfer learning method through fine-tuning pre-trained neural models, such as autoregressive language model (GPT-2) [8], and Text-to-Text Transfer Transformer (T5) [9].

Following, are the research questions this thesis work would like to address:

- **RQ 1**: Can an effective template-based approach be devised for generating paraphrases of user preference statements which are relevant, grammatically correct, and sound natural?
  - **RQ 1.1**: Does mining different expressions from movie reviews is an effective method to enrich a template-based approach by finding synonyms for user-tags?
- **RQ 2**: Can an effective neural network architecture be devised for generating paraphrases of user preference statements which are relevant, grammatically correct, and sound natural?
  - **RQ 2.1**: Which neural paraphrase generation architecture from prior work is most suitable for this task?
  - **RQ 2.2**: Can synthetic training data be effectively used for fine-tuning?
- **RQ 3**: Which of the template-based and neural approaches performs better?

1.4 Outline

The following are explanations on the outline of the thesis:

Chapter 2 - Related Works: This chapter will explain technical information, previous works relevant to this thesis with the elaboration on benefits and challenges for consideration in the thesis work.

Chapter 3 - Template-based Approach: This chapter will explain the overall idea of the template-based approach, including process flow and how the process interconnects with each other.

Chapter 4 - Template-based Results: This chapter will explain the implementation and experiments related to the template-based approach, including experimental setup and the summary of results.

Chapter 5 - Neural Paraphrasing Approach: This chapter will explain the overall idea and process flow of the neural paraphrasing approach.

Chapter 6 - Neural Paraphrasing Results: This chapter will explain the experiments related to the neural paraphrasing approach, including experimental setup and the summary of results.

Chapter 7 - Conclusion This chapter will explain the thesis's conclusion by addressing research questions and explaining future works.

Chapter 2

Related Work

This chapter will explain technical information and previous works which are used as the basis for this thesis. Section 2.1 will explain briefly about recommender systems in general, including explanations on MovieLens dataset which contains user-tags used in this thesis, explainable recommendations and user preference summarization. Section 2.2 will talk about Natural Language Processing (NLP) and information extraction techniques in NLP. Section 2.3 will explain about the use of neural network architecture in paraphrase generation and the models used in this thesis. Section 2.4 will explain about transfer learning method in NLP, including the pre-trained model used in this thesis.

2.1 Recommender Systems

A recommender system is a software or platform which can provide a recommendation on items to the user [10]. It has been implemented in software or websites which offer various items/products such as e-commerce (Amazon, E-bay), movies (Netflix, IMDB, MovieLens), travel (Tripadvisor), books (Goodreads, LibraryThing), music (Spotify) and many more.

In daily life, we often have to make decisions among available choices and alternatives that we do not have previous experience with, for example, which book to read next, where to stay on vacation, which movies to watch next, and many more [11]. Selections of products, movies, music, books, news, and many other items are widely available, which are mostly irrelevant, thus, adding complexity in decision making. Therefore, we often rely on reviews or suggestions from other people who have experienced them. This behavior is basically what recommendation systems try to adapt [10]. One of the essential characteristics of a recommender system is to connect people with the relevant items or products [12].

In utilizing recommender systems, the provider might have goals such as increasing sales, selling more alternative items, or increasing user satisfaction. On the other hand, the user has other goals, such as finding the most suitable products or finding a more economical alternative. Ideally, recommender systems should be able to bridge the different goals between providers and users [10].

There are several ways to categorize recommender systems; however, three common approaches are:

1. Collaborative filtering, which takes into account the opinion of other users with a similar preference and generates recommendations according to items liked by those users with similar preferences [10, 13, 14]. Several works related to this approach are [12, 15, 16].
2. Content-based filtering, which evaluates items that the user liked in the past, gives recommendations with similar characteristics to those liked items in the past [10]. Works in this category are [1, 17].
3. Hybrid recommender system, which combines 2 or more approaches. The goal is to leverage the complementary benefits of those approaches [18]. Examples are [19, 20].

2.1.1 The MovieLens Dataset

MovieLens[1] is a movie recommender system built by a research group called GroupLens from the University of Minnesota in 1997. This platform is non-commercial based, operated, and developed with the primary purpose of supporting research and education. In general, the system works by asking the user to give ratings (0-5) to some movies, and the system will be able to give personalized recommendations to the user.

In 1998, MovieLens dataset [21] was launched to the public. The data was collected from the activities of the members on the platform, which reflects information about the movie preference of the members. Several datasets have been published over the years which have been used in many research work related to recommender systems, for example, item-based recommendation system [22], and trust-based recommendation system [23].

Datasets that have been released are 100k, 1M, 10M, and 20M. The size indicates the approximate number of ratings on each dataset. These datasets are sampled differently; however, shared some characteristics such as: (1) Only contain user which has minimum

[1] https://movielens.org

Dataset	Data Period	Rating Scale	Users	Movies	Ratings	Tag
ML 100K	9/1997 - 34/1998	1-5, stars	943	1,682	100,000	0
ML 1M	4/2000 - 2/2003	1-5, stars	6,040	3,706	1,000,209	0
ML 10M	1/1995 - 1/2009	0.5-5, half-stars	69,878	10,681	10,000,054	95,580
ML 20M	**1/1995 - 3/2015**	**0.5-5, half-stars**	**138,493**	**27,278**	**20,000,263**	**465,564**

Table 2.1: Brief summary of MovieLens data, the last row in boldfaced is the dataset used in this project. Table reprinted from [21] with slight adjustments.

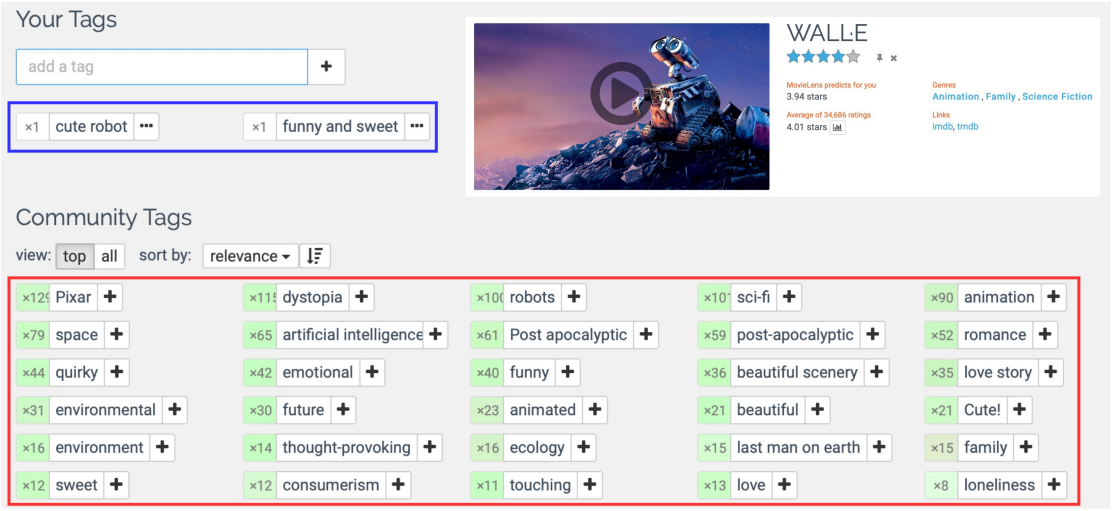


Figure 2.1: Screenshot from MovieLens website to show examples of tags assigned to the "WALL.E" movie. Tags inside the blue box are the tags assigned by the active user, tags inside the red box are the tags assigned by all users. Numbers on the left side of the tags show how many users assigned those tags to the movie.

20 ratings, and (2) they each contain tuple information <user, item, rating, timestamp> where userid is anonymous [21]. The summary of the datasets is presented in Table 2.1.

In 2005, the tagging mechanism was added to the system, which enables users to assign tags to the movie in the form of words or phrases [21]. The example of tags can be seen in Figure 2.1. Only 10k and 20M datasets contain the tags information since these datasets were released after 2005.

In this project, we choose to use the 20M dataset, as it contains user-tags information. One of the limitations of the data is that it contains a non-consistent rating scale, as shown in Table 2.1, the item rating scale change from 1-5 to 0.5-5. However, this limitation does not have a significant impact on this project since user-tags information is the main feature used in this work.

Please note that the tags in this MovieLens dataset are referred as either tag(s) or user-tag(s). We will use tag(s) in most parts of this document for simplicity, and user-tag(s) when there is a need to emphasize or to avoid ambiguity.

Goals	Definition
`Transparency`	To provide explanations on how the system works.
`Scrutability`	Allow users to correct the system.
`Trust`	Increase users' confident level towards the system.
`Effectiveness`	Help user in better decision making.
`Persuasiveness`	Convincing users to try the product.
`Efficiency`	Help users make decisions in a shorter time.
`Satisfaction`	To increase positive users' experience in using the system.

Table 2.2: Possible goals of explainable recommendations. Table adapted from [25].

.

2.1.2 Explainable Recommendations

Explainable recommendations have gained highlights in the recommender systems field of research. As explained by Zhang and Chen [2], explainable recommendations are systems that provide recommendations on items to users as well as explanations on why those items are being recommended. Explainability will help the users to understand how the system works and what is the idea behind the recommendation, which, in general, will improve the recommender system's effectiveness and trustworthiness [2, 3, 24]. For example, users will be more convinced if there is an explanation about why the items are being recommended for them and make it easier to make a decision. One example of an explanation from the Amazon website is "Customers who viewed this item also viewed...."

Table 2.2 shows possible goals of explainable recommendations explained by Tintarev and Masthoff [25]. It is challenging to create explanations that can fulfill all of the goals since they can be conflicting one and another. In most cases, it will be about balancing the trade-off between goals.

There are various types of explanations explored in previous works, such as tag clouds [26], sentences [3], and visual explanation through images [27].

2.1.3 User Preference Summarization

Work by Balog et al. [1] is about movie recommender system concept which having goals of: (1) being transparent; (2) explainable; (3) scrutable. The concept of explainability in their work is different from the previous works. Instead of explaining why certain items are recommended, they adjust the concept to explaining how the system understands user preferences through user preference summarization.

User preferences are represented by the pairs of user-tag and modeled in a set of basic pairwise tag interactions templates [1], as described in Figure 2.2. Following this set of

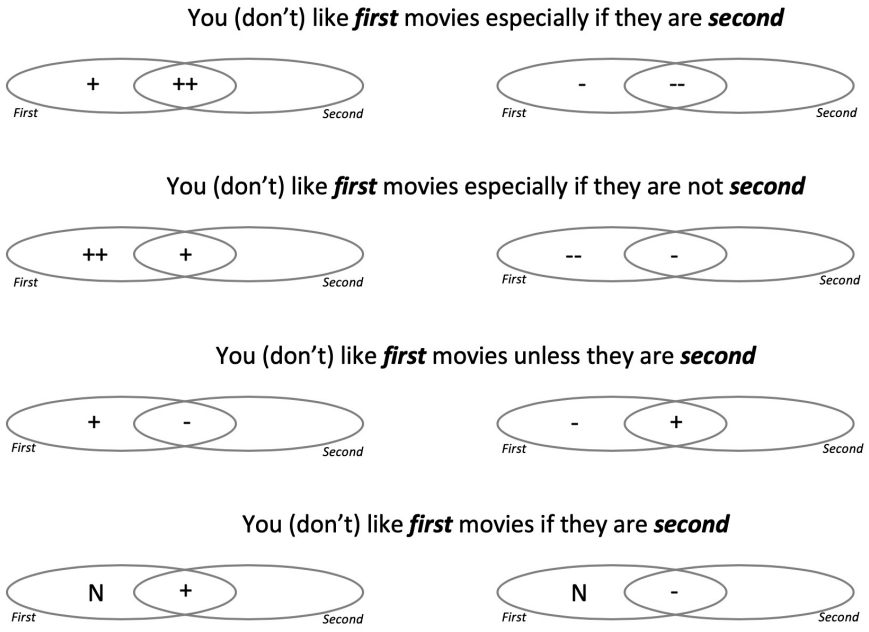


Figure 2.2: Diagram describing the interactions between pair of tags for the basic template solution introduced by Balog et al. [1]. *"First"* is the first tag, and *"Second"* is the second tag. + (like), − (dislike), N (neutral) is user preference level towards the tag, where double signs (++, −−) shows stronger level of preference. Image adapted from [1].

basic pairwise tag interactions, the system will generate textual representations of user preference summaries in the English language, which we call user preference statements. The pairwise tag interactions method is used since they can capture sufficient depth of explanation, i.e., not too general and not too detailed, which can be understood and scrutinized by the user [1].

By letting the users understand how the system perceives their preferences, it will allow the user to give feedback or correction to the system (scrutability) to keep the system updated whenever there are changes to the user preferences [1]. Overall, it will increase the robustness and performance of the recommender system.

This thesis is adapting the pairwise tag interactions templates as the base for implementation, focusing on making improvements on the templates and using other methods to make the generated user preference statements to sound more natural.

2.2 Natural Language (NLP)

The field of NLP was triggered by the idea of having a computer that has the ability to process human language [28]. The work in this field has started approximately from the late 1940s, with various areas of focus that change over time, significantly influenced by

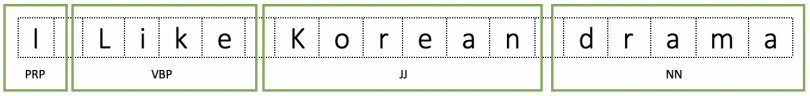


Figure 2.3: Ilustration for simple POS tagging.

the advancement of technology [29], i.e., machine translation (MT), question-answering system, and conversational agent.

This field mainly works on data in the form of text from natural language, which is considered as unstructured data. In the early phase of the NLP process, information extraction needs to be done to get structured information out of the data. There are various techniques/methods used for extracting information, and some of them are used in this work, such as part of speech tagging, chunking, and named entity recognition.

2.2.1 Part-of-Speech (POS) Tagging

Quoted from Bird et al. [30], "the process of classifying words into their parts-of-speech and labeling them accordingly is known as part-of-speech tagging, POS tagging, or simply tagging." A simple POS tagging example can be seen in Figure 2.3, where every single token in a sentence is labeled with its POS-tags, i.e., "PRP," "VBP," "JJ," "NN."

Different corpora have different conventions on tagging. In this project, NLTK is used where the default tagger is using the Penn Treebank tagset [31]. The POS tagset can be seen in Table 2.3.

2.2.2 Chunking

While POS tagging is labeling on the single-token level, chunking is a process of labeling sequences of multi-token (multi-word) into chunks. Illustration in Figure 2.4 shows the result of chunking. The POS-tags of single-tokens are written in blue, while the chunks of multi-tokens are written in red. There are two Noun Phrase (NP) chunks identified, i.e., "the little black cat" and "the sofa."

There are many ways to perform chunking. Two approaches are utilized for this thesis:

1. Classifier based chunking
   It is utilizing chunker trained using CoNLL-2000 [32] Chunking Corpus. The corpus contains 270k Wall Street Journal text with annotations of POS-tags and chunks [30]. There are three types of chunks in this corpus:

Tag	Meaning	Tag	Meaning
CC	Coordinating conj.	TO	infinitival to
CD	Cardinal	UH	Interjection
DT	Determiner	VB	Verb, base form
EX	Existential	VBD	Verb, past tense
FW	Foreign word	VBG	Verb, gerund/present pple
IN	Preposition	VBN	Verb, past participle
JJ	Adjective	VBP	Verb, non-3rd ps. sg. present
JJR	Adjecvtive, comparative	VBZ	Verb, 3rd ps. sg. present
JJS	Adjective, superlative	WDT	Wh-determiner
LS	List item marker	WP	Wh-pronoun
MD	Modal	WP$	Possessive wh-pronoun
NN	Noun, singular or mass	WRB	Wh-adverb
NNS	Noun, plural	#	Pound sign
NNP	Proper noun, singular	$	Dollar sign
NNPS	Proper noun, plural	.	Sentence-final punctuation
PDT	Predeterminer	,	Comma
POS	Possessive ending	:	Colon, semi-colon
PRP	Personal pronoun	(	Left bracket character
PP$	Possessive pronoun	)	Right bracket character
RB	Adverb	"	Straight double quote
RBR	Adverb, comparative	‘	Left open single quote
RBS	Adverb, superlative	"	Left open double quote
RP	Particle	’	Right close single quote
SYM	Symbol	"	Right close double quote

Table 2.3: Penn treebank POS tagset. Table reprinted from [31].

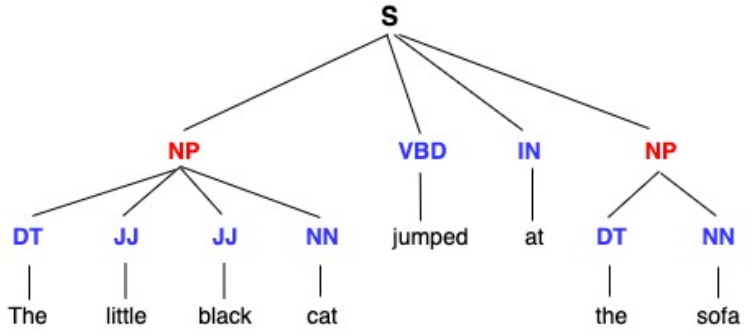


Figure 2.4: Example of Chunking

- Noun Phrase (NP) chunk, for example “the little cat.”
- Verb Phrase (VP) chunk, for example “chased the dog.”
- Preposition Phrase (PP) chunk, for example “in the car.”

By using this technique, the chunking process is not only based on the word-level POS-tags; it also considers the context of the sentence. However, since it only can identify the above types of chunks, we combine it with another approach explained on the next item.

2. Regular expression based chunking
   Apart from the chunk types mentioned above, we also want to include other chunk types, such as Adjective Phrase (AP) chunk, e.g., “visually stunning.” This approach

Named entity type	Example
ORGANIZATION	WHO, UNESCO
PERSON	Tom Cruise, Uma Thurman
LOCATION	Amazon River, Mount Kilimanjaro
DATE	10-07-2020, December
TIME	19:00 p.m., seven p m
MONEY	GBP 10.40, USD 100
PERCENT	eight pct, 8%
GPE	Indonesia, South East Asia

Table 2.4: Example of common types of named entity. Table adapted from [30].

is to utilize regular expressions to recognize patterns of POS-tags of single-tokens to identify the chunks. Example of simple chunking of AP in Listing 1.

```
1  tag = "visually stunning"
2
3  tag_token = word_tokenize(tag)
4  tag_token = nltk.pos_tag(tag_token)
5
6  #Set regular expressions
7  grammar = r"""
8              AP: {<RB|PRP\$>*<JJ>+}   # To chunk AP such as "very beautiful"
9              AP: {<RB|AP><VBG>}       # To chunk AP such as "visually stunning"
10         """
11 chunker = nltk.RegexpParser(grammar)
12 print(chunker.parse(tag_token))
13
14 >>> (S (AP visually/RB stunning/VBG))
```

Listing 1: Simple example of chunking using regular expressions to chunk Adjective Phrase (AP). The input text is "visually stunning." As seen in the result at the bottom, this text is identified as AP. Note that the code is using Python and NLTK library.

2.2.3 Named Entity Recognition

Named entities are noun phrases which refer to specific types of real-world objects such as persons, dates, organizations, countries, and many more [30]. Commonly used types of named entities in information retrieval are listed in Table 2.4.

Named entity recognition is a process to identify and classify the named entity found in raw text, to be categorized based on the pre-defined named entity types [30]. Libraries such as NLTK[2] and spaCy[3] provides functionality for named entity recognition. In this thesis we are using NLTK.

[2]https://www.nltk.org/
[3]https://spacy.io/

2.3 Neural Paraphrase Generation

Paraphrase generation is the task of generating a sentence with different style or expression given an input sentence, while still preserving the meaning. It is still one of the challenging subjects in NLP due to the complex nature of natural language. Nevertheless, it can be an essential part of supporting other applications in NLP, such as paraphrase detection, recommender system, question-answering system, and dialogue-based AI [4, 7]. Various approaches have been explored in the past, such as rule-based method [33], thesaurus-based automatic lexical substitution [34], and statistical machine translation method [6].

Along with the development in deep learning, sequence-to-sequence (Seq2seq) [35] techniques has been widely explored, which has demonstrated positive results in several areas of NLP, such as neural machine translation [36], neural generative question answering [37], and speech recognition [38].

Specific to paraphrasing generation task, in 2016, Prakash et al. [7] introduced a state-of-the-art stacked long short term memory (LSTM) network with residual connection between layers for retaining important information from the previous layer, such as essential words. Gupta et al. [5] used combination of deep generative model (VAE) and Seq2sec with LSTM cells, with a mechanism of introducing original sentences to both encoder and decoder sites, with the aim that the resulted paraphrases will be able to capture the main idea of the original sentences. Brad and Rebedea [39] proposed paraphrasing by utilizing transfer learning to tackle the problem with limited training data for paraphrasing tasks. They used several different types of training data and experimented on transferring the learned knowledge between one and another to find the sequence giving the best result.

Models proposed by Prakash et al. [7] and Gupta et al. [5] were choosen as the base models for neural paraphrasing part of this project.

2.3.1 Stacked Residual LSTM Networks

Sutskever et al. [35] introduced sequence-to-sequence (Seq2seq) model as illustrated in Figure 2.5 where it consists of encoder and decoder block. The encoder will take the input and produce the vector representation of it, while the decoder will take the vector representation of the input and generate the output sentence. With this architecture, the model can handle the sequence of input in variable length and produce a sequence of output, which also in variable length, while the normal neural network has a fixed length of input and a fixed length of output [7]. Each cell in the encoder/decoder can be recurrent neural network (RNN) cell or other models with similar mechanisms, such

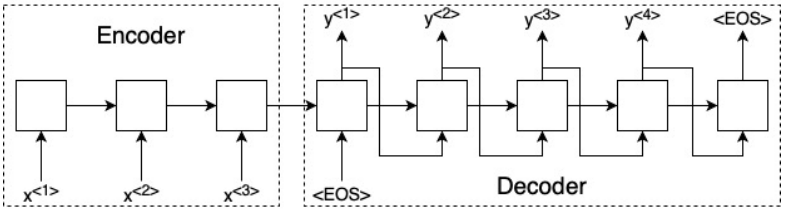


Figure 2.5: Sequence to sequence model with encoder - decoder model. Image adapted from [35].

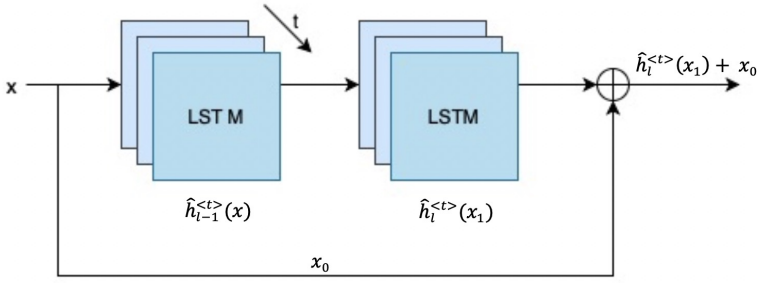


Figure 2.6: Stacked residual LSTM architecture with residual connection added after $n = 2$ layers. Image adapted from [7].

as gated recurrent unit (GRU) and LSTM. LSTM was introduced by Hochreiter and Schmidhuber [40], which provides a solution to the vanishing or exploding gradient problem with RNN [41].

The main idea proposed by Prakash et al. [7] is to add residual connections between stacked LSTM networks. By stacking LSTM networks, the network becomes deeper. It is expected that the deeper the model is, the more it will be able to learn and grasp more complex patterns. The residual networks (ResNet) was first introduced by He et al. [42], where they added residual connection to deep convolutional neural network (CNN). Their experiment identifies a situation that when the network gets deeper, the accuracy gets lower, which was not due to overfitting. It is known as the degradation problem, and to overcome this, the residual connections are added to the model. This same concept is adapted to the stacked LSTM network by Prakash et al. [7].

The proposed model is illustrated in Figure 2.6. The LSTM networks are stacked vertically where each LSTM cell at each time steps in layer l gets an input from each LSTM cell at the corresponding time step in $l-1$ layer. Aside input from the previous layer, each LSTM cell in time step t also gets hidden state from the LSTM cell in previous time step $t-1$. Therefore, hidden state for the layer with no residual connection is denoted as $\hat{h}_l^{<t>} = f_{hl}(h_{l-1}^{<t>}, h_l^{<t-1>})$, while layer with residual connection is denoted as $\hat{h}_l^{<t>} = f_{hl}(h_{l-1}^{<t>}, h_l^{<t-1>}) + x_{l-n})$ [7].

	PPDB	WikiAnswers	MSCOCO
Input	south eastern	what be the symbol of magnesium sulphate	a small kitten is sitting in a bowl
Target	the eastern part	chemical formulum for magnesium sulphate	a cat is curled up in a bowl
Generated	south east	do magnesium sulphate have a formulum	a cat that is sitting on a bowl
Input	organized	what be the biggest galaxy know to man	an old couple at the beach during the day
Target	managed	how many galaxy be there in you known universe	two people sitting on the dock looking at the ocean
Generated	arranged	about how many galaxy do the universe contain	a couple standing on top of a sandy beach
Input	counseling	what do the ph of acid range to	a little baby is sitting on a huge motorcycle
Target	be kept informed	a acid have ph range of what	a little boy sitting alone on a motorcycle
Generated	consultations	how do acid affect ph	a baby sitting on top of a motorcycle

Table 2.5: Example of generated paraphrase using stacked residual LSTM networks. 'Input' is the input text, 'Target' is the paraphrase of input text available in dataset, and 'Generated' is the generated paraphrase by the model. Table content reprinted from [7].

The model was trained and tested on three different datasets: (1) PPDB, which contains data with short text (50% of data are less than 3 words); (2) WikiAnswers, which contains questions pairs crawled from WikiAnswers website; (3) MSCOCO dataset which are image captions annotated by human. The example of generated paraphrase from work [7] can be seen in Table 2.5.

2.3.2 Deep Generative Framework for Paraphrase Generation

In 2015, Bowman et al. [43] proposed a generative model that combines variational autoencoder (VAE) with LSTM. This model uses distributed latent representations of the entire sentence; therefore, it does not only generates words, but also enables the generated paraphrase to grab high-level information such as sentence style, topic, and syntactic properties. However, the generated paraphrases might not capture the main idea/essence of the original sentence. In addressing this issue, Gupta et al. [5] added a mechanism of conditioning both encoder and decoder in the model to the original sentence. The proposed model also uses beam search mechanism so that it can generate multiple paraphrases. They claimed that the variations of generated paraphrases would not degrade far from the initially generated paraphrase because they are picked from top-k beam search results, which are generated from different latent variable (which called z in the paper). While naturally, using beam search in the general Seq2seq model might cause degradation of quality when generating paraphrase variations.

The architecture of the model proposed by Gupta et al. [5] can be seen in Figure 2.7. The model consists of:

- Encoding/input side
  Which consists of two LSTM encoder. The first encoder will take the word embeddings for the original sentence $S(o) = \{w(o)_1, ..., w(o)_n\}$, then output the final state representation of first encoder (h, c) as input to the first stage of second LSTM encoder together with the word embeddings of paraphrase version of the

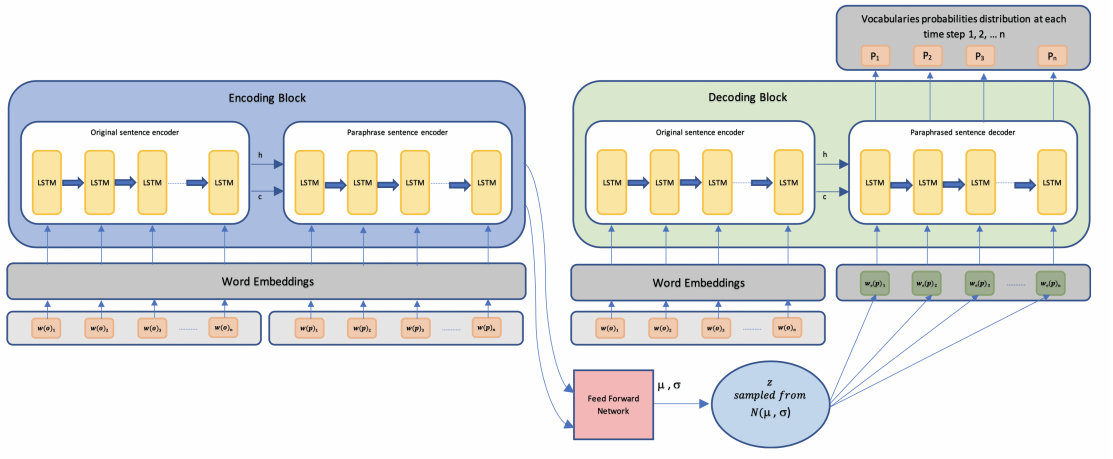


Figure 2.7: VAE-LSTM architecture. Image adapted from [5].

original sentence $S(p) = \{w(p)_1, ..., w(p)_n\}$. The output from the second encoder in the encoding block will be fed to a feed forward network to get parameter μ and σ, where latent variable z is assumed to have Gaussian distribution $z \sim \mathcal{N}(\mu, \sigma^2)$.

- Decoding/output side
  Consists of one encoder and one decoder. In the encoder, the original sentence is re-introduced. The paraphrase will be generated at the decoder block. The first stage in the decoder block will take input from the encoder of the original sentence. Each stage in the decoder block will also take input of latent variable z sampled from the distribution mentioned above, concatenated with results from previous stages. The decoder will generate probabilities for the vocabulary at each time step, and the words can be picked using beam-search.

This model takes pairs of original sentences and its paraphrases as the training data. The training data used are datasets from MSCOCO (pairs of image captions) and Quora (pairs of questions); however, in the author's implementation page[4], only Quora dataset is available.

The resulted paraphrases from this model seem to have quite good grammar structure; although most don't have the same meaning with the original sentences, they are somewhat related. Example of results quoted from [5], for the original sentence "What is my Gmail account?," the generated paraphrase is "is there any way to recover my Gmail account?." Moreover, it can capture new words related to the original sentence, for example, for the original sentence "A man with luggage on wheels standing next to a white van," the generated paraphrase is "a young man standing in front of an airport." It can acquire information that it is happening at the airport, by the words "standing", and "luggage on wheels" in the original sentence.

[4]https://github.com/arvind385801/paraphraseGen

2.4 Transfer Learning from Pre-trained Model

The neural models explained in Section 2.3.1 and Section 2.3.2, are built for specific task, which is paraphrase generation. The common problem in NLP task is insufficient labeled training data suitable for a specific task. In general, the availability of unlabeled text data is massive; however, when conditioned on specific criteria for a specific task, the subset then usually becomes comparably small. It will take a substantial number of resources to collect human annotations [44], while in general, deep learning model requires a high amount of data in order for them to learn well and to reduce possibility of overfitting [45].

Transfer learning has gained much attention in NLP and can be a potential answer to the mentioned problem. The transfer learning method involves pre-training models with general context by using massive unlabeled data from various corpus. Then, it will be followed by transferring the learned knowledge by fine-tuning to a specific task in a supervised manner [44]. By pre-training the model on a large corpus of unlabeled data, the model can learn linguistic information such as grammar, dependencies, and context. It can be further fine-tuned with a smaller dataset for the specific task, thus relieving the need for annotating a large set of data.

Several well-known pre-trained models are BERT [46] which is a bidirectional encoder-transformer based model, GPT-2 [8] which is unidirectional decoder-transformer based, XLNet [47] which is an autoregressive language model, and T5 [9] which is an encoder-decoder transformer based with text-to-text framework.

In this thesis, we experimented on fine-tuning GPT-2 and T5 pre-trained models.

2.4.1 GPT/GPT-2

Generative Pre-Training (GPT) [44] was initially released in June 2018 by OpenAI[5], and followed by GPT-2 [8] in February 2019. OpenAI has released 4 pre-trained GPT-2, which is small (124M), medium (355M), large (774M), and extra-large (1.5B). The model was trained on 40 GB text data from the WebText dataset, which was created by OpenAI.

The GPT architecture for training the small (124M) model is illustrated in Figure 2.8. It consists of 12 layers of decoder-only transformer architecture adapted from [48], which is a modification of Transformer blocks by [49]. GPT-2 largely follows this architecture with some modifications: (1) moving normalization to the input of each sub-block; (2) adding normalization layer after final self-attention block [8].

[5]https://openai.com/

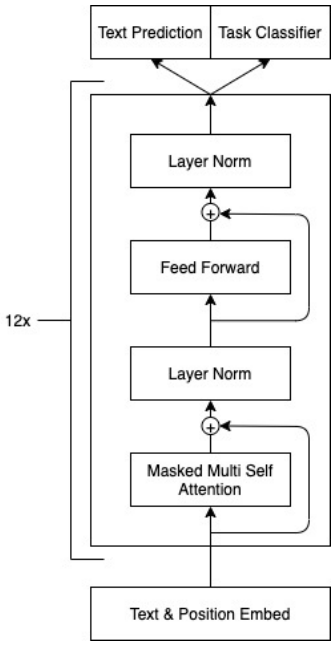


Figure 2.8: GPT architecture for small (124M) model. Image reprinted from [44].

As explained by Radford and Sutskever [44], there are several strong points of their model compared to previous similar approaches. Many research works have used word embeddings trained on unlabeled data [50, 51] to improve diverse specific tasks; however, the word embedding is limited to capturing word-level information, while their model can gain higher-level semantic properties. Previous work that has the closest concept is the work by Howard and Ruder [52], which involves pre-training neural networks and fine-tuning it for the text classification task. The major difference is that the latter uses LSTM network in their architecture, which is more restricted to shorter range structures in comparison to the use of transformer networks, which can capture longer range structures. They also introduced a fine-tuning mechanism, which involves little adaptation without the need to alter the model's architecture, unlike previous works [53, 54] on transfer learning which requires architecture modification.

In [44], it is documented that the knowledge gained by the pre-trained GPT model was successfully transferred to a more specific task with the fine-tuning mechanism. Compared to state of the art, they managed to improve results of 9 out of 12 datasets that were experimented related to question answering, semantic similarity assessment, entailment determination, and text classification tasks.

2.4.2 T5

Raffel et al. [9] demonstrated a thorough study in comparing pre-training method, dataset, transfer learning approach from various existing pre-trained models with different

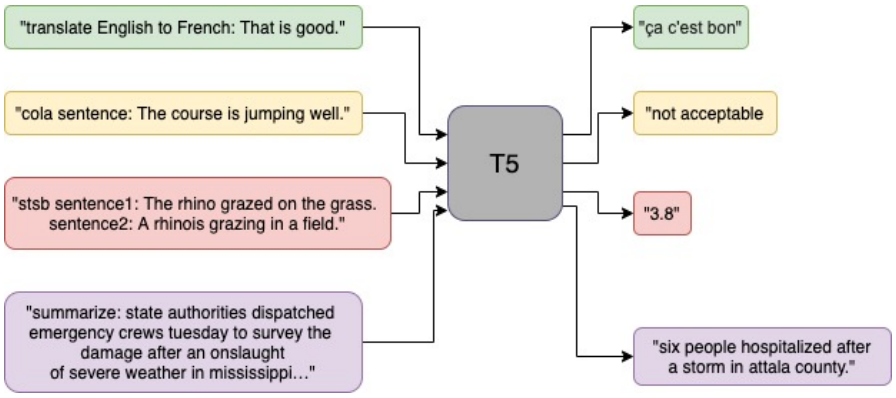


Figure 2.9: Text-to-Text Transfer Transformer (T5) framework. Image adapted from [9].

.

architecture. With the knowledge gained from the study, they created a new model called Text-to-Text Transfer Transformer (T5), which was pre-trained on the dataset called Colossal Clean Crawled Corpus (C4) consisting around 750 GB English text data.

Align with the name, T5 conditioned all NLP task in a text-to-text framework, which means it receives text input and produces text output. This model can even handle regression by outputting the text representing the numbers, instead of the number itself [9]. This framework is illustrated in Figure 2.9.

While GPT-2 uses decoder-only transformer architecture, T5 adapts the original architecture from [49], which is an encoder-decoder transformer. They demonstrated that the architecture is very suitable for their text-to-text framework, where it performs well in text generation and classification tasks. They also showed that the computational cost does not differ much from encoder-only [46] or decoder-only models [8, 44].

Several sizes of pre-trained T5 models released are T5-Small (60 million parameters), T5-Base (220 million parameters), T5-Large (770 million parameters), T5-3B (3 billion parameters), and T5-11B (11 billion parameters).

Chapter 3

Template-based Approach

This chapter will explain one of the main line approach for paraphrasing briefly explained in Section 1.3, which is the template-based approach.

Recall from explanation in Section 2.1.1, tags in MovieLens dataset are referred as either tag(s) or user-tag(s). We will use tag(s) in most parts of this document for simplicity, and user-tag(s) when there is a need to emphasize or to avoid ambiguity, especially in this chapter and Chapter 4, where another term such as POS-tag(s) will be introduced.

This line of work consists of several modules as shown in Figure 3.1, which are done in the following sequence:

1. Development of "Basic Template-based Approach" (basic template)
   This is a module built for generating textual summary of user preference (user

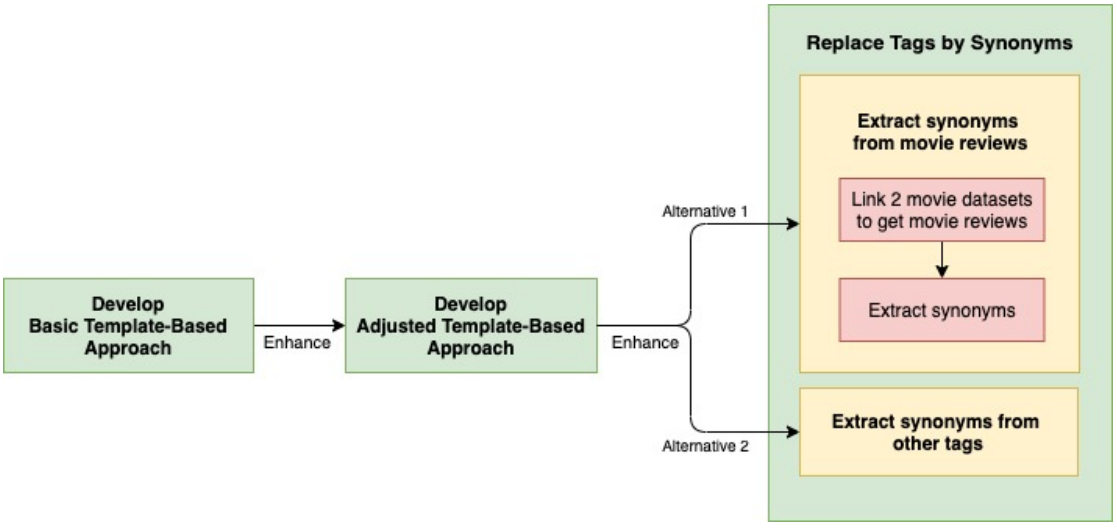


Figure 3.1: Modules in template-based approach.

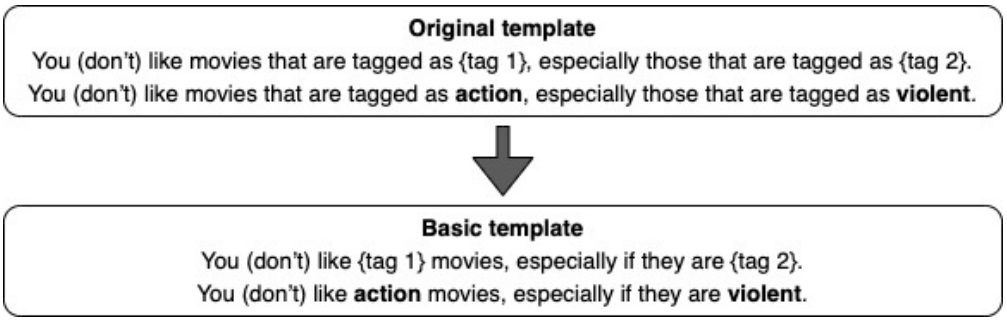


Figure 3.2: Initial change to the original template. The sentence pattern template use "{}" for placeholders for the tags, while texts in bold indicate when those have been instantiated with the tags.

preference statement), following basic sentence pattern based on the pairwise tag interactions (Figure 2.2). This will be further explained in Section 3.1.

2. Development of "Adjusted Template-based Approach" (adjusted template)
   Which is a further development of the basic template to enhance the user preference statements to have more varieties in the sentence patterns but still adhering to the pairwise tag interactions rules. This will be further explained in Section 3.2.

3. Replace user-tags by synonyms
   This is a further enhancement to the template-based approach by replacing tags with similar words or phrases. The first alternative is to extract similar words or phrases from movie reviews. Section 3.3, will explain why a second alternative, which is extracting synonyms from other tags, was explored.

User preference is represented by a pair of tags, i.e., tag 1 and tag 2, accompanied by the user preference level for each tag. User preference level is represented by $--$, $-$, N, $+$, and $++$; where $+$, $-$, and N, indicate positive, negative, and neutral sentiments accordingly, with double signs increasing the intensity level. To summarize the user preference in natural language, the template-based approach will take input of user preference in the form of, e.g. ((tag 1, N),(tag 2, +)).

3.1 Basic Template-Based Approach

Basic template is built on the basis of pairwise tag interactions as described in Figure 2.2, which is adapted from work [1]. As our overall motivation is to make the generated user preference statement sound more natural, modification is made on the original template in [1], by removing the words "tagged with" which does not sound natural as illustrated in Figure 3.2.

After the modification, the basic pairwise tag interactions is formulated in this basic template as follows:

1. `You (don't) like {tag 1} movies especially if they are {tag 2}.`
   For ((tag 1, +), (tag 2, ++)) or ((tag 1, −),(tag 2, −−)).
2. `You (don't) like {tag 1} movies especially if they are not {tag 2}.`
   For ((tag 1, ++), (tag 2, +)) or ((tag 1, −−),(tag 2, −)).
3. `You (don't) like {tag 1} movies unless they are {tag 2}.`
   For ((tag 1, +),(tag 2, −)) or ((tag 1, −),(tag 2, +)).
4. `You (don't) like {tag 1} movies if they are {tag 2}.`
   For ((tag 1, N),(tag 2, +)) or ((tag 1, N),(tag 2, −)).

{tag 1} and {tag 2} can be replaced with randomly chosen pair of tags.

The tags processing and filtering method and more detailed information will be explained in Chapter 4. These tags are created and assigned to individual movies by MovieLens users, which can be in the form of single words or phrases. The quality of the tags is uncontrolled, as users can write anything they like. Therefore it can have poor grammatical structures and meaning.

As the tags come in different forms and different part-of-speech (POS), it is evident that this basic template will not be suitable for all tags. Therefore, further development of the basic template is needed to improve the template-based approach.

3.2 Adjusted Template-Based Approach

The adjusted template is an enhancement of the basic template, to be able to: (1) deal with user-tags with different categories based on POS-tags and other properties; and (2) to gain a wider variety of sentence patterns, however still adhere to the pairwise tag interactions.

The adjusted template is made with added flexibility in the sentence pattern with basic structure as seen in Figure 3.3. Here, the full sentence is treated as it has two parts: sentence 1 that contains tag 1 and sentence 2 contains tag 2. Each part of the sentence will be enriched with patterns variations, which will be modified to suit different user-tags categories.

Creating the adjusted template involves below processes:

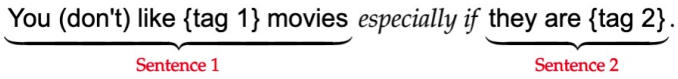


Figure 3.3: Ilustration for adjusted template, where not only to enable to replace tag 1 and tag 2, but divide the full sentence into two parts (sentence 1 and sentence 2) which can also be replaced with various sentence pattern.

Category	Entity / Special case	Sentence 1 pattern		Sentence 2 pattern
All	Ends with movie/movies/ film/films	**You (don't) like** {tag 1}	(especially/particularly) if / (unless/except if) / if	they are (not) {tag 2}
	Entity: GPE	**You (don't) like** movies from {tag 1} **You (don't) like** movies about {tag 1}		they are (not) from {tag 2} they are (not) about (tag 2)
	Movie Genre	**You (don't) like** {tag 1} movies **You (don't) like** movies with {tag 1} genre **You (don't) like** movies full of {tag 1}		they are (not) {tag 2) they (don't) contain {tag 2} they are (not) full of {tag 2}
Adjective, Verb & Preposition	General	**You (don't) like** {tag 1} movies		they are (not) {tag 2}
Noun & Numeral	General	**You (don't) like** movies about {tag 1}		they are (not) about {tag 2}
	Starts with adj/dt ends with verb-ing	**You (don't) like** movies with {tag 1}		they are (not) with {tag 2}
	Entity: PERSON Type: director	**You (don't) like** movies directed by {tag 1} {tag 1} movies is (not) for you		they are (not) directed by {tag 2} they are (not) movies by {tag 2}
	Entity: PERSON Type: actor	**You (don't) like** movies starred by (tag 1) **You (don't) like** movies starring {tag 1} **You (don't) like** movies played by {tag 1}		they are (not) starred by{tag 2} they are (not) starring {tag 2} they are (not) played by {tag 2}
	Entity: PERSON Type: unknown	**You (don't) like** movies about {tag 1}		they are (not) about {tag 2} they are (not) {tag 2} movies

Table 3.1: The table shows the variety of sentence patterns based on the mapped categories and sub-categories. Sentence 1 and sentence 2 patterns can be paired randomly depending on the category of each tag 1 and tag 2. The green part of the text can be replaced by variations in Table 3.2.

1. User-tags analysis and mapping

   To map the user-tags into potential categories based on properties such as POS-tags, named entity, and other properties such as genre and type of profession. The detailed explanation of the process is explained in Section 4.1.

2. Error analysis

   Based on the categories resulted in the user-tags analysis and mapping, we perform error analysis to the basic template. In the analysis, observation is made to failure cases as a basis to decide on what kind of sentence pattern needed to be added in the adjusted template to handle the failure cases and to fit against the available user-tags categories. Error analysis was also performed to the adjusted template to see how far the improvement was from the basic template, and to understand the failure cases in the adjusted template. The detail explanation on error analysis is explained in Section 4.2

The above processes resulted in the adjusted template with diverse sentence patterns, as listed in Table 3.1. As a reminder note, tag 1 and tag 2 are randomly picked; therefore, they can fall into different categories. Moreover, the "like" and "don't like" phrases can be changed randomly to alternative phrases, as in Table 3.2.

Alternative phrases for	
You like	**You don't like**
You love	You hate
You prefer	You don't prefer
You enjoy	You dislike
You are into	You are not into
You would watch	You wouldn't watch
You like to watch	You don't like to watch
You like watching	You don't like watching
You love to watch	You hate to watch
You love watching	You hate watching
You prefer to watch	You don't prefer to watch
You prefer watching	You don't prefer watching
You enjoy watching	You dislike watching
You are interested in	You are not interested in

Table 3.2: The phrase "You (don't) like" from the sentence pattern in Table 3.1 can be replaced with these phrases to add more variety.

3.3 Replace User-Tags by Synonyms

The further approach for improving the template-based approach is to replace user-tags with more colorful representations to express sentences generated by the adjusted template in different ways.

The first method is to extract similar words or phrases (synonyms) from movie reviews. This method is considered reasonable since tags can be seen as a mini-reviews/opinions from users towards movies; therefore, it is expected that we can extract more colorful similar representations for a tag from movie reviews linked to that tag.

This approach involves two processes as follows:

1. Linking MovieLens dataset to Amazon movie dataset
   So far, we only use tags from MovieLens dataset, which does not have movie reviews. Movie reviews used in this approach are from Amazon movies, which are currently not linked to the MovieLens dataset. Therefore a linking process needs to be done between MovieLens dataset and Amazon movie dataset. The linking is done by matching movie titles from the two datasets by string-to-string matching. The detail is described in Section 4.3.

2. Extracting synonyms from Amazon movie reviews for tags in Movielens
   After linking the two datasets, we extract similar words or phrases for the tags. The detail is described in Section 4.4.

As can be seen in Section 4.4, the results of this method are "related" words or phrases for the tags; however, they cannot be considered as synonyms.

In order to gather more insights, a questionnaire was made to collect feedback on how humans would paraphrase the tags and the whole sentences. The questionnaire setup and result analysis are explained in Section 4.5. Based on the feedback, we see that 43% of tags replacements suggested by the respondents are matching with other tags in the dataset. It allows for another potential method, which is extracting synonyms from other available tags explained in Section 4.6.

Chapter 4

Template-Based Results

This Chapter will explain experimental results related to the template-based approach.

4.1 User-Tags Analysis and Mapping

This experiment is to analyze and map user-tags into categories as briefly explained in Section 3.2.

4.1.1 Dataset

The dataset used for the basic template and adjusted template is from MovieLens 20M Dataset[1], in particular "tags.csv" which contains 35,173 user-tags which are assigned by 7,801 userId to 19,545 movieId. The tags are filtered to only include those assigned by at least 5 users and assigned to at least 2 movies. Tags with inappropriate content are also filtered out, which resulted in 5610 tags.

4.1.2 Experimental Setup and Result

There is two main focus in the user-tags mapping, which are: (1) to find out which part-of-speech (POS) it belongs to; and (2) what named entity it contains. From the mapping result, the tags will be grouped into several categories, and sentence patterns will be created to match the categories.

Below is the more detailed explanation on the mapping process:

[1] https://grouplens.org/datasets/movielens/20m/

1. **Part-of-speech tagging and chunking**

   Quoting explanation from [30], "The process of classifying words into their parts-of-speech and labeling them accordingly is known as part-of-speech tagging, POS tagging, or simply tagging." POS-tag(s) is the term used in this report for the label. Using the NLTK library, we can label each word in user-tags with its POS-tag; however, some user-tags which are in the form of phrases (multi-word) should be handled differently. For user-tags in the form of phrases, chunking is done to categorize the whole phrase into either adjective phrase (AP), noun phrase (NP), verb phrase (VP), or prepositional phrase (PP). Chunking is done by utilizing regular expressions and classifier-based chunker as explained in Section 2.2.2.

   This approach, however, is not resulting in perfect results. Some miss-tagging can be seen, in the following cases:

   (a) Miss-tagging single words

   - Miss-tagging noun <-> adjective, for example: (1) "queen," "epidemic," and "psychopath" are tagged as adjectives instead of nouns; (2) "tear jerker" and "disgusting" are tagged as nouns instead of adjectives.
   - Verb that ends with "-ing" or "-ed" can act as an adjective, POS tagging fails to label some of single word verb-ing or verb-ed as adjectives such as "amazing," "annoying," "captivating," "twisted," and "biased." Those are miss-tagged as verbs instead.

   To minimize this issue, the single word tag is being put into a sentence, for example "it is a {tag} film," "it is a {tag} movies," or "it is a {tag} scenery," so that the tagger can see the context of that word in a sentence. The result is very dependant on what sentence being chosen. This is not a one-size-fits-all solution, as our experiment shows that when it performs strongly in tagging, for example, adjectives, it performs poorly in tagging nouns. Therefore the number of miss-tagged adjectives become lower, but on the other hand, the miss-tagged nouns became higher. The better approach is to find a sentence that has a balanced result, in this case, we choose "it is a {tag} film."

   (b) Miss-tagging multi-word

   When chunking adjective phrase (AP), which ends with verb-ing, for example, "visually stunning" and "strangely compelling," some noun phrases such as "bad ending" and "excellent acting" were also captured due to the inability in recognizing "ending" and "acting" as nouns.

   After mapping, the proportion POS-tags is shown in Table 4.1.

2. **Named entity recognition**

   This task is to determine what entity the user-tags have, such as persons, locations,

	Percentage	**Example of tags**
Adjectives	12.06%	"animated," "stunning," "strangely compelling" (AP)
Nouns	80.39%	"abduction," "good musicals" (NP), "bad acting" (NP)
Verbs	0.96%	"climbing," "based on a book" (VP)
Adverbs	0.12%	"bully"
Prepositions	0.14%	"on hold" (PP), "on the run" (PP), "in netflix queue" (PP)
Numerals	0.5%	"007," "16mm"
Uknown	5.82%	"doesn't live up to its full potential"

Table 4.1: POS-tags proportion of the user-tags. Adjectives include single word adjectives and adjective phrases (AP), also the same case with Nouns, Verbs, and Prepositions. "Unknown" is a category for multi-words tags, in which the chunking process cannot conclude their POS-tags.

organizations, etc. From the total number of recognized entities, 90% of the entities are "PERSON," therefore, in the adjusted template, there are sentence patterns to accommodate tags that have "PERSON" entities. Furthermore, to add more detailed information about the profession of the "PERSON" in the movie industry. They are checked against the list of movie directors and the list of top-1000 actors or actresses from IMDb, to get additional information for the tags if they are "actor" or "director." When checking against this list, we also captured several names that failed to be identified as "PERSON" by the named entity recognition.

4.2 Error Analysis of Basic Template and Adjusted Template

The error analysis is performed in the following sequence: (1) analyzing the failure cases in the basic template; (2) observing the failure cases and find sentence patterns to be added to the adjusted template, which can handle these failures. Apart from that, also create sentence patterns to fit all other identified user-tags categories; (3) observing improvement in the adjusted template compared to the basic template, and understanding failure cases resulted from the adjusted template.

4.2.1 Dataset

The dataset used for this section is the same as in Section 4.1.

4.2.2 Experimental Setup and Result

At first glance, user-tags that fall within the adjective category seem to fit sentence patterns in the basic template. To check this hypothesis, we randomly select 100 pairs

Basic template - adjective tags		
Percentage		**Examples of success and failure case**
Success cases	76%	- You don't like *quite romantic* movies unless they are *hillarious.* - You like *confrontational* movies especially if they are *animated.*
Failure cases	24%	Miss-tagging noun as adjective (100%): - You like *sentimental* movies especially if **they are not *comic.*** - You don't like *inaccurate* movies unless **they are *musician.***

Table 4.2: Error analysis from basic template using only adjective tags. 100% of failure cases are caused by nouns which are miss tagged as adjective

of adjective tags with randomly selected user preferences level for each tag and created 100 sentences using the basic template. The same is done for non-adjective tags. The analysis result from the success/failure cases can be seen in Table 4.2 and Table 4.3. Note that the bold dark-red text shows where the errors lie.

The result in Table 4.2 seems to go in hand with the initial hypothesis, that the basic template is suitable for adjective tags since the failure cases are mostly due noun tags which are miss-tagged as adjectives. Table 4.3 shows that the basic template is not suitable for non-adjectives especially nouns. As seen in Table 4.1, nouns contribute as the largest proportion of the tags. Based on the failure cases, we create the adjusted template where new sentence patterns are created to handle the failure cases.

The adjusted template contains a variety of sentence patterns based on categories and sub-categories, as described in Table 3.1. Nouns and numerals (cardinals) are grouped together, while the remaining categories are grouped together with adjectives. The basic template's sentence patterns are adapted for adjective tags in the adjusted template since the basic template is suitable for adjective tags based on the previous conclusion. Nouns category contains more sub-categories compared to other categories, thus has more variety of sentence patterns. As 30% of nouns have entity "PERSON," we created specific sentence patterns to handle this sub-category. In addition, several patterns are made to handle some special cases regardless of the POS-tags, which can be seen in the first category "All" in the table.

To compare basic template and adjusted template, we generate 100 sentences using randomly selected 100 pairs of tags from all categories for each template. The results of success/failure cases analysis are in Tables 4.4 and 4.5. There is a significant improvement in the success case from 25% to 84%.

The largest failure cases in the adjusted template came from some nouns which fit grammatically to the sentence pattern; however, they do not sound natural. Ond of the example is shown in Table 4.5, "You love movies about *good plot*" is grammatically correct; however, it will be more suitable to write it as "You love movies with *good plot.*"

Basic template - non adjective tags		
Percentage		**Examples of success and failure case**
Success cases	10%	- You don't like *tom berenger* movies especially if they are *over the top.* - You like *scarlett johansson* movies if they are *screwball comedy.*
Failure cases	90%	Not suitable sentence pattern for Noun - General (56%): - You like *dustin hoffman* movies especially if **they are not *dog.*** - You don't like *good ending* movies unless **they are *parallels worlds.***
		Not suitable sentence pattern for Noun - Person tags (32%): - You don't like *1800s* movies unless **they are *michael j. fox.*** - You don't like *marvel comics* movies if **they are *zac efron.***
		Others (12%): - You don't like *french movie* movies especially if **they are *not as good as i expected.*** - You like *reboot* movies especially if **they are *south korea.***

Table 4.3: Error analysis from basic template using only non-adjective tags. With 90% failure cases, we can see that the basic templates are not suitable for non-adjectives.

Basic template - all type tags		
Percentage		**Examples of success and failure cases**
Success cases	25%	- You like *thoughtful* movies, especially if they are *futuristic.* - You like *firefighters* movies, especially if they are *based on a computer game.*
Failure cases	75%	Not suitable for sentence patterns for Noun - General (57%): - You don't like *dark comedy* movies, especially if **they are *unrealistic characters*.** - You like *fascism* movies, unless **they are *disfunctional family.***
		Not suitable sentence patterns for Noun - Person tags (37%): - You like *village* movies, especially if **they are not *keanu reeves.*** - You like *consciousness* movies if **they are *meg ryan.***
		Tag quality and others (6%): - You don't like *great concept* movies if **they are *villain nonexistent or not needed for good story.*** - You like *terrorism* movies, especially if **they are *1990s.***

Table 4.4: Error analysis of the basic template using all types of tags, for comparison with the error analysis of the adjusted template on Table 4.5.

Adjusted template - all type tags		
Percentage		**Examples of success and failure cases**
Success cases	84%	- You like watching movies about *village*, particularly if they are not starring *keanu reeves.* - You love watching movies about *fascism*, except if they are about *disfunctional family.* - You dislike watching movies with *horrible ending* if they are about *political corruption.*
Failure cases	16%	Not suitable sentence patterns for some form of Nouns (50%): - You are into movies about *weak dialogue* if **they are movies by *ernst lubitsch.*** - You love movies about *good plot*, unless **they are about *zero gravity.***
		Miss-tagging noun <-> adjective, therefore, the tag does not fit the sentence pattern (37%): - You hate to watch movies about *mentor*, except if **they are about *baffling.*** - **You hate movies about *wierd*** if they are starred by *yul brynner.*
		Tag quality & others (13%): - You dislike movies about *great concept* if **they are *villain nonexistent or not needed for good story.*** - You like to watch movies about *doctors* if **they are *seen it before.***

Table 4.5: Error analysis for adjusted template using all types of tags.

Movie Title	MovieLens Id	Movie Title	MovieLens Id
Aladdin (1992)	588 ; 114240	Chaos (2005)	47254 ; 67459
Emma (1996)	838 ; 26958	Offside (2006)	48682 ; 80330
Men with Guns (1997)	1788 ; 26982	Blackout (2007)	66140 ; 85070
Hamlet (2000)	3598 ; 65665	Darling (2007)	93279 ; 130062
War of the Worlds (2005)	34048 ; 64997	Girl, The (2012)	97773 ; 101212
Casanova (2005)	42015 ; 128862	Clear History (2013)	104155 ; 122940
20,000 Leagues Under the Sea (1997)	102190 ; 114130	Johnny Express (2014)	111519 ; 128991
Beneath (2013)	104035 ; 115777	Paradise (2013)	113459 ; 121586

Table 4.6: Movie title duplicates.

Observing the POS component of "*good plot*," it is a noun phrase that starts with an adjective and ends with a noun. It is challenging to apply a strict rule saying that if the tags are in adjective-noun form, thus the sentence pattern is "You dislike movies with {tag}," since this will also not be suitable for other noun phrase with same structure such as "*"bad police*" and "*post apocalypse*."

Please note that the analysis of success and failure case is done by the author, so it is subject to opinion and linguistic knowledge of the author.

4.3 Linking MovieLens Dataset to Amazon Movie Dataset

In this task, we want to link MovieLens Id with Amazon product Id (ASIN) to extract Amazon movie reviews for the linked MovieLens movies in a future step.

The general idea is to match the movie titles between MovieLens and Amazon movies.

4.3.1 Dataset

4.3.1.1 MovieLens Dataset

This experiment uses MovieLens Id and Title from MovieLens 20M dataset[2], in particular "movies.csv" and "links.csv." There are 27,278 unique MovieLens Id; however, there are duplicates in the title (with different MovieLens Id), which can be seen in Table 4.6.

4.3.1.2 Amazon Movie Dataset

This experiment is using Amazon dataset[3], in particular "meta_Movies_and_TV," which contains dictionaries with Amazon ASIN and movie titles. The number of items

[2]https://grouplens.org/datasets/movielens/20m/
[3]http://deepyeti.ucsd.edu/jianmo/amazon/index.html

in the dataset is 203,970; however, there are 21,927 duplicates, so the dataset ended up with 182,043 unique ASIN. Note:

- 67 ASIN do not have titles.
- One movie can have several different ASIN (movie, dvd, blueray, etc.).
- There are 3059 titles duplicates for different movies which are distributed to 6893 ASIN, for example, ASIN B00151QYGC, B0016RRPPE, and B003WSKWUU, all have identical title "The Master", but, checking manually on the Amazon website, all of them are not referring to the same movie.
- There is no year information in the dataset, so it is difficult to distinguish versions of movies or different movies with the same title.

4.3.1.3 IMDb Movies Dataset

IMDb dataset[4] is used, in particular the following data:

- title.basics.tsv.gz - contains IMDb Id, primary title, original title.
- title.akas.tsv.gz - contains movie title a.k.a.

Since there is a link between MovieLens 20M movie Id and IMDb Id, we want to extract variations of movie titles for MovieLens movies from IMDb dataset, such as "primary" titles, "original" titles, and IMDb movies a.k.a (a list of all available alternative movie titles).

The mapped IMDb Ids in MovieLens dataset only contain n last digits; therefore, they have to be converted first prior to extracting titles from the IMDb dataset. IMDb Id is supposed to be in "ttXXXXXXX" format, where "tt" is followed by minimum 7 digit numbers (can be more). Therefore, the following changes were made:

- If the mapped IMDb Id is 5 digit numbers ("12345"), it becomes "tt0012345."
- If the mapped IMDb Id is 6 digit numbers ("123456"), it becomes "tt0123456."
- If the mapped IMDb Id is in 7 or more digits, "1234567" becomes "tt1234567" and "12345678" becomes "tt12345678."

There are 46 mapped IMDb Id in MovieLens dataset that cannot be found in the IMDb dataset.

[4]https://datasets.imdbws.com/. Information courtesy of IMDb (http://www.imdb.com). Used with permission.

4.3.2 Experimental Setup

String-to-string exact matching ("==") is used when matching movie titles between MovieLens and Amazon movie titles. Partial matching is avoided since it is more important to get correct matches than as many matches as possible, yet contain high error match possibility.

Before performing the string-to-string match, pre-processing is done to movie titles from both datasets.

4.3.2.1 MovieLens Movie Title Pre-Processing

For each MovieLens movie title, the pre-processing will resulted in a list containing the cleaned title and all alternative titles extracted during the process. The following steps are done during pre-processing of movie titles:

- Lower case the titles and remove any spaces at the beginning and the end.
- Most of MovieLens titles contain year information in parentheses, for example "(1995)," while Amazon movie titles do not have year information. Therefore, year information in MovieLens titles is being removed.
- Both MovieLens and Amazon have inconsistencies in placing the articles in phrases. Some articles are placed at the beginning, while some are placed at the end, such as "Island, the" and "The Island." Therefore articles "the," "a," "an," "le," "la," "les," "il," "el," and "l' " are moved to the front of phrases.
- MovieLens titles can contain a.k.a, which usually found in parentheses. Some start with "a.k.a." and some are not, which can be seen in this example: "Island, The (a.k.a. Naked Island) (Hadaka no shima) (1960)." This movie "Island, The" has two a.k.a ("Naked Island" and "Hadaka no shima"). Therefore we extract all the a.k.a into a list as alternative titles to matched for that movie title.
- Remove inconsistencies:
  - Semicolon (":") and hyphen ("-") are used interchangeably in Amazon movie titles as a mark for subtitles. For example, "star wars III : revenge of the sith," and "star wars III - revenge of the sith." They appear in various formats, such as having space before or after the mark, or both before and after the mark, or none. Therefore, all ":" and "-" are removed from both MovieLens and Amazon movie titles.

MovieLens Title	List of Pre-processed Title and a.k.a
'City of Lost Children, The (Cité des enfants perdus, La) (1995)'	['the city of lost children', 'la cité des enfants perdus', 'city of lost children', 'thecityoflostchildren', 'lacitédesenfantsperdus', 'cityoflostchildren']
'Island, The (a.k.a. Naked Island) (Hadaka no shima) (1960)'	['the island', 'naked island', 'hadaka no shima', 'the naked island', 'island', 'theisland', 'nakedisland', 'hadakanoshima', 'thenakedisland']
'High School Musical 3: Senior Year (2008)'	['high school musical 3 senior year', 'high school musical 3', 'senior year', 'high school musical iii senior year', 'high school musical three senior year', 'high school musical iii', 'high school musical three', 'highschoolmusical3senioryear', 'highschoolmusical3', 'senioryear', 'highschoolmusicaliiisenioryear', 'highschoolmusicalthreesenioryear', 'highschoolmusicaliii', 'highschoolmusicalthree']

Table 4.7: Example of MovieLens title and its pre-processed results. The result for each MovieLens title is in a list containing the pre-processed title and all alternative titles to be matched.

  - Remove double spaces.
  - Remove all commas (","), question marks ("?"), exclamation marks ("!"), and full stops (".") for both MovieLens and Amazon movie titles.

- Adding a.k.a to the list of alternative tittles by:
  - Taking "original title" and "primary title" from linked IMDb movies. Note that only "primary" and "original" titles which are used as additional a.k.a. List of IMDb movies a.k.a in the IMDb dataset, is not used since they introduce more matching errors, as some of the titles become too general. For example, "star wars 3" movie can have IMDb a.k.a of "star wars," which is very general, or can have totally different titles, which are sometimes correct, but most of the time, it leads to matches to wrong movies.
  - Converting numbers to roman numerals, as sometimes Amazon movie titles use the roman numerals.
  - Removing all spaces due to inconsistencies in writing the title, for example, "Goldeneye" and "golden eye."

Examples of pre-processed result for MovieLens Titles can be seen in Table 4.7.

4.3.2.2 Amazon Movie Title Pre-Processing

Like the pre-processing of MovieLens movie title, pre-processing of each Amazon movie title will result in a list containing the cleaned title and all alternative titles extracted during the process. The followings are performed during pre-processing:

- The dataset contains 4552 ASIN with titles in the form of long scripts. For these cases, titles are extracted from the scripts by splitting the scripts by line breaks and

finding lines that start with "Amazon.com:." Example of lines in scripts containing title:

- `"Amazon.com: Sonic X: Chaos Emerald Chaos (Season 2): Jason Griffith, Dan Green, Mike Pollock: Movies & TV Title : "Sonic X: Chaos Emerald Chaos (Season 2)"`
- `"Amazon.com: Dragon Ball Z: Season 5 (Perfect and Imperfect Cell Sagas): Christopher Sabat, Sean Schemmel, Dameon Clarke, Chris Cason: Movies & TV" Title : "Dragon Ball Z: Season 5 (Perfect and Imperfect Cell Sagas)"`

In the above examples, the movie title is in dark red font. By analyzing the pattern, we observe that the lines consist of several parts separated by ":" in the following sequence "<Amazon.com> : <title> : <actor/actress> : <category>." However, the part <actor/actress> is not always available, and some titles also contain ":" in the title text. Therefore, we applied some rules in extracting the titles. There are 203 ASIN which script do not contain titles. We added them to the previously mentioned 67 ASIN without titles, and the total becomes 270, which is 0.15% from total ASIN.

- Convert titles to lower case, remove spaces at the beginning and the end.
- Re-arrange articles such as: "the," "a," "an," "le," "la," "les," "il," and "l' " to the beginning of the titles.
- Remove "vhs," "[vhs]," ": widescreen edition," "dvd," and "[dvd]" from the end of titles.
- Remove inconsistencies the same way as explained in Section 4.3.2.1 item 5.
- Adding a.k.a to the list of alternative titles by:
  - Converting numbers to roman numerals, as sometimes Amazon movie titles use the roman numerals.
  - Removing all spaces due to inconsistencies in writing the title, for example, "Goldeneye" and "golden eye."
  - Removing all strings inside parentheses (), as most are only additional information.

4.3.3 Experimental Results

With the above pre-processing, the string-to-string matching resulted in 16,622 (61%) MovieLens Id matched with ASIN, and 20,285 (11%) ASIN matched with MovieLens Id.

MovieLens id	MovieLens title	MovieLens pre-processed title		Amazon ASIN	Amazon title	Amazon pre-processed title
189	Reckless (1995)	reckless	Matched with	630222439X	Reckless VHS	reckless
189	**Reckless (1995)**	**reckless**		**6304018940**	**Reckless VHS**	**reckless**
189	Reckless (1995)	reckless		B005ZVINLM	Reckless	reckless
189	Reckless (1995)	reckless		B00WKWVHX0	Reckless	reckless

Table 4.8: Example of matching error due to several different movies having same title in Amazon dataset. In this table, only the boldfaced row is the true match. The others are totally different movies although having identical title.

Amazon ASIN	Amazon Movie Title
B00U4FYWXM	2001: A Space Odyssey Limited Edition SteelBook (Region Free UK Import) Limited to 2,000 copies OOP / Sold Out!
B00UGPQONQ	2001: A Space Odyssey [Blu-ray] by Warner Home Video

Table 4.9: Example of movie titles in the Amazon dataset with inconsistent formats.

We observed, there is some error in matches mostly due to:

- Some items have the same title in both MovieLens and Amazon dataset, but they are actually different movies. Example: MovieLens "Babe (1995)" in MovieLens dataset is matched to "Babe VHS" in the Amazon dataset. While checking the ASIN on the Amazon website, it resulted in "The babe," released in 1997. This type of error could have been reduced if movie year information were available in the Amazon dataset.
- Several different movies have the same title in the Amazon dataset, as shown in Table 4.8. Only the line with bold is the correct match, as the other ASIN are actually linked to different movies with the same title. Like the previous item, this type of error could have been avoided if movie year information were available in the Amazon dataset.

The analysis was also made to the non-matching cases; the major causes are as follows:

- Inconsistencies in the title format, especially in the Amazon dataset, for example, movie "2001: A Space Odyssey" has 2 ASIN with titles written differently, as in Table 4.9. It is difficult to distinguish the additional information which is "Limited Edition SteelBook (Region Free UK Import) Limited to 2,000 copies OOP / Sold Out!" in the first example and "[Blu-ray] by Warner Home Video" in the second example as they do not have a certain pattern. In the pre-processing, everything in "()" and "[]" was removed, but still, there is additional information in the text which were not removed since they are outside of "()" and "[]." Therefore, in this case, MovieLens Id 924 "2001: A Space Odyssey (1968)" did not match 2 ASIN in Table 4.9 even though they refer to the same movie.
- Movies not available in Amazon dataset (meta tv and movies), examples:

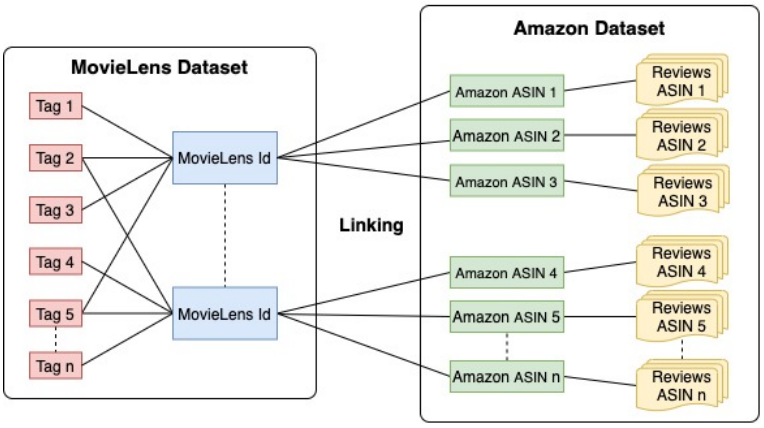


Figure 4.1: Relation between the linked datasets. Please note that this is the ideal case of the mapping assuming there is no duplicates in MovieLens ID. As explained in Section 4.3, there are small number of cases where movies have identical title but have more than one MovieLens ID

- Some movies are only in the form of bundles, which consist of several movies in one product, such as ASIN B001ILHYIQ "Desperately Seeking Susan / Mystery Date Double Feature," therefore, MovieLens Id 2369 "Desperately Seeking Susan (1985)" cannot be matched. However, if we search directly on the Amazon website, there is ASIN specific for this movie (B00INMHJGU).
- MovieLens ID 2373 "Red Sonja (1985)" does not have a match in the Amazon dataset; however, it is available on the Amazon web if searched directly.
- Some items are not available as movies but in another type of media, such as audio cd/music in the Amazon dataset. For example, MovieLens Id 106220 "Buried Alive (2013)" and MovieLens Id 106222 "Aziz Ansari: Dangerously Delicious (2012)," they are available as audio cd/music in Amazon dataset, not as movies.

4.4 Extracting Synonyms from Amazon Movie Reviews

Through this experiment, we would like to extract synonyms for user-tags from Amazon movie reviews. Based on the previous experiment, we have MovieLens movie Id mapped to Amazon ASIN; therefore, we can connect tags assigned to the MovieLens movie Id with the Amazon movie reviews. The relation between the data in datasets can be seen in Figure 4.1.

4.4.1 Dataset

The dataset used in this experiment are:

- The mapped MovieLens Id <-> Amazon ASIN which resulted from experiment in Section 4.3.

- Amazon movie reviews from the mapped Amazon ASIN.
  The movie reviews can be found in "Movies_and_TV.json"[5], which contains total of 8,757,407 non-empty reviews distributed to 182,010 ASINs. A very basic pre-processing were implemented, such as removing new line character, and some punctuations. Not all reviews will be used in this experiment as they are filtered to only reviews which connected with the mapped ASINs with length of words > 5. Also there are high number of duplicates for the reviews, due to some ASINs sharing the same movie reviews.

- User-tags from MovieLens
  Only tags connected to the mapped MovieLens Ids are used. Some MovieLens Ids do not have tags even though they are mapped to Amazon ASIN; therefore, they will be excluded.

By imposing all filters, removing duplicates, and getting intersections between all the above data, we end up with 126,328 reviews which connected to 12,791 MovieLens Ids and 3,221 tags.

4.4.2 Experimental Setup

The data are indexed using Elasticsearch. The code for initializing index mapping can be seen in Appendix A.2, Listing 2. Each document in the index contains 1 movie review with a length of > 5 words. The index has 3 fields:

- MovieLens Id
  This has 'keyword' type, and can have multiple values. Due to duplicates of same movie with different MovieLens Id, a review can relate to more than one MovieLens Id.

- Tags
  This field is a 'keyword' type and have multiple values, as one MovieLens Id can have more than one tags assigned to it.

- Review
  This field is a 'text' type, with standard tokenizer, stop words removal, and shingle filter. The term vectors is activated for this field. The shingle filter is used with

[5] http://deepyeti.ucsd.edu/jianmo/amazon/index.html

	Corpus 1 **Reviews retrieved for tag**	**Corpus 2** **All reviews in index**
Frequency of word (X)	Number of reviews retrieved for tag that contain term / shingle (X_1)	Number of documents that contain term/shingle (X_2)
Corpus size (N)	Number of reviews retrieved for tag (N_1)	Number of documents in the index (N_2)

Table 4.10: Parameters to calculate log likelihood with Equations (4.1) and (4.2).

setting of minimum shingle with length of 2, and maximum shingle length of 4, and including the unigrams. Therefore the term vectors contains the single terms and shingles up to length of 4. Note that, the stop words removal is only for the unigrams not the shingles, with the consideration to get gramatically correct phrases, for example "guardian of the galaxy" is more grammatically correct than "guardian galaxy."

To extract the synonym, we first retrieve reviews for each tag by searching from the index using keyword tag with minimum hits of 10. This means we only consider tags which have minimum 10 reviews to make the statistics more reliable. Therefore, there will be tags which do not have synonym retrieved.

For a given tag and for each term or shingle that appears in reviews retrieved for that tag, count the log likelihood score. The log likelihood function[6] as seen in Equations (4.1) and (4.2) adapted from [55] (denoted as G^2). This score will reflect on how often a specific term or shingle in the movie reviews co-occurs with the related tag. The higher the score, the more often the term or phrase co-occur with the tag which have higher possibility of those to become synonym for the tag.

$$G^2 = 2 \sum_{i=1}^{k} X_i \, ln \left(\frac{X_i}{E_i} \right) \tag{4.1}$$

$$E_i = \frac{N_i \ \sum_{i=1}^{k} X_i}{\sum_{i=1}^{k} N_i} \tag{4.2}$$

k is number of corpus, which is 2 in this case, X is the observed values, and N is the number of items in the corpus. Please see Table 4.10 for more detail information about the parameters.

After calculating the log likelihood for each term in reviews retrieved for the given tag, the resulted terms or shingles are sorted by log likelihood score from highest to lowest. A higher score means a higher likelihood of being a synonym for the tags.

[6] http://ucrel.lancs.ac.uk/llwizard.html

4.4.3 Experimental Results

Example of top-10 retrieved for tag 'space battle' is listed in Table 4.11. From this example, we can observe redundant information such as there is shingle with length of 2 "star wars," but there are also single terms "star" and "wars" that do not add information or variation to the list. Therefore, an additional step to remove the redundancy is applied by going through the sorted list one by one from the highest rank and removing redundant words/phrases as it goes, in this example, "star wars" will be kept while "star" and "wars" will be excluded. Another example is for "jedi," "the jedi," "of the jedi," and "return of the jedi," only the last one is being kept since the previous three words or shingles are contained in the last shingle. The step by step explanation is as follows:

- Initially, retrive more than top-10 result (in this case, top-30 is used), so that we can still get top-10 results after removing some redundant terms/shingles from the list.
- Based on observation, there are shingles which begins with "the" and some ends with "is" which can cause duplicates for example "cinderella story" and "the cinderella story," or "Drew Barrymore" and "Drew Barrymore is." In this case this algorithm will keep the one without beginning "the" and without ending "is." Therefore, check for this duplication cases and exclude those from the list.
- From the updated list, for each term/shingle (x) starting from highest rank to lowest rank except for the first term/shingle:
  - Check all previous terms/shingles which token length are smaller than x starting from smallest length l until $length(x)-1$. If the term/shingle contained in x then exclude them from the list
  - Check all previous terms/shingles which token length is larger than x starting from $length(x)+1$ until maximum token length(which is 4 in this case), if x contained in at least one of those the term/shingle, exclude x from the list.
  - There is a variable to keep count of the deleted or added item, until we get top-10 results, break the process.

The final top-10 result examples after removing redundancies can be seen in Table 4.12 and Table 4.13.

The results show that by applying this method, we can manage to get terms which are reasonably related to the tags. However, they are mostly not synonyms.

	Top-10 results for 'space battle'	
1	star wars	1798.68
2	wars	1681.90
3	jedi	1161.87
4	star	1057.51
5	lucas	1046.62
6	the jedi	942.02
7	of the jedi	900.87
8	luke	891.60
9	return of the jedi	885.87
10	vader	759.59

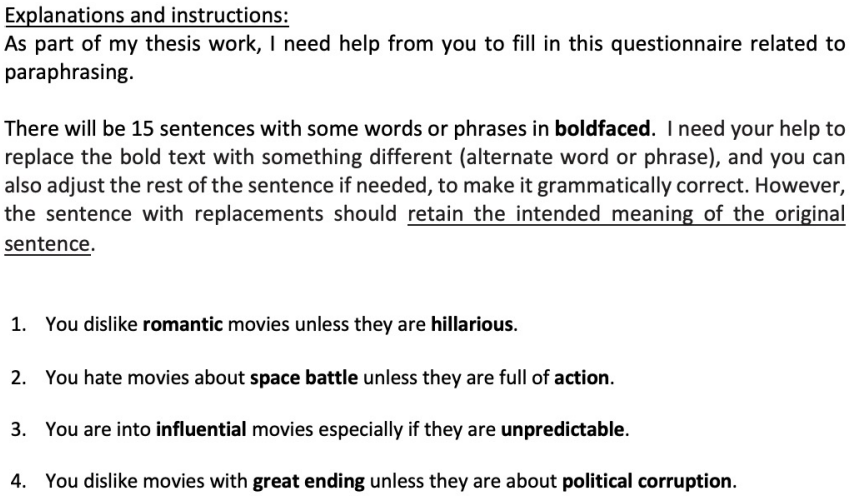
Table 4.11: Top-10 result based on log-likelihood score for 'space battle' tag.

Top-10 Result Non-Adjective Tags						
	romantic		**animated**		**gruesome**	
1	drew barrymore	1906.88	wallace and gromit	130.80	anthony hopkins	1571.15
2	cinderella story	1352.44	region 2	87.35	spanish	1430.66
3	prince	1005.18	sheep	71.78	hannibal	1418.51
4	fairy	683.77	dvd player	54.02	fantasy	1203.86
5	stepmother	654.93	w g	53.20	jodie foster	1180.10
6	danielle	640.00	close shave	52.06	del	1102.71
7	ever after	586.04	players	49.61	pan's labyrinth	1068.02
8	sandra bullock	581.16	wrong region	43.83	fairy	1029.64
9	huston	571.67	wendolene	43.60	lambs	950.69
10	dougray	540.70	not work	43.50	silence of the	920.86

Table 4.12: Example of top-10 results for adjective tags after removing redundant results, sorted based on the log-likelihood scores.

Top-10 Result Non-Adjective Tags						
	comic		**based on a video game**		**space battle**	
1	marvel	2863.31	resident evil	3136.76	star wars	1798.68
2	fun	1095.13	video game	2107.91	lucas	1046.62
3	chris pratt	1059.71	milla	1914.01	luke	891.60
4	groot	977.85	games	1792.94	return of the jedi	885.87
5	his	888.57	lara	1692.72	vader	759.59
6	her	782.40	angelina jolie	1649.73	ewoks	755.20
7	guardians of the galaxy	749.72	croft	1615.11	trilogy	742.72
8	film	737.69	tomb raider	1578.35	empire	735.80
9	who	684.49	jovovich	1559.07	original	680.89
10	can't wait	677.60	zombies	1553.22	darth	623.95

Table 4.13: Example of top-10 results for non-adjective tags after removing redundant results, sorted based on the log-likelihood scores.


Explanations and instructions:
As part of my thesis work, I need help from you to fill in this questionnaire related to paraphrasing.

There will be 15 sentences with some words or phrases in **boldfaced**. I need your help to replace the bold text with something different (alternate word or phrase), and you can also adjust the rest of the sentence if needed, to make it grammatically correct. However, the sentence with replacements should retain the intended meaning of the original sentence.

1. You dislike **romantic** movies unless they are **hillarious**.
2. You hate movies about **space battle** unless they are full of **action**.
3. You are into **influential** movies especially if they are **unpredictable**.
4. You dislike movies with **great ending** unless they are about **political corruption**.

Figure 4.2: Screenshot of the questionnaire.

To analyze of how many synonyms we can possibly get from this method, top-10 results for 20 adjectives and 20 non-adjectives are extracted. They are reviewed to check if any of the top-10 result contains synonyms. The results are:

- 3 out of 20 adjective tags get synonym, i.e., "touching: sad," "surreal: fantasy," "brutal: violence."
- 1 out of 20 non-adjective tags get synonym, i.e., "bollywood: india."

The result indicates the possibility of getting synonyms for adjective tags (15%) are slightly higher than non-adjective tags (5%). However, both are considered very low number. For the case of non-adjective tags it is especially difficult to get synonym for places, names, and numbers.

4.5 Questionnaire Related to paraphrasing

With the experiment in Section 4.4, we managed to extract words or phrases related to the tags; however, not close to synonyms. A questionnaire is made to gather input from human subjects as to how they would paraphrase the tags or the sentence pattern if given user preference statements generated by the template-based approach. The screenshot of the questionnaire is in Figure 4.2.

4.5.1 Experimental Setup

The questionnaire contains 15 sentences generated from the adjusted template with each sentence contains randomly picked tags (total of 30 tags), where the tags are boldfaced. The person should replace the bold text with something different (alternate word or phrase). They also can adjust the rest of the sentence if needed, for example to make it more grammatically correct, or for any other reasons. However, the sentences with replacements should retain the intended meaning of the original sentences.

Seven persons participated in answering the questionnaire.

4.5.2 Experimental Results

From the feedback, analysis made to the replacements for tags suggested from the feedback, which are divided into 3 categories:

- From other tags: the replacement for tags are from other available tags in the dataset.
- Not from tags: the replacement for tags are outside of available tags in the dataset.
- No changes: no changes made by the respondent.

The summary can be seen in Figure 4.3. By accumulating the counts for each category, the proportion of each category from all replacements feedback is: 43 % of "From other tags," 40% of "Not from tags," and 17% of "No Changes."

Some people suggest changes in the sentence pattern, with several reasons as in Table 4.14. The number of changes is not large, where most of them are very subtle and still keeping the main pattern.

From the feedback results, we see that the biggest proportion of tags replacement suggested by the respondents are available as other tags in the dataset itself. Therefore, the better approach might be to find similar word/phrase for tags from the other available tags, instead of from the movie reviews.

4.6 Extracting Synonyms from Other Tags

As seen in the result of questionnaires, we see that almost half of the tags can be replaced with another similar tags available in the dataset itself. Therefore, in this experiment we want to extract those similar words/phrases.

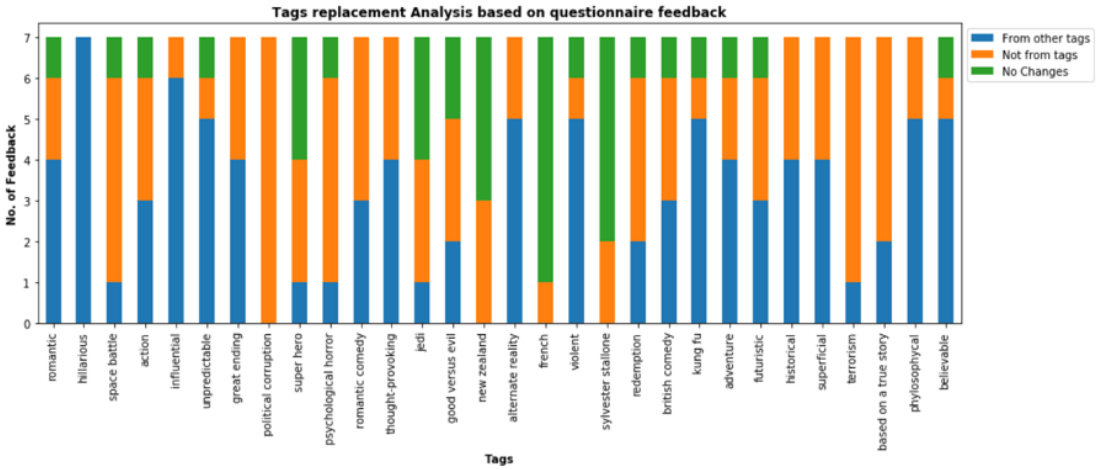


Figure 4.3: Counts of each category for the tags replacement based on the questionnaire feedback.

	Cause of changes to sentence pattern	Example	
1.	Tag replacement having other POS-tag type. For example from Adjective to Verb.	From	You love movies with **romantic comedy** genre, especially if they are **thought-provoking**.
		To	You love movies with **humorous romance** genre, especially if they **make you think**.
2.	Tag replacement having opposite sentiment for the user preference of the original.	From	You love **historical** movies especially if they are not **superficial**.
		to	You love movies **based on true stories**, especially if they are **authentic**.
3.	Being creative	From	You dislike movies with **great ending**, unless they are about **political corruption**.
		to	You only like movies with an **awesome ending** if they have something about **corrupt politicians**

Table 4.14: The cause of changes of sentence pattern proposed in the feedback.Tags and the tags replacements are in boldfaced, while the part of sentence that were changed is in red font.

4.6.1 Experimental Setup

The tags are taken from the MovieLens dataset, filtered to only the tags which are assigned to at least 5 movies and assigned by at least 2 users. The tags are also filtered from inappropriate content. That leaves us with 5610 tags. Movies which are used are only the ones connected with these 5610 tags.

For a given tag, and for each candidate tag, log-likelihood score is calculated using Equation (4.1) and Equation (4.2), with variables in Table 4.15. The higher the score is, the more that candidate tags occurs together with the given tag, which gives higher possibility of the candidate tag to become synonym or similar word/phrase.

4.6.2 Experimental Results

Example of top-10 results can be seen in Tables 4.17 and 4.16. These results are better in comparison with the results from extracting similar words/phrases from Amazon movie reviews as in Tables 4.12 and 4.13. With this method, we managed to get very reasonable similar representation of the tags, instead of just related words/phrases.

	Corpus 1 **Movies tagged with tag**	**Corpus 2** **All movies**
Frequency of word (X)	Number of movies tagged with both tag and candidate tag (X_1)	Number of movies tagged with candidate tag (X_2)
Corpus size (N)	Number of movies tagged with tag (N_1)	Total number of movies (N_2)

Table 4.15: Parameters to calculate log likelihood with Equations (4.1) and (4.2).

Top-10 result **Adjective Tags**						
	romantic		**animated**		**gruesome**	
1	romance	142.59	animation	277.42	disturbing	126.55
2	love story	84.37	cartoon	167.63	creepy	117.13
3	chick flick	76.95	disney	132.95	horror	107.33
4	romantic comedy	71.99	disney animated feature	126.61	menacing	106.44
5	funny	70.19	cute	119.49	tense	67.15
6	overrated	56.29	2d animation	110.45	visceral	65.61
7	love	52.98	talking animals	105.97	atmospheric	65.19
8	girlie movie	51.51	pixar	87.5	gory	51.93
9	happy ending	49.04	funny	79.28	ominous	47.08
10	boring	46.02	fun	78.57	cannibalism	41.52

Table 4.16: Top-10 similar words or phrases for adjective tags extracted from other available tags, sorted based on the log-likelihood scores.

Top-10 result **Non-Adjective Tags**						
	comic		**based on a video game**		**space battle**	
1	based on comic	41.7	video game adaptation	105.66	far future	29.42
2	comic book	39.0	adapted from:game	69.75	sci-fi	28.36
3	adapted from:comic	36.42	videogame	49.92	sci fi	27.84
4	action	35.38	resident evil	30.85	space	27.83
5	super-hero	32.92	video game	26.88	mark hamill	25.59
6	based on a comic	32.14	paul w.s. anderson	19.21	darth vader	25.59
7	superhero	30.81	computer game	18.2	luke skywalker	25.59
8	dark hero	25.51	over-the-top action	17.4	jedi	24.68
9	franchise	23.9	ridiculous characters	17.4	carrie fisher	22.78
10	marvel	22.9	milla jovovich	15.67	starship pilots	22.78

Table 4.17: Top-10 similar words or phrases for non-adjective tags extracted from other available tags, sorted based on the log-likelihood scores.

Please note that the result of this experiment will also be used in generating synthetic training data in Section 6.2. In generating the training data, the tags are split into 80/20 for train/test split, and the similar word/phrase for each tag in the train split only can be extracted from the train split itself to avoid leaking information from the test split to the model. In this case, the result will be slightly different as due to the splitting, the number of tags reduced to 4488 in the train split.

4.7 Summary

The template-based approach is initialized by developing basic-template on pairwise tag interactions. It takes a pair of user-tag with user preference level for each tag as input and generating a user preference statement as output.

Error analysis was performed to understand how well-suited the basic template is with the tags. We found that the basic-template seems to be suitable for most adjective tags; however, it shows significant errors when dealing with other tags categorized to other part-of-speech (POS). We analyzed the failure cases and identified new sentence patterns to handle them. We enhanced the basic template into the adjusted template by adding these newly created sentence patterns, which increase the success cases from 25% to 84%.

We attempted to improve the adjusted-template further by replacing tags with their synonyms, which initially performed through mining synonyms in movie reviews. However, this gives mostly "related words," which we can not use since they can likely change the meaning of the original sentence. We found that a better approach is by extracting synonyms from other tags, which gives very reasonable results of "similar words/phrases."

Chapter 5

Neural Paraphrasing Approach

As mentioned in Section 1.3, the second main line approach is to employ neural paraphrasing by experimenting on several different neural model architectures.

5.1 Task-Specific Neural Model

In this experiment, we use neural models which are built specifically for paraphrase generation task. Two state-of-the-art neural paraphrasing models chosen are Stacked Residual LSTM Networks [7] and Deep Generative Framework for Paraphrase Generation [5].

The first neural model was trained using the Quora dataset, and the second neural model was trained using the MSCOCO dataset. The test data for both models are user preference statements generated from the adjusted template in the template-based approach (Chapter 3). We feed the user preference statements to the trained models, to get the paraphrases. The experimental setup and results are described in Section 6.1.

Observing the results, the generated sentences from these two models can not be considered as paraphrases since they do not retain the same meaning as the originals, and are grammatically poor. We believe the underlying problem is related to the training data. The training data does not seem suitable for our task, e.g., it does not contain sufficient information on specific topics such as movies, and the size of training data is considerably small for training deep learning models properly. Another thing is that our test data have particular patterns adapted from the pairwise tag interactions (explained in Section 2.1.3) that the model needs to understand; however, they cannot get this information from the available training data.

The challenge of acquiring large labeled data suitable for a specific task is common in deep learning for NLP tasks [45]. Some labeled training data might be effective for a

specific task, but not necessarily equally effective when used for other tasks. Therefore, we switch our focus from task-specific neural models to exploring the transfer learning method, which will be explained in the next section.

5.2 Pre-Trained General Purpose Neural Model

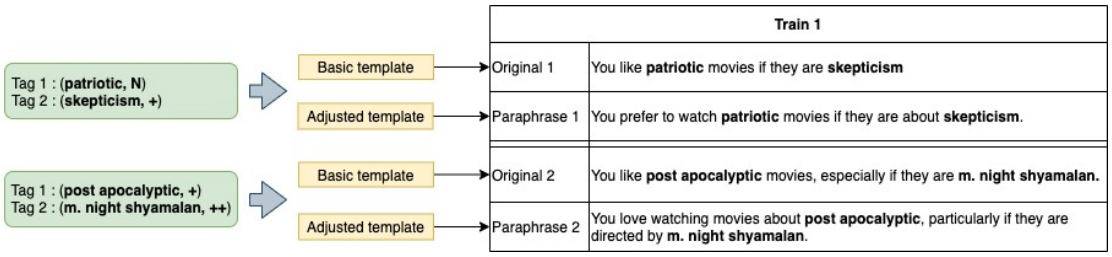
As also mentioned in Section 2.4, transfer learning in the NLP task is getting more attention nowadays. It involves the use of models which have been pre-trained with large unlabeled training data from various corpora to grasp knowledge about texts in general. This approach is attractive since there are massive unlabeled text data available online [9]. The knowledge learned by the pre-trained model can be transferred for use in specific tasks, by performing fine-tuning mechanisms to the model.

Since the pre-trained models already have a comprehensive knowledge of general representation of language [45], e.g., grammar, language contexts, and semantics, the fine-tuning steps can usually be done using a much smaller dataset with enough data to make the pre-trained model to understand the new task.

In this approach, GPT-2 [8] and T5 [9] pre-trained models are chosen to be fine-tuned for paraphrasing task. Both of the models are briefly explained in Section 2.4.

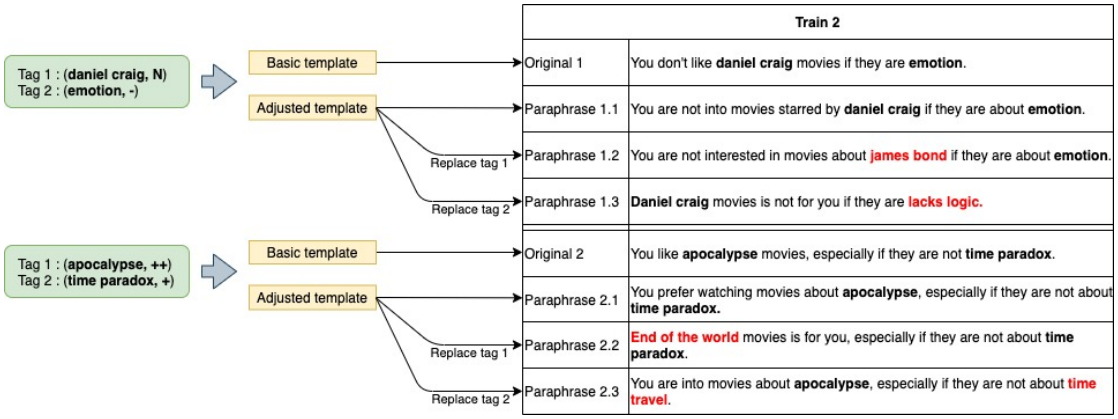
The training data consists of pairs of original sentence and paraphrase. Ideally, gathering human-generated paraphrases would potentially result in better training data. However, due to limited time, it is not feasible to collect a large dataset of human paraphrases in this experiment. Also, that would not be a scalable approach. This experiment will use synthetic training data, which is the user preference statements generated from the template-based approach. The original sentence will be generated from the basic template, and the pair paraphrase will be generated from the adjusted template. There are 3 types of datasets experimented for fine-tuning:

- Train 1
  Each train data will consist of original sentence generated from basic template, and one paraphrase generated from adjusted template without any changes on the tags; see Figure 5.1 for illustration.
- Train 2
  Each train data will consist of original sentence generated from basic template, and three paraphrases generated from adjusted template. The first paraphrase is without any changes to the tags. The second paraphrase is with tag 1 replacement. The third paraphrase is with tag 2 replacement; see Figure 5.2.



Train 1	
Original 1	You like **patriotic** movies if they are **skepticism**
Paraphrase 1	You prefer to watch **patriotic** movies if they are about **skepticism**.
Original 2	You like **post apocalyptic** movies, especially if they are **m. night shyamalan.**
Paraphrase 2	You love watching movies about **post apocalyptic**, particularly if they are directed by **m. night shyamalan**.

Figure 5.1: Example of train 1 dataset. Tags are in boldfaced.



Train 2	
Original 1	You don't like **daniel craig** movies if they are **emotion**.
Paraphrase 1.1	You are not into movies starred by **daniel craig** if they are about **emotion**.
Paraphrase 1.2	You are not interested in movies about **james bond** if they are about **emotion**.
Paraphrase 1.3	**Daniel craig** movies is not for you if they are **lacks logic.**
Original 2	You like **apocalypse** movies, especially if they are not **time paradox**.
Paraphrase 2.1	You prefer watching movies about **apocalypse**, especially if they are not about **time paradox.**
Paraphrase 2.2	**End of the world** movies is for you, especially if they are not about **time paradox**.
Paraphrase 2.3	You are into movies about **apocalypse**, especially if they are not about **time travel**.

Figure 5.2: Example of train 2 dataset. Tags are in boldfaced. Red font shows replaced tag.

MSCOCO	
Original 1	a man in head phones is brushing his teeth
Paraphrase 1	a man is brushing his teeth while listening to headphones
Original 2	a woman standing next to a little girl playing a game on nintendo wii
Paraphrase 2	a woman playing on a wii system with a little girl

Figure 5.3: Example of MSCOCO dataset.

- Mixed Train 2 - MSCOCO
  This is mixed of Train 2 and MSCOCO [56] dataset, which has been pre-processed and filtered by Prakash et al. [7]. MSCOCO dataset consists of pair of human-annotated image captions, which describe the same image. The example of MSCOCO dataset is shown in Figure 5.3.

The process flow of the fine-tuning is illustrated in Figure 5.4. It is started with preparing the train data by adjusting its format for use in different models. The next step is to perform fine-tuning using different parameters and training data. Afterward, paraphrases will be generated from all fine-tuned models using test data as input. We will score the performance of each model using the method explained in Section 6.2.2 and select the models with the best performance. At the end of the process, feedback from 5 people will be collected to evaluate paraphrases generated from the best models. In the

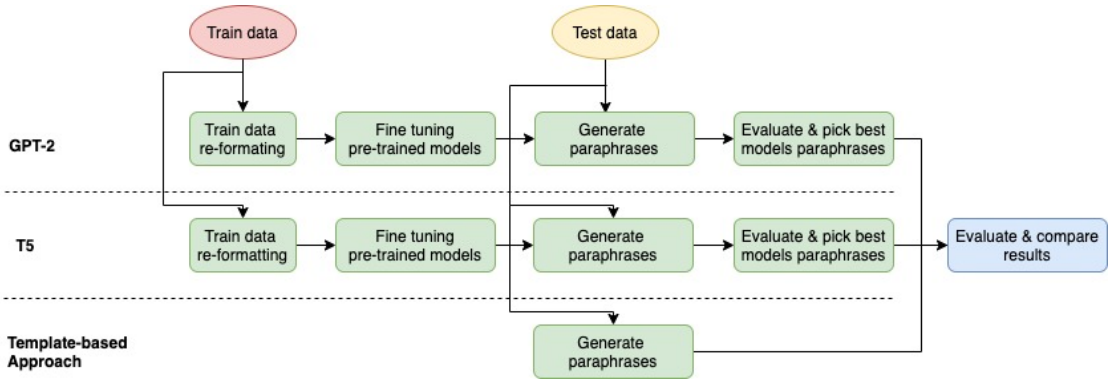


Figure 5.4: Fine-tuning process flow.

same questionnaire, we also collect human feedback for paraphrases generated from the template-based approach. Based on the feedback, these models and the template-based approach will be compared.

In this thesis work, automatic quantitative scoring methods such as BLEU [57], METEOR [58], and ROUGE [59] are not explored since they are not suitable for use, due to the paraphrases generated by the templates are not ideal "ground truth" for the original sentences. The methods are more suitable for cases where the paraphrases in the training data are human generated.

The evaluation process is dependant on human judgment, which involves scoring the generated paraphrases one by one. Considering time limitation, we have to limit the number of parameters for fine-tuning to a reasonable number of combinations. For example specific on training dataset, we decided to focus in trying different training dataset types as explained above, while keeping the amount of training data consistent.

More detailed explanation about the experiment is explained in Section 6.2.

Chapter 6

Neural Paraphrasing Results

6.1 Task-Specific Neural Model

This section describes details regarding the approaches briefly mentioned in Section 5.1, including experimental setups, datasets, and the obtained results based on running them on default parameters. However, since the results from the two task-specific neural models are far from expected, no further fine-tuning was done. Our experiment shows that the task-specific model approach is not suitable due to insufficient labeled training data relevant to this task. We then put more focus in our work toward fine-tuning the general-purpose pre-trained models discussed in Section 6.2.

6.1.1 Deep Generative Framework for Paraphrase Generation

6.1.1.1 Dataset & Experimental Setup

In this experiment, we use implementations in GitHub repository[1] from the author of [5], with some code adjustments.

We also use the training data shared in this repository. The training dataset is originally from Quora[2], released in January 2017, which has been pre-processed and filtered by the author [5]. It consists of 50k pairs of questions and paraphrases.

For the test data, we use user preference statements generated from our adjusted template from the template-based approach.

[1] https://github.com/arvind385801/paraphraseGen
[2] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

Original sentence from test data	Paraphrase
You don't prefer **avant garde** movies especially if they are nerdy.	Should we not watch Which is the best **idea**?
You don't like **movies** about talent unless they are movies by sofia coppola.	which life is the best **movie** about **movies** ?
You prefer movies starred by lance henriksen if they are about **war** hero.	What are some examples of **war** in India?
You don't prefer teen movies unless they are about **asia** argento.	What is being a house **language**?

Table 6.1: Example of generated paraphrases using Deep Generative Framework for Paraphrase Generation. The boldfaced texts show related words. Showing that the model can catch a little bit of the context from the original sentences; however, results are still very far from the original sentence.

The model was trained using default parameters without any changes. It was run on GPU server at the University of Stavanger with GPU specification of Tesla V100-PCIE-32GB.

6.1.1.2 Experimental Results

Some examples from generated paraphrase using Deep Generative Framework are shown in Table 6.1. We see that the generated paraphrases do not seem relevant to the original sentences in our test data. Even though the model can catch some ideas (in bold text) from the original sentences, the results have different meanings from the original sentences. In addition to that, we discover that the model is generating paraphrases in the form of questions. We believe that the reason behind this problem is related to the training data.

The training data is relatively small in size and not enough variety in topics and forms so that the trained model cannot generalize well when given test data with another format and with another topic, i.e., user preference statements related to movies. It can be seen that the generated sentences are all in the form of questions due to training data from Quora are all in the form of questions. Just by checking roughly the text in the training data, there is only 1.7% content about "movie," "film," or "genre," therefore, the model possibly did not get a good knowledge of the topic related to movies.

6.1.2 Stacked Residual LSTM Networks

6.1.2.1 Dataset & Experimental Setup

The code implementation is reusing the author's [7] implementation from GitHub page[3]. The code was run without any changes of parameters.

The model was trained on MSCOCO dataset with 331,163 pairs of original sentence and paraphrase.

Test data is using sentences generated from our adjusted template from the template-based approach.

[3] https://github.com/iamaaditya/neural-paraphrase-generation/tree/dev

Original	Paraphrase-1
You are not into movies about avant garde especially if they are nerdy.	a group of people sitting around a table with a
You prefer movies starred by lance henriksen if they are about war hero.	a table with a pair of scissors
You don't prefer teen movies unless they are about asia argento.	a group of people sitting around a table with a
You don't like movies about talent unless they are movies by sofia coppola.	a group of people sitting around a table with

Table 6.2: Example of generated paraphrase using Stacked Residual LSTM.

6.1.2.2 Experimental Results

Generated paraphrases results from this experiment can be seen in Table 6.2. The model does not seem to be able to generate paraphrases for our test data. The generated sentences contain many repetitions, and the model does not seem to understand the context.

We believe this is due to similar reasons as the previous model that the training data is not suitable for this task in both insufficient size and the coverage of the topics, therefore the trained model cannot generalize well when tested on sentences with different forms or topic.

6.2 Pre-Trained Model Fine-Tuning

This section describes experiments in fine-tuning the general-purpose pre-trained neural models for use in paraphrasing task.

As the pre-trained models have been trained using very large data, they are supposed to have good understanding of language semantics, grammar, and long term dependencies in language. Our goal in the fine-tuning process is to make the model understand that their new task is to generate paraphrases. The fine-tuning is performed by applying several different parameters and dataset types to the models.

6.2.1 Dataset

For the fine-tuning purpose, we use synthetic data for the main dataset generated using the template-based approach.

To generate the main dataset, movie tags are split into 80/20 train/test split.

6.2.1.1 Training Data

The training data is generated using only the tags from 80% split. There are two types of main training data:

1. Train 1
   For each randomly selected pair of tags, generate 50k pairs of:
   - Original: sentence with basic template.
   - Paraphrase: sentence with adjusted template.

   The ilustration is in Figure 5.1.

2. Train 2
   For each randomly selected pair of tags, generate 50k pairs of:
   - Original: sentence with basic template.
   - Paraphrase 1: sentence with adjusted template.
   - Paraphrase 2: sentence with adjusted template, by changing only 1^{st} tag with similar tag.
   - Paraphrase 3: sentence with adjusted template, by changing only 2^{nd} tag with similar tag.

   Note that similar tags are the tags resulted in the method as describe in Section 4.6, which is randomly chosen from top-3 results with weights of 0.7, 0.2, and 0.1 respectively from the first to the third rank. The ilustration is in Figure 5.2.

Aside from the main train data, it is also experimented on the following datasets:

- Mixed of train 2 and MSCOCO dataset
  This dataset is a combination of the above-explained train 2 dataset with the MSCOCO dataset, which contains 331,163 pairs of original sentence and paraphrase. This dataset introduces the model with varieties of original-paraphrase pairs other than the ones generated by the template-based approach. We call this "mixed train 2 - MSCOCO" dataset. Ilustration of MSCOCO dataset is in Figure 5.3.

6.2.1.2 Test Data

The test dataset consists of single sentences (original sentences) generated using the basic template using tags from the 20% test split tags without their paraphrases.

6.2.2 Performance Evaluation Method for Fine-Tuned Models

As explained in Section 5.2, the quantitative scoring method such as BLEU [57], METEOR [58], and ROUGE [59] are not used in this work as there is no ideal "ground truth"

in the dataset. Therefore this evaluation method is created to measure the performance of each fine-tuned model.

The evaluation is based on observation from the author to each of the generated paraphrase results from the fine-tuned models. Each generated paraphrase will be scored based on 3 criteria, each with a score of 1-5:

- Grammar: Is it grammatically correct?
- Relevance: To what extent it retains the same meaning as the original?
- Creativity: How much can it introduce replacements to parts of the sentence which has relevant meaning? This criterion will give more points to models with more relevant words/phrases replacements.

We observe inconsistencies in the quality of the generated paraphrases, which also mentioned in [60]. For example, one model can generate good paraphrase for sentence A; however, poor paraphrase for sentence B. There are also cases where the model generates different qualities of paraphrases for the same original sentence. Therefore, each model will be scored against generated paraphrases on 5 test sentences (one paraphrase for each test sentence), and the score will be averaged.

After selecting the best-performing models based on the scores in this evaluation, more objective feedback from 5 people will be gathered, and the results will be evaluated with the method explained in the next section.

6.2.3 Evaluation for the Selected Best Models

Best models are selected based on the highest score based on the previous method. More objective evaluation is gathered by sending out questionnaires, where the respondents will score paraphrases results from the selected best models.

The scoring method will be based on 3 criteria, each with a score of 1-5:

- Grammar: Is it grammatically correct?
- Relevance: To what extent it retains the same meaning as the original?
- Sounding natural: How natural does the statement sound?

The feedback is gathered from 5 people, where they are requested to review 5 generated paraphrases from each model. We then calculate the average scores and compare the best models. The more detailed explanation about the questionnaire is in Section 6.2.6.

Note that there is a slightly different criterion in this method compared to the previous method in Section 6.2.2, i.e., "creativity" vs. "sounding natural." It is due to during initial evaluation, the focus is to find models that can generate paraphrase that is able to say the original sentence in different ways while leaving out the judgment on "sounding natural" for more objective input from 5 different people.

6.2.4 GPT-2 Fine-Tuning

The GPT-2 fine-tuning is inspired from the work introduced by Witteveen and Andrews [60], where they demonstrated that their approach can be used for utilizing pre-trained large language model (GPT-2) for paraphrase generation through fine-tuning.

6.2.4.1 Experimental Setup

There are 3 types of GPT-2 pre-trained models that are released by OpenAI, which is 124M, 355M, and 774M. These numbers in the model names show how many parameters the model has. In this experiment, I am using the 355M and 774M models. The library used is GPT-2-simple[4] library, which wraps the model fine-tuning and generation scripts for OpenAI's GPT-2[5].

It was run on the GPU server at the University of Stavanger with the specification of Tesla V100-PCIE-32GB. It needs to be run in a virtual environment since it is using the older version of Tensorflow (version 1.15) than the one installed on the server, and there are some other packages needed to be installed before fine-tuning.

The sizes of the checkpoints are 1.5 GB for 355M and 3 GB for the 774M.

6.2.4.2 Train and Test Data Format

- Train data format
  The train data format, generally is written in the following format:
  "`original sentence 1.>>>>>>paraphrase sentence 1.`
  `<|end of text|>`
  `...` "
  Where ">>>>>>" was chosen as a token to separate the original and the paraphrase sentence to give an indication to the model that paraphrase generation

[4] https://github.com/minimaxir/gpt-2-simple
[5] https://github.com/openai/gpt-2

Parameters	Values
Pre-trained model size	355M, 774M
Optimizer and learning rate	(Adam, 0.00002), (SGD, 0.0006)
Dataset	train 1, train 2, mixed train 2 - MSCOCO
Steps (epoch)	1, 3, 5, 10, 15, 20, 25, 50, 75, 100, 150, 200, 250, 500, 750, 1000
Temperature	0.7, 1.0, 1.2

Table 6.3: Fine-tuning datasets and parameters.

starts after this token. Token "<|end of text|>" is used to mark the end of each pair of data.

Specific for train 2 data, since each original sentence has 3 paraphrases, it is written in the following format:

"`original sentence 1.>>>>>>paraphrase 1 sentence 1.`
`<|end of text|>`
`original sentence 1.>>>>>>paraphrase 2 sentence 1.`
`<|end of text|>`
`original sentence 1.>>>>>>paraphrase 3 sentence 1.`
`<|end of text|>`
`...` "

- Test data format

  The test data is in the following format:

  "`test sentence 1.>>>>>>`
  `test sentence2.>>>>>>`
  `...` "

6.2.4.3 Fine-Tuning

The parameters for fine-tuning are listed in Table 6.3. As explained in Section 5.2, we need to limit the number of parameters combination to a reasonable number, since the evaluation is based on human judgments, which will take time if the number of paraphrases to be evaluated becomes to many. Therefore, the fine-tuning is performed in two stages.

At the first stage, we attempt to obtain the best values for the optimizer, learning rate, and temperature, to be used as fixed parameters in further fine-tuning. For each parameter, several trial values are tested while keeping all other parameters unchanged. In this stage, the best value selection is based on simple observations on which paraphrase results seem better than others. These parameters are selected from the first stage:

- Optimizer and learning rate: Optimizer and learning rate: Optimizer (adam,0.00002) (which is the default) gives a better result than (SGD,0.0006) when run for the same number of steps. With SGD, the model seems to learn at a slower pace, which can be observed as the model failed to learn the task and just generated random text.

- Temperature: 1.0 is chosen. Temperature is a parameter in the text generation phase, after the fine-tuning. It is a float number, where the randomness in generating the text is increasing when a higher temperature number is used.

The second stage fine-tuning is to tune on combinations of all remaining parameters that we want to focus on in the evaluation phase. These parameters are: (1) pre-trained model size, (2) dataset types, and (3) steps, while optimizer and temperature using fixed values that we chose in the first stage. The resulted fine-tuned models are given 5 test sentences to generate 5 paraphrases (one for each test sentence). Finally, the results are evaluated based on the method explained in Section 6.2.2.

6.2.4.4 Results Evaluation

Most of the fine-tuned models that used train 1 as dataset, generated almost the same paraphrases with the adjusted template without changing the tags. Therefore, only the best models from fine-tuned models which used train 2 and mixed train 2 - MSCOCO are selected.

Examples of generated paraphrases from different fine-tuned GPT2 models are shown in Table 6.4. The best models are selected for the following combination of models and datasets: (1) 355M and train 2; (2) 355M and mixed train 2-MSCOCO; (3) 774M and train 2; (4) 774M and train 2-MSCOCO. They are tuned on various numbers of steps from small numbers to 1000 steps. Explanation about the models are shown in Table 6.6, and the summary of scores is listed in 6.6.

From the results we observe:

- In general, the models seem to understand that their new task is to generate paraphrase. Most of the models can generate paraphrases that have different patterns from the original test sentences and still retain the intended meaning and the pairwise tag interactions to some extent. They also managed to introduce changes for the tags with new words and phrases, i.e., “espionage” to “secret agent,” “fighting the system” to “killing the president,” and “scandal” to“sensational.” However, the models seemed to capture the style of the adjusted template patterns

Original	**You don't like action thriller movies if they are espionage.**
GPT2_774M_train2_50	You don't like movies about action thriller if they are about secret service.
GPT2_355M_train2_50	You don't like action thriller movies if they are spy
GPT2_774M_mixed_100	You don't prefer to watch movies about action thriller if they are about secret agent.
GPT2_355M_mixed_250	You dislike watching action thriller movies if they are about espionage.
Original	**You like bollywood movies, especially if they are not fighting the system**
GPT2_774M_train2_50	You like to watch bollywood movies, especially if they are not about killing the president.
GPT2_355M_train2_50	You enjoy movies about bollywood, especially if they are not about fighting the system.
GPT2_774M_mixed_100	You like watching movies about bollywood, particularly if they are not about fighting the system.
GPT2_355M_mixed_250	You enjoy watching movies about bollywood if they are not about fighting the system.
Original	**You like cannibalism movies, unless they are intelligent thriller.**
GPT2_774M_train2_50	You love watching using senior budget movies, except if they are intelligent thriller.
GPT2_355M_train2_50	You like to watch movies on extreme nature movies, except if they are intelligent horror.
GPT2_774M_mixed_100	You prefer movies about cannibalism, unless they are about intelligent thriller.
GPT2_355M_mixed_250	You would watch movies about cannibalism, except if they are intellectually stimulating.
Original	**You don't like scandal movies if they are artificial human.**
GPT2_774M_train2_50	You're not interested in movies about scandal if they are artificial human scripted.
GPT2_355M_train2_50	You view movies about murder if they are about artificial human.
GPT2_774M_mixed_100	You don't like to watch sensational movies if they are about artificial human.
GPT2_355M_mixed_250	You dislike watching documentaries about organ transplant if they about artificial human.
Original	**You don't like anthropomorphic movies, especially if they are fairy tales.**
GPT2_774M_train2_50	You hate to watch fox movies, especially if they are about fairy tales.
GPT2_355M_train2_50	You don't like to watch movies about anthropomorphic creatures.
GPT2_774M_mixed_100	You don't prefer to watch anthropomorphic movies, especially if they are about greek mythology.
GPT2_355M_mixed_250	You're not into movies about anthropomorphic creatures, especially if they are dystopian.

Table 6.4: Examples of the generated paraphrase results with different parameters or set up from GPT-2 fine-tuning.

Model name	**Explanation of parameters**
GPT2_774M_train2_50	dataset: train 2; pre-trained model: 774M; steps: 50
GPT2_355M_train2_50	dataset: train 2; pre-trained model: 355M; steps: 50
GPT2_774M_mixed_100	dataset: mixed train 2 - Mscoco; pre-trained model: 774M; steps: 100
GPT2_355M_mixed_250	dataset: Mixed train 2 - Mscoco; pre-trained model: 355M; steps: 250

Table 6.5: Models explanation. Note: All models were fine-tuned using (Adam, 0.00002) and temperature of 1 when generating paraphrase text.

Model	**Avg. Score for 5 generated sentences**				**Average score**
	Relevance	**Grammatic**	**Creativity**	**Total Avg. Score**	
GPT2_774M_train2_50	4,4	4,4	2,8	11,6	3,87
GPT2_355M_train2_50	4,2	4,4	2,8	11,4	3,8
GPT2_355M_mixed_250	4,2	4,8	2,2	11,2	3,73
GPT2_774M_mixed_100	4,4	4,4	2,2	11	3,67

Table 6.6: Score for the best model from each GPT-2 model size (355M, 774M) and train data (train2, mixed train2 - MScoco). Top-2 models (in red) are to be compared with T5 best models in Section 6.2.6.

and showed signs that they are memorizing patterns in training data. The signs of memorizing occurred from the early steps of fine-tuning. It might be caused by many repetitions in the training data since the number of unique sentence patterns in the adjusted template would not reach 50k. Radford et al. [8] also noticed the memorizing behavior when there are many duplications in their dataset.

- Lower number of steps gives better results.
- The scores for these 4 best models do not differ significantly. They seem to perform almost at the same level. The use of different datasets (train 2 and mixed train 2 - MSCOCO) does not show a significant difference in this experiment. None of the models is significantly more superior than the other. However, we believe the use of the mixed train 2 - MSCOCO dataset can still be explored further with experimenting on the proportion of train 2 and MSCOCO in the mixed dataset, to find which composition can result in improvements.
- The larger pre-trained models the lower steps needed to get the best performing models.

6.2.5 T5 Fine-Tuning

6.2.5.1 Experimental Setup

Google has released T5 pre-trained models[6] with different sizes: T5-Small (60 million parameters),T5-Base (220 million parameters), T5-Large (770 million parameters), T5-3B (3 billion parameters), and T5-11B (11 billion parameters). This experiment is using T5-large and T5-3B.

The library used for fine-tuning is T5 library package which implementation shared in Google Research Github page[7]. The fine-tuning ran on cloud TPU V2-8 through Google Colab[8].

The checkpoint size of the pre-trained models are 1.5 GB for large, and 5.3 GB for 3B model.

6.2.5.2 Train and Test Data Format

The train data can be in a form of TSV file with the format "<original><tab><paraphrase>," and it will be transformed to Python dictionary "{'original': 'original sentence', 'paraphrase': 'paraphrase sentence'}" before the fine-tuning.

For the test data, it is in Python list where each element is the original sentence text from the test data.

[6] https://github.com/google-research/text-to-text-transfer-transformer
[7] https://github.com/google-research/text-to-text-transfer-transformer
[8] https://colab.research.google.com/notebooks/intro.ipynb

Parameters	values
Pre-trained model size	large, 3B
Optimizer and learning rate	(Adafactor, 0.003)
Dataset	train1, train 2 mixed train 2 - MSCOCO
Steps (epoch)	1, 5, 10, 15, 20, 50, 75, 100, 150, 200, 250, 500, 1000, 1500
Temperature, Beam size	(1,1), (0,3)

Table 6.7: Fine-tuning datasets and parameters for pre-trained T5.

6.2.5.3 Fine-Tuning

The parameters for fine-tuning the T5 models are described in Table 6.7. We use the same approach as in GPT-2 fine-tuning, where the process is done in two stages.

The first stage is to find the best values for temperature and beam size to be used for fixed parameters in further fine-tuning. Beam size is an integer number >= 1 for the number of beams used for beam search (default = 1), while temperature is a float from 0 - 1.0 (default = 1.0). For each parameter, several possible values are tested while keeping all other parameters unchanged. In this stage, the best value selection is based on simple observations on which paraphrase results seem better than others. From the trials in the first stage, the temperature 1.0 and beam size one are selected. They are then used in stage 2 as fixed parameters.

At The second stage, we fine-tune with the combination of all remaining parameters: (1) pre-trained model size, (2) dataset, and (3) steps, that we want to focus on the evaluation stage while keeping fixed values for temperature and beam size to the values chosen in stage 1. After getting the fine-tuned models, then they are used to generate paraphrase by feeding 5 sentences from test data. Each fine-tuned model produces 5 paraphrases, one for each test data. Results are evaluated based on evaluation criteria explained in Section 6.2.2.

6.2.5.4 Results Evaluation

The examples of generated paraphrases from T5 fine-tuned models can be seen in Table 6.8 with model explanations in Table 6.9, and the scores in Table 6.10. Similar to the fine-tuning step for GPT-2, only the best models fine-tuned using train 2 and mixed train 2-MSCOCO are selected. This is due to the lack of variations obtained using train 1 models, where generated paraphrases are almost the same as the adjusted template, without undergoing further changes.

From the generated paraphrase examples, we observe:

- The models seem to understand the task which is to generate paraphrase. Most of the models were able to generate paraphrases with different sentence patterns than the test data, which retained the intended meaning of the original sentences and reflected the same pairwise tags interactions to some extent. Also, they managed to introduce new words or phrases to replace the tags, i.e., "bollywood" to "india," "cannibalism" to "cannibalistic plot," and "scandal" to "corruption." However, similar to the observation in GPT-2 results, there is an indication that the models are memorizing the patterns in training data. This is suspected due to repetitions of sentence patterns in the train data, which makes the models easily grasp the patterns.

- Like the results in GPT-2 fine-tuning, the scores for 4 best models for T5 also do not differ significantly. The use of different datasets (train 2 and mixed train 2 - MSCOCO) does not show a significant difference in the scores. Even though models with train 2 dataset resulted in slightly higher scores, we believe that the use of the mixed train 2 - MSCOCO dataset can be observed further by playing around with the composition to improve the results.

- Majority of the best models is using low number of steps except for the second model, so it can still be concluded that lower steps gives better results.

- The larger the pre-trained models parameter, the lower steps it needed to get best performing models.

6.2.6 Objective Evaluation and Comparison

In this section we are comparing the 2 best models from GPT-2 fine-tuning, 2 best models from T5 fine-tuning, and also template-based approach. A questionnaire was made to gather feedback from 5 different people to get more objective input on the paraphrase results from these 5 models. Please see Figure 6.1 for the screenshot of the questionnaire. The questionnaire consist of 5 original sentences, each with paraphrases from 5 different models. Where the order of paraphrases from the models are shuffled for each section. The respondent was asked to rate from 1-5 related to relevance, grammatical, and how natural the paraphrase. The evaluation method is explained in Section 6.2.3.

The result from questionnaire can be seen in Table 6.11, and based on observation:

- From the results, two models from T5 fine-tuning get the highest scores, which are quite far from the other models. They are the only models that receive higher scores than the average within every criteria. On the other hand, for all criteria,

Original	**You don't like action thriller movies if they are espionage.**
T5_3B_train2_75	espionage movies is not for you if they are full of action.
T5_large_train2_1000	you hate to watch movies full of action thriller if they are about espionage.
T5_3B_mixed_5	you don't like action thriller movies if they are about espionage.
T5_large_mixed_50	you hate to watch movies full of action thriller if they are about espionage.
Original	**You like bollywood movies, especially if they are not fighting the system**
T5_3B_train2_75	you enjoy watching movies from india, particularly if they are not fighting the system.
T5_large_train2_1000	you love watching movies about bollywood, especially if they are not full of anti-authoritarian propaganda.
T5_3B_mixed_5	you love movies about india, especially if they don't fight the system.
T5_large_mixed_50	you prefer movies from south korea, especially if they are not fighting the system.
Original	**You like cannibalism movies, unless they are intelligent thriller.**
T5_3B_train2_75	you like watching movies full of violence, except if they are intelligent thriller.
T5_large_train2_1000	you prefer watching movies about cannibalism, unless they contain intelligent thriller.
T5_3B_mixed_5	you like watching movies about cannibalism, except if they are about intelligent thrillers.
T5_large_mixed_50	you love watching movies with cannibalistic plot, unless they are smart thriller.
Original	**You don't like scandal movies if they are artificial human.**
T5_3B_train2_75	you don't prefer watching movies about scandal if they are about artificial human.
T5_large_train2_1000	you are not into movies about corruption if they are artificial human.
T5_3B_mixed_5	you do not like movies about scandal if they are about artificial human beings.
T5_large_mixed_50	you don't prefer to watch movies about scandal if they are about artificial human.
Original	**You don't like anthropomorphic movies, especially if they are fairy tales.**
T5_3B_train2_75	you dislike watching movies about anthropomorphic, particularly if they are full of fantasy.
T5_large_train2_1000	anthropomorphic movies is not for you, particularly if they are about fairy tale.
T5_3B_mixed_5	you dislike watching movies about anthropomorphism, especially if they are about fairy tales.
T5_large_mixed_50	you hate to watch movies about anthropomorphic, especially if they are about fairy tales.

Table 6.8: Examples of the generated paraphrase results with different parameters or set up from T5 fine-tuning.

Model name	Explanation of parameters
T5_3B_train2_75	dataset: train 2; pre-trained model: 3B; steps: 75
T5_large_train2_1000	dataset: train 2; pre-trained model: large; steps: 1000
T5_3B_mixed_5	dataset: mixed train 2 - MSCOCO; pre-trained model: 3B; steps: 5
T5_large_mixed_50	dataset: mixed train 2 - MSCOCO; pre-trained model: large; steps: 50

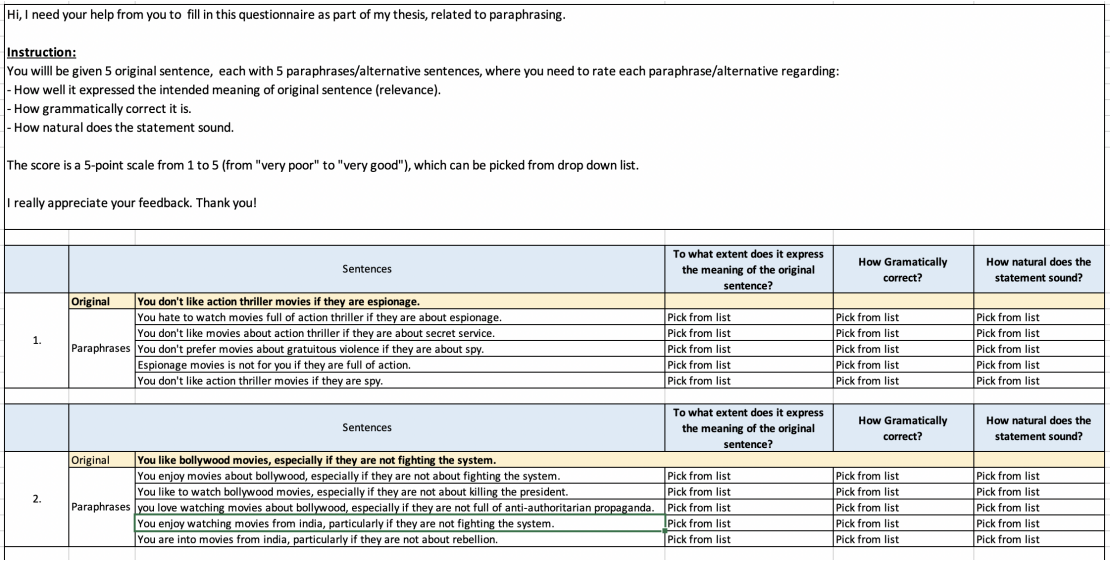
Table 6.9: Models explanation. Note: All models in this table were fine-tuned using (Adafactor, 0.003). When generating paraphrase is using beam 1 and temperature 1.

Model	Avg. Score for 5 generated sentences				Average score
	Relevance	Grammatic	Creativity	Total Avg. Score	
T5_3B_train2_75	3,8	4,6	2,4	10,8	3,6
T5_large_train2_1000	4,4	4,6	1,8	10,8	3,6
T5_3B_mixed_5	4,2	4,4	2	10,6	3,53
T5_large_mixed_50	4	4,6	1,8	10,4	3,47

Table 6.10: Score for the best model from each T5 model size (large, 3B) and train data (train2, mixed train2 - MScoco). The top-2 (in red) is to be compared in with GPT-2 best models in Section 6.2.6.

the other models score lower than average. Template-based approach and GPT-2 fine-tuning receive slightly different scores, where the template-based is performing slightly better than GPT-2.

- Observing the scores, there is a trend that the models perform best in "grammatic" criteria, and almost all model's paraphrases having slightly higher score in "sounding natural" compared to being "relevance" to the original sentences.



Hi, I need your help from you to fill in this questionnaire as part of my thesis, related to paraphrasing.

Instruction:
You willl be given 5 original sentence, each with 5 paraphrases/alternative sentences, where you need to rate each paraphrase/alternative regarding:
- How well it expressed the intended meaning of original sentence (relevance).
- How grammatically correct it is.
- How natural does the statement sound.

The score is a 5-point scale from 1 to 5 (from "very poor" to "very good"), which can be picked from drop down list.

I really appreciate your feedback. Thank you!

		Sentences	To what extent does it express the meaning of the original sentence?	How Gramatically correct?	How natural does the statement sound?
1.	Original	**You don't like action thriller movies if they are espionage.**			
	Paraphrases	You hate to watch movies full of action thriller if they are about espionage.	Pick from list	Pick from list	Pick from list
		You don't like movies about action thriller if they are about secret service.	Pick from list	Pick from list	Pick from list
		You don't prefer movies about gratuitous violence if they are about spy.	Pick from list	Pick from list	Pick from list
		Espionage movies is not for you if they are full of action.	Pick from list	Pick from list	Pick from list
		You don't like action thriller movies if they are spy.	Pick from list	Pick from list	Pick from list

		Sentences	To what extent does it express the meaning of the original sentence?	How Gramatically correct?	How natural does the statement sound?
2.	Original	**You like bollywood movies, especially if they are not fighting the system.**			
	Paraphrases	You enjoy movies about bollywood, especially if they are not about fighting the system.	Pick from list	Pick from list	Pick from list
		You like to watch bollywood movies, especially if they are not about killing the president.	Pick from list	Pick from list	Pick from list
		you love watching movies about bollywood, especially if they are not full of anti-authoritarian propaganda.	Pick from list	Pick from list	Pick from list
		You enjoy watching movies from india, particularly if they are not fighting the system.	Pick from list	Pick from list	Pick from list
		You are into movies from india, particularly if they are not about rebellion.	Pick from list	Pick from list	Pick from list

Figure 6.1: Screenshot of the questionnaire for rating the paraphrase results from 4 best models and template based approach.

Model	**Avg. Score for each criteria**				**Average score**
	Relevance	**Grammatic**	**Natural**	**Total Avg. Score**	
T5-3B	4	4,36	4,12	12,48	**4,16**
T5-large	3,92	4,24	3,84	12	**4**
Template-based	3,4	4,04	3,48	10,92	**3,64**
GPT2-774M	3,12	4	3,44	10,56	**3,52**
GPT2-355M	3,24	3,88	3,32	10,44	**3,48**
Average score for each criteria	**3,5**	**4,1**	**3,64**		

Table 6.11: Score of best models based on feedback from 5 people, sorted from the best average score.

- Comparing the number of parameters in both T5 and GPT-2, it seems the models with larger parameters generate slightly better results than the models with smaller parameters i.e., GPT-2 774M has better score than GPT-2 355M, and T5 3B has better score than T5 large. This is also in line with most of the previous scoring result in Table 6.6 and 6.10. However the difference are not that significant, so for this case, it seems using smaller size of models is sufficient, as it has advantage of needing less resources when doing fine-tuning (GPU Memory, and also size of repository when saving all checkpoints for the fine-tuned models).

- In previous scoring Table 6.6 and 6.10, GPT-2 models have higher score than T5 models, due to the "creativity" part, that GPT-2 introduce more replacements to parts of sentence compared to T5 which is having more subtle changes. This could probably indicates that the "creativity" is not always in line with "sounding"

natural and "relevance," since there is a chance that the replacements might sound more un-natural compared to the original.

The type of train data is one of the essential parameters for fine-tuning based on the experiment in Sections 6.2.4 and 6.2.5. The experiment uses train 1, train 2, and mixed train 2 - MSCOCO, as explained in Section 6.2.1. The use of train 1 data where the paraphrase only changed the sentence pattern without changing the tags resulted in the same type of paraphrases from the fine-tuned models. Most of the generated paraphrases only have variety in sentence patterns without further changes.

Meanwhile, the use of train 2 dataset and mixed train2-MSCOCO dataset, which introduces more variation to models in terms of sentence patterns and tags replacements, the paraphrases from the fine-tuned models can also introduce more replacements in parts of the sentences apart from only changing sentence patterns. Some of the replacements are quite creative and relevant, i.e., "bollywood" to "india," "cannibalism" to "cannibalistic plot," and "scandal" to "corruption." This shows that how we format our training data will impact the results significantly.

However, most of the paraphrase results seem to closely resemble the adjusted template pattern, which is good in terms of capturing the pairwise tag interactions, but to some extent, we would like it to have some more varieties from the adjusted template. We believe by making some adjustments toward the composition of training data; the results could be improved, for example, by lowering the number of train 2 data to "just enough," since 50K of original-paraphrase contains a lot of pattern repetition. It is also suggested to explore further in the mixed composition of train 2 dataset and MSCOCO dataset.

6.3 Summary

Our experiment utilizing neural network models, specifically pre-trained for paraphrasing-purpose, shows poor results, due to the lack of annotated data relevant to the topic within the given task. One of the models was trained using the Quora dataset, which consists of question sentences; therefore, the model's knowledge seems limited to generating question sentences, which is not suitable for our task. The models also could not identify the context of the sentence well, as the training data only have tiny proportion relevant topic.

To collect enough annotated training data through, for example, a crowdsourcing platform or expert service will be time-consuming. This is also not a scalable approach, as the collected data might only useable for this specific task. Therefore we experimented on the transfer learning approach.

We utilized pre-trained general-purpose neural models with fine-tuning to transfer the pre-learned knowledge and introduce the model to the new specific task .i.e., generating paraphrases. We fine-tuned the models with several parameters but focusing more on model size, dataset types, and the number of steps as the main parameters. Both fine-tuned GPT-2 and T5 models show good performance in generating paraphrasing, with "T5-3" and "T5-large" topping the list. However, there are signs of memorizing, which assumed caused by repetitive sentence patterns within the training data. The memorizing behavior was also observed by Radford et al. [8] when there are many repetitions in their dataset.

Chapter 7

Conclusions

7.1 Summary

We explored different frameworks for generating paraphrase for textual descriptions of user preferences (user preference statements) to sound more natural by saying them with alternative expressions while adhering to correct grammar and preserving the intended meaning. The two main approaches experimented on are template-based approach, and neural paraphrase approach by trying several neural network models to be utilized for paraphrasing. This thesis's objective is to identify which approach can be effectively devised to generate paraphrases for the user preference statements that are relevant, grammatically correct, and sound natural.

In the line of template-based approach, the steps in the experiment are: (1) creating the basic template; (2) enhancing it with more sentence patterns in the adjusted template; (3) mining similar expressions for tags through Amazon movie reviews. The last step did not work as we expected; however, we discovered an alternative approach, i.e., by mining similar expressions from other available tags, which gives better outcomes.

In the line of neural paraphrase approach, the experiments are grouped into: (1) utilizing neural models specifically created for paraphrasing, i.e., VAE-LSTM [5] and Stacked Residual LSTM [7]; (2) employing transfer learning method by fine-tuning neural models which have been pre-trained for general purpose, i.e., GPT-2 [8] and T5 [9].

RQ 1: Can an effective template-based approach be devised for generating paraphrases of user preference statements which are relevant, grammatically correct, and sound natural?

Based on results in Table 6.11, template-based approach receives quite decent scores (3,4

for relevance, 4,01 for grammatic, and 3,48 for sounding natural); therefore, we conclude this approach can be effective for generating paraphrases with the above-mentioned criteria, although there is still room for improvement, specifically for making a more robust template-based solution.

RQ 1.1: Does mining different expressions from movie reviews is an effective method to enrich a template-based approach by finding synonyms for user-tags?
Mining different expressions from movie reviews turn out not an effective method for finding synonyms for tags. As seen in Tables 4.12 and 4.13, The extracted words or phrases are more towards "related" words or phrases, but not close to "synonym." However, we discovered and implemented another approach, which is extracting similar expressions from other available tags. As seen in Tables 4.16 and 4.17, with this method, we managed to get more reasonable similar words or phrases.

Referring to results in Table 6.11, both fine-tuned GPT-2 and T5 get scores above 3 for all criteria; therefore, the use of transfer learning in neural models can be effective for generating paraphrase fulfilling those criteria.

RQ 2: Can an effective neural network architecture be devised for generating paraphrases of user preference statements which are relevant, grammatically correct, and sound natural?
In general, the neural paraphrasing approach can be effective for generating paraphrases for this task; however, not all neural network architectures in our experiments can be utilized for our task. It will be explained more in answer to RQ 2.1.

RQ 2.1: Which neural paraphrase generation architecture from prior work is most suitable for this task?
We could not obtained good results from neural models specifically made for paraphrase generation ((VAE-LSTM [5] and Stacked Residual LSTM [7]). The main issue is insufficient labeled training data specifically related to the topic for the task, i.e., user preference summary related to movies. To resolve this, we might need to collect a large number of human-generated paraphrases as training data; however, it will be time and resource demanding. This will also not be scalable because the collected data might not be useful for other tasks.

Utilizing transfer learning through fine-tuning pre-trained models eliminates the need for large training datasets. This seems to be the most suitable approach in the line of neural paraphrase approach. Both models are able to understand their new task, which is generating paraphrase. They also managed to absorb the pairwise tag interactions,

and they show a good level of "creativity" in introducing new words and phrases in parts of sentences. As shown in Table 6.11, T5 fine-tuned model gets the highest score in all category which is relevant, grammatic, and sounding natural. However, further trials and observations should be carried out in future works to understand the behavior of the models better and minimize possible memorizing issues.

RQ 2.2: Can synthetic training data be effectively used for fine-tuning?
Yes, the results show that the models fine-tuned with synthetic training data can understand the paraphrasing task with decent results. The setup of the training data is one of the critical elements to fine-tune the pre-trained models successfully. The style of generated paraphrases gets much influence from how the training data setup is fed to the model. From observation, the more the paraphrases differ from the original sentences in the training data, i.e., having more creative patterns and words/phrase replacements, the more it will induce the fine-tuned models to also becoming more "creative." Therefore, it is crucial to find the most effective way to set up the training data and improve the quality of synthetic training data.

There are still many ways to play around with the training data setup, size, and proportion of mixed data, which we believe can improve the fine-tuning results.

RQ 3: Which of the template-based and neural approaches performs better?
Both template-based approach with synonyms replacement from other tags and neural paraphrase with fine-tuning on pre-trained models are potential approaches that have their strong points and drawbacks, with potential for further improvements.

- The template-based approach results are more predictable with less "surprise" factor as the quality of paraphrase depends on how well-defined the rules and sentence pattern, how good is the tags categorization, and how good is the quality of tags replacements by similar words. However, the drawback is the variety of the paraphrases is limited to some extent, depending on the complexity of the rules. One can also end up with very complicated rules to implement to handle the complex nature of language.
- The neural paraphrase approach using T5 models with fine-tuning shows the best result for all categories as shown in Table 6.11, which indicates its strongest potential for implementation to get a more "sounding natural" paraphrase. However, further experiments need to be carried out to understand the model's behavior better and find the right parameters to improve the effectiveness of transfer learning for this specific task. As mentioned in Section 6.2.2, one of the concerns which could be a

material for further work is occasional inconsistencies in the quality of the generated paraphrases, that the same model can generate different quality of paraphrases from one or different test data.

7.2 Future Directions

As briefly mentioned in the previous section, there is still room for improvement for both lines of approaches. Our suggestion for the focus areas in future works are:

1. In the template-based approach:
   - Increasing the accuracy of POS tagging and chunking to have a better categorization of user-tags, thus, can further reduce the failure cases in the adjusted template.
   - Add more variety of sentence patterns in the adjusted template. Based on the experiment, gathering input from humans can be valuable to enrich sentence patterns. In the experiment in Section 4.5 we received some creative inputs for sentence pattern, which we added to the adjusted template.
   - Currently, the pairwise tag interactions are based on 5 levels of user preferences: $--$, $-$, N, $+$, and $++$. More advanced improvement is to add intensity of user preference level, for example, by adding $---$, or $+++$; however, it means that the pairwise tag interactions need to be re-defined. It will contribute to more diverse pairwise tag interactions.
2. In the line of neural paraphrasing with fine-tuning:
   - Experimenting more on the training data sizes and different mix proportions between datasets in mixed training data. We observe problems with memorizing, which most probably due to repetition in training data; therefore, we can experiment with much smaller data size to lower the level of repetitions and just enough amount of data for the model for understanding the task.
   - With the consideration that fine-tuning does not need a large amount of training data, collecting human-generated paraphrases might be another way of improving the quality of the dataset, for example, through professional human paraphrasing or crowdsourcing platform. However, depending on the amount of data needed, it might still be resource-demanding.
   - Related to the previous item, if we use good quality human-generated paraphrases in the dataset, or if we managed to generate synthetic paraphrases which are

reliable enough to be treated as ground truth, automatic quantitative evaluation method such as METEOR [58], ROUGE [59], or BLEU [57] can be explored. It will potentially improve the efficiency of the evaluation compared to only depending on the human judgment method.

Appendix A

Supplementary information

A.1 Link to Codes Repository

The codes developed during this thesis work can be found at the following GitHub repository:
https://github.com/rennyoctavia/Towards-More-Natural-Explanations-of-User-Preferences

A.2 Listing

Listing 2 is the code showing index mapping as explained in Section 4.4.2.

```
INDEX_MAPPING =
{
    "settings" : {
        "index" : {
            "number_of_shards" : 1,
            "number_of_replicas" : 1
        },
        "analysis": {
            "analyzer": {
                "my_english_analyzer": {
                  "type": "custom",
                  "tokenizer": "standard",
                  "stopwords": "_english_",
                  "filter" : ["shingle-filter","lowercase","stop"]
                    }
                },
             "filter":{
                "shingle-filter":{
                "type":"shingle",
                "min_shingle_size":2,
                "max_shingle_size":4,
                "output_unigrams":True
                }
            }
        }
    },
    "mappings": {
        "properties": {
            "review": {
                "type": "text",
                "term_vector": "with_positions",
                "analyzer": "my_english_analyzer"
            },
             "tags": {
                "type": "keyword"
            },
            "movielens_ids": {
                "type": "keyword"
            },
        }
    }
}
```

Listing 2: Index mapping for Elasticsearch explained in Section 4.4.2.

Bibliography

[1] Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. Transparent, scrutable and explainable user models for personalized recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 265–274, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361729.

[2] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *CoRR*, abs/1804.11192, 2018.

[3] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research Development in Information Retrieval*, SIGIR '14, page 83–92, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450322577.

[4] Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. Paraphrase generation with deep reinforcement learning. *CoRR*, abs/1711.00279, 2017.

[5] Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. A deep generative framework for paraphrase generation. *CoRR*, abs/1709.05074, 2017.

[6] Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, page 834–842, USA, 2009. Association for Computational Linguistics. ISBN 9781932432466.

[7] Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek V. Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. Neural paraphrase generation with stacked residual LSTM networks. *CoRR*, abs/1610.03098, 2016.

[8] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[9] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019.

[10] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook*, volume 1-35, pages 1–35. 10 2010.

[11] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3): 56–58, March 1997. ISSN 0001-0782.

[12] Koen Verstrepen, Kanishka Bhaduriy, Boris Cule, and Bart Goethals. Collaborative filtering for binary, positiveonly data. *SIGKDD Explor. Newsl.*, 19(1):1–21, September 2017. ISSN 1931-0145.

[13] Ben Schafer, Ben J, Dan Frankowski, Dan, Herlocker, Jon, Shilad, and Shilad Sen. Collaborative filtering recommender systems. 01 2007.

[14] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering, 2013.

[15] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003. ISSN 1089-7801.

[16] Xiwang Yang, Yang Guo, Yong Liu, and Harald Steck. A survey of collaborative filtering based social recommender systems. *Comput. Commun.*, 41:1–10, March 2014. ISSN 0140-3664.

[17] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. *Semantics-Aware Content-Based Recommender Systems*, pages 119–159. 01 2015. ISBN 978-1-4899-7636-9. doi: 10.1007/978-1-4899-7637-6_4.

[18] Erion Çano and Maurizio Morisio. Hybrid recommender systems: A systematic literature review, 2019.

[19] Thomas Hornung, Cai-Nicolas Ziegler, Simon Franz, Martin Przyjaciel-Zablocki, Alexander Schätzle, and Georg Lausen. Evaluating hybrid music recommender systems. In *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 01*, WI-IAT '13, page 57–64, USA, 2013. IEEE Computer Society. ISBN 9780769551456.

[20] Lina Yao, Quan Sheng, Aviv Segev, and Jian Yu. Recommending web services via combining collaborative filtering with content-based features. pages 42–49, 06 2013. ISBN 978-0-7695-5025-1.

[21] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *TiiS*, 5:19:1–19:19, 2015.

[22] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, January 2004. ISSN 1046-8188.

[23] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, RecSys '07, page 17–24, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937308.

[24] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. How should i explain? a comparison of different explanation types for recommender systems. *Int. J. Hum.-Comput. Stud.*, 72(4):367–382, April 2014. ISSN 1071-5819.

[25] Nava Tintarev and J.F.M. Masthoff. Evaluating the effectiveness of explanations for recommender systems. *User Modeling and User-Adapted Interaction*, 22:399–439, 2011.

[26] Fatih Gedikli, Mouzhi Ge, and Dietmar Jannach. Understanding recommendations by reading the clouds. In Christian Huemer and Thomas Setzer, editors, *E-Commerce and Web Technologies*, pages 196–208, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-23014-1.

[27] Xu Chen, Yongfeng Zhang, Hongteng Xu, Yixin Cao, Zheng Qin, and Hongyuan Zha. Visually explainable recommendation, 2018.

[28] Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2. 02 2008.

[29] Karen Sparck Jones. *Natural Language Processing: A Historical Review*, pages 3–16. Springer Netherlands, Dordrecht, 1994. ISBN 978-0-585-35958-8.

[30] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. 01 2009. ISBN 978-0-596-51649-9.

[31] Ann Taylor, Mitchell Marcus, and Beatrice Santorini. The penn treebank: An overview. 01 2003.

[32] Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, ConLL '00, page 127–132, USA, 2000. Association for Computational Linguistics.

[33] Kathleen R. McKeown. Paraphrasing questions using given and new information. *Comput. Linguist.*, 9(1):1–10, January 1983. ISSN 0891-2017.

[34] Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. UNT: SubFinder: Combining knowledge sources for automatic lexical substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 410–413, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[36] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[37] Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. Neural generative question answering. *CoRR*, abs/1512.01337, 2015.

[38] Xiangang Li and Xihong Wu. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. *CoRR*, abs/1410.4281, 2014.

[39] Florin Brad and Traian Rebedea. Neural paraphrase generation using transfer learning. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 257–261, Santiago de Compostela, Spain, September 2017. Association for Computational Linguistics.

[40] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[41] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[43] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. *CoRR*, abs/1511.06349, 2015.

[44] Alec Radford and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[45] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey, 2020.

[46] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[47] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019.

[48] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences, 2018.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[50] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[51] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[52] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification, 2018.

[53] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models, 2017.

[54] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[55] Noel Cressie and Timothy Read. Pearson's x 2 and the loglikelihood ratio statistic g 2 : A comparative review. *International Statistical Review*, 57, 04 1989.

[56] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[57] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics.

[58] Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, page 228–231, USA, 2007. Association for Computational Linguistics.

[59] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[60] Sam Witteveen and Martin Andrews. Paraphrasing with large language models, 2019.