Semi auto-taggers for music

Combining audio content and human annotations for tag prediction

by

A.J.C. Lugtenburg

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday December 13, 2022 at 09:30 AM.

Student number:	4370805	
Project duration:	Jan 1, 2019 – Dec 13, 2022	
Thesis committee:	Dr. C. C. S. Liem MMus,	TU Delft, supervisor
	Dr. J. H. Kim,	TU Delft, daily supervisor
	Dr. J. C. van Gemert	TU Delft

This thesis is confidential and cannot be made public until December 13, 2022.

TU Delft

Abstract

Auto-tagging systems can enrich music audio by providing contextual information in the form of tag predictions. Such context is valuable to solve problems within the MIR field. The majority of recent auto-tagging research, however, only considers a fraction of tags from the full set of available annotations in the original datasets. Because of this restriction, potential relationships between tags remain unconsidered and tagging may be less rich. These relationships suggest alternative ways to establish an auto-tagging system. For instance, a few accurate annotations from experts can improve the richness and quality of the auto-tagging system by providing explicit context in addition to audio content features. In this work, we propose an adaptation to the auto-tagging task, semi auto-tagging, to demonstrate such potential. In our framework, tags are allowed as contextual input to the tag prediction system in addition to audio content information. The system then suggests additional relevant tags. We implement two models that fit within the framework: content-aware matrix factorization and graph convolutional networks. To see whether we can improve upon a traditional auto-tagger, we compare these models with a multilayer perceptron as a baseline. Experimental results show that semi auto-tagger models can predict relevant tags both in the absence and presence of an audio content feature, and can predict tags for previously unseen songs similarly to an audio content auto-tagger. Based on a tag embedding comparison, we find that semi auto-tagger models can better learn implicit relationships between tags with a similar text string representation when compared to the baseline.

Preface

Before you lies my master thesis, written to fulfill the requirements of the degree of Master of Science (MSc) in Computer Science. For the duration of this project, I have been given many opportunities to develop myself and learn and I am incredibly grateful for that.

I would like to dedicate this section to thanking a number of people as without them I would not have gotten this far. First, I would like to express my gratitude to Cynthia Liem and Jaehun Kim (my daily supervisor). During the many meetings I have had with Jaehun I've learned many new things and I would not be the engineer I am today without this knowledge. I would also like to thank Jaehun for the many fun discussions we had about technology and wish him all the best for his future career abroad. Second, I would like to thank my parents, family, and friends. Without their support I would not have been able to become who I am today.

It has not been an easy road for me to conclude this work, but I am proud of the result. To whoever is reading this, I hope you enjoy reading this work.

Jochem Lugtenburg
Delft, December 2022

Contents

1 Introduction 1
1.1 Research Questions . . . 4
1.2 Scientific Contribution . . . 5
1.3 Thesis Structure . . . 5

2 Background and Related Work 7
2.1 Music Tags . . . 7
2.1.1 What defines a vocabulary of tags? . . . 7
2.1.2 Obtaining tags . . . 8
2.1.3 Issues with tag vocabularies . . . 9
2.2 Music Auto-Tagging . . . 11
2.2.1 Defining Music Auto-Tagging . . . 11
2.2.2 Audio input representations . . . 11
2.2.3 Traditional Auto-Tagging systems . . . 11
2.2.4 Deep Learning Based Approaches . . . 13
2.3 Models. . . . 16
2.3.1 Deep Neural Networks . . . 16
2.3.2 Matrix Factorization . . . 17
2.3.3 Graph Representation Learning . . . 18

3 Methodology 21
3.1 Limitations of auto-tagging . . . 21
3.1.1 Formalization: Semi Auto-Tagging . . . 22
3.2 Models. . . . 24
3.2.1 Content-Aware Matrix Factorization . . . 24
3.2.2 Graph Convolutional Network . . . 25

4 Experimental Setup 27
4.1 Datasets . . . 27
4.1.1 Features . . . 27
4.1.2 Million Song Dataset (Last.fm) . . . 28
4.1.3 MagnaTagATune . . . 29
4.2 Experimental Design . . . 32
4.2.1 Model and Hyperparameter selection . . . 32
4.2.2 Loss function . . . 32
4.2.3 Evaluation. . . . 33
4.3 Implementation Details. . . . 34
4.3.1 Hardware . . . 34
4.4 Research Questions . . . 35
4.4.1 RQ1: Effectiveness of a Semi Auto-tagging system . . . 35
4.4.2 RQ2: Effective use of the vocabulary . . . 36

5 Results 37
5.1 Classification Accuracy . . . 37
5.1.1 AUC-micro . . . 37
5.1.2 AUC-macro . . . 37
5.1.3 AUC-samples . . . 39
5.2 Ranking Accuracy . . . 40
5.2.1 NDCG (full) . . . 40
5.2.2 NDCG (50% tail). . . . 40

5.3 Embedding Similarity . . . 42
5.4 Discussion and limitations . . . 44
5.4.1 Discussion . . . 44
5.4.2 Limitations . . . 44

6 Conclusion 47
6.1 Contribution . . . 47
6.2 Future Work . . . 49

A Experimental Design 51

B Classification Accuracy Results 59

C Ranking Accuracy Results 65

D Embedding Similarity Results 73

E Song predictions 75
E.1 MSD 128 5 seeds . . . 76
E.2 MTAT 128 5 seeds . . . 79

F Supporting figures 81

Bibliography 89

1 Introduction

The field of Music Information Retrieval (MIR) obtains and uses information about music to solve a range of problems related to music. These problems can be solved by making use of information extracted from the audio content, such as audio waveforms, or by making use of contextual information that helps in the understanding of music. Audio content can only provide a limited amount of information, which means that common MIR tasks such as recommendation cannot always be solved using audio-only. Contextual information on the other hand provides information that audio content does not provide, for example in the form of metadata. This information can then be used to group songs, artists, or albums, for example using their genre. Context often comes in the form of a set of tags that compose a description of the music [53]. Music tags are descriptive words or phrases that can provide (high-level) information for entities related to music such as songs, albums, or artists [53]. Figure 1.1 shows an example of songs that are labeled using a range of different tags. Multiple tags can be assigned to a single song and tags can be of different categories such as genre (*pop*, *classic rock* or *pop-rock*), instruments (*guitar*), opinion (*Favourite Songs, Favorite*) and many others. Such a rich description of music is valuable as contextual input to MIR systems. For example, tags can be used to reduce the impact of the *cold-start* problem in music recommendation [29], where recommendations cannot be made for new songs that have not been listened to by users and therefore have not been seen by the system before [51]. The context provided by tags can alleviate this problem by grouping similar songs together, which allows for recommendations to be made for those songs. Additionally, a similar cold-start problem can be solved when recommending songs to users. If new users do not have a listening history, tags of interest supplied by the user, or tags obtained from the first songs consumed by a user can be used as input to a recommender system. Tags can also be used to learn representations directly from the audio to learn a model [18, 26, 35, 37, 45]. Such representations can then be reused as input to other tasks, such as finding similar songs. Another use case of tags can be found in search engines where users can query for music by looking for specific concepts or keywords [53]

Two possible sources for obtaining tags are humans and machines. Humans can tag in several ways [74]: trough social tagging (crowd-sourced) [53], expert annotation [74] or tagging games [54, 55]. Social-tagging is a crowd-sourced approach [53] where large groups of people tag a collection of songs to obtain a so-called folksonomy [2] of social tags. An example of this can be found at *Last.fm* [1] where users have provided tags for songs they listened to. Last.fm is a community of music listeners where users can track their listening behavior and taste to obtain new music recommendations. The benefit of using a service with a large user base is that a varied set of tags can be obtained in very large quantities. A downside of this is, however, that tags can be "noisy" in many ways, as users are unrestricted in the tags they apply. Examples of such noise are that tags can be ambiguous, contain (spelling) errors, or do not have any relationship with the song. Also, social tags applied by humans can be sparse since unpopular or new songs often are untagged, which makes it difficult to make recommendations for these songs using their annotations [28].

Expert annotation is performed by knowledgeable taggers that tag from a relatively small, but accurate vocabulary [74]. An example of expert taggers can be found at music streaming services

[1] `https://www.last.fm/about`, Last.fm about page

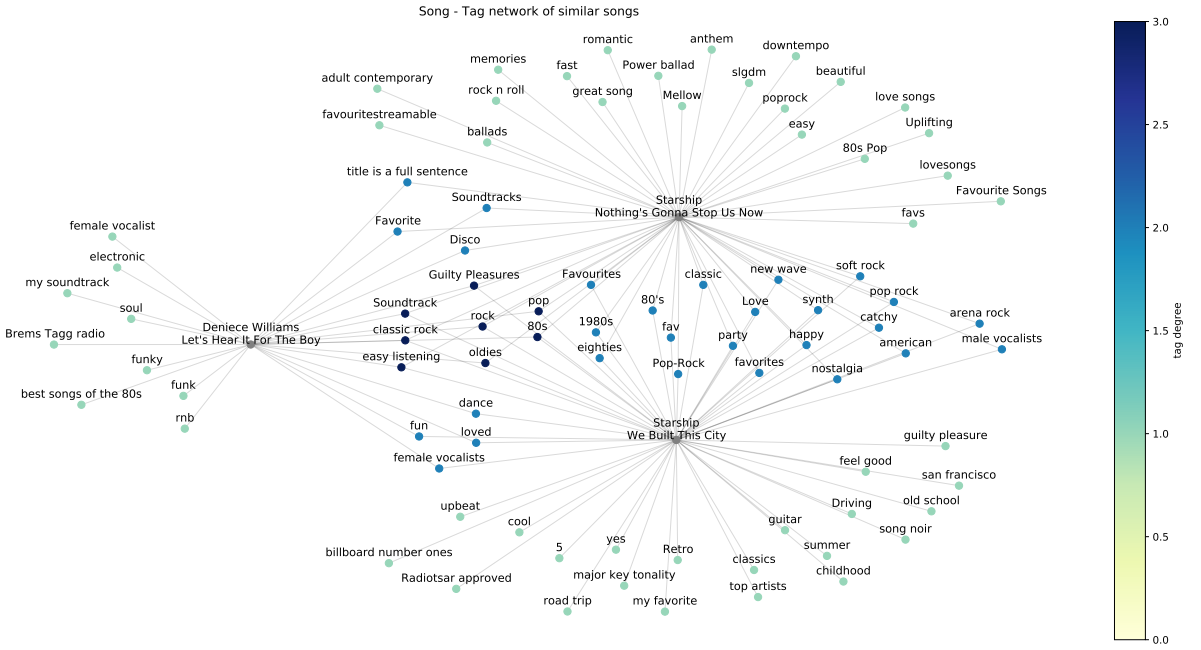


Figure 1.1: Song-tag relationships for 3 similar songs from the top-1000 most frequent tags in the Last.fm subset of the Million Song Dataset. Grey nodes represent songs, listing the artist and title. Colored nodes represent tags. Tag degree shows the number of songs that are connected to each tag node.

such as Pandora. Pandora's[2] Music Genome Project aims to gather music knowledge using trained musicologists [1]. Since these experts are trained specifically for the tagging task this approach suffers less from noise, however, the vocabulary is restricted which leads to less rich tagging. Additionally, experts are an expensive resource both in terms of money and scalability since tags can only be applied at a limited rate which makes it difficult to tag at scale. Tagging games such as TagATune [54, 55] balance crowd-sourcing with a fixed vocabulary where tag collection is gamified to encourage players to tag more accurately.

Machine tagging, on the other hand, is typically referred to as auto-tagging [10]. The goal of an auto-tagger is to predict tags directly from song audio, without human interference to resolve the issue of tag sparsity. A benefit of this approach is that it is scalable because songs can be tagged at a much higher rate compared to human annotators, and computing resources are generally cheap. However, achieving high accuracy is complicated with these models because the quality of tagging datasets used to train them varies. Datasets vary in vocabulary size, richness, and noise, and often follow a long-tail distribution. Primarily, two datasets are used. The first [5, 11], Million Song Dataset (MSD) is a folksonomy collected from *Last.fm*. The second, MagnaTagATune (MTAT) [6, 55] is collected using a more controlled environment in the form of a game. From these datasets often only a subset of the most frequent tags in the available dataset is used. A limitation in the size of the used subset causes the vast majority of the available song-tag mapping to remain unconsidered, thus limiting the richness an auto-tagger can achieve. In the case of MSD often only 50 out of 500K tags are considered. Some reasons for this are that smaller subsets are easier to analyze [28], some tags being sparse [29], dataset imbalance [16] or size of the output space [19]. Recent work [16, 25] shows that high classification accuracy can be achieved on these subsets, but using these subsets may hinder real-world use cases for these models as many less frequent but useful tags are ignored. Additionally, other fields have already been able to achieve annotation using very large vocabularies, for example, in the field of computer vision. In image classification, [52] models can successfully classify images from 1000 unique labels with a relatively low rate of error.

So far we have discussed humans and machines and their (dis)advantages. Humans can provide larger vocabularies, and more accurate tagging (experts). Auto-tagging algorithms can provide scalability.

[2] https://www.pandora.com/about, Pandora about page

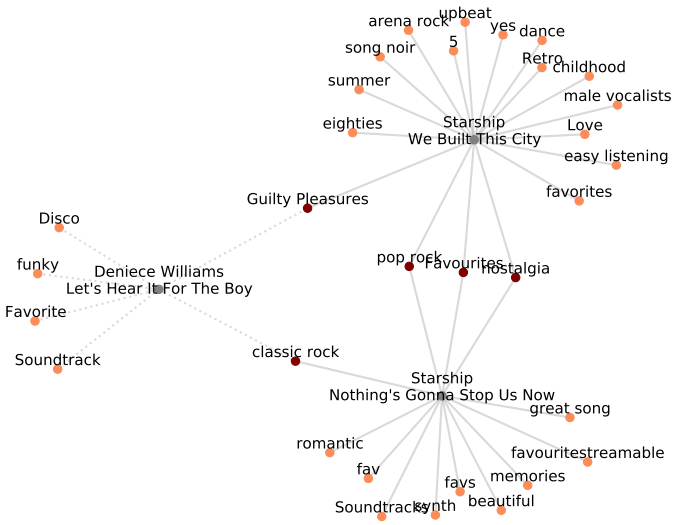


Figure 1.2: Graph of songs and tags based on three songs from the Million Song Dataset. Node color indicates tag degree. Dashed edges indicate a previously unseen song for which context has been provided by a human expert, unlocking access to more context already available in the dataset.

Current solutions to the auto-tagging problem are often only audio content-based. In addition to this content, it may however be beneficial to also consider context as an important factor when predicting tags, as is currently present in human tagging. Humans are likely to select semantically similar tags. Popular datasets such as the *Last.fm* dataset contained within MSD can be interpreted as a network of songs and tags in which tags that have a similar semantic meaning are closer within the graph. The bigger or richer the vocabulary, the more of these relationships become available. For example, a song that is tagged *rock* is likely to include *guitar*, the instrument around which the genre is centered. There are ways in which we can make use of this context within auto-tagging systems. For example, we can benefit from (hidden) implicit relationships that exist between tags within an auto-tagging dataset by explicitly learning from them. Likewise, for new songs, we can make use of external tag annotations as input to the model. Similar to how audio features currently provide content information, external (expert or system user) seed information can provide context for those unseens songs that are not yet connected to the song-tag graph contained in the dataset by default. By doing so we can suggest several relevant tags based on expert input. Additionally, an expert-machine hybrid system can be an alternative form of tagging that may be both more efficient and effective for the human tagger. Figure 1.2 shows how this could look on a subset of the data shown in Figure 1.1. A previously unseen song from *Deniece Williams* is connected to the existing dataset via external (tagger) seed input. After it is connected to the graph, a tagging system can benefit from the context of the known songs in the network to suggest additional tags. Similar completion problems exist in MIR, for example the 2018 ACM RecSys challenge [15, 86] in which participants were asked to solve automatic music playlist continuation. Models were given incomplete playlists and were asked to recommend songs from a big vocabulary of 2 million songs based on the songs that were already given. It was shown that systems were able to come up with more relevant playlist continuations as the number of given songs increased.

In this work, we propose an adaptation of the auto-tagging problem which we name *semi auto-tagging*. In our framework tags are allowed as contextual input to the model in addition to an audio content feature. By combining these two inputs, we can suggest relevant tags based on both the audible (content) properties of the song and tags (context) that have been provided by an external tagger. We hypothesize that there is merit in such an approach and that we can achieve comparable or better performance compared to a content-based auto-tagging baseline. Our hypothesis is validated by implementing two model candidates that fit in our framework: a deep-learning-based graph approach [69] and a matrix factorization [58] model. We compare their tagging performance with a simple content-based auto-tagging baseline by looking at ranking accuracy and classification accuracy. We investigate the effect of a content-driven feature by comparison with a random feature to test whether semi auto-

tagging models can make use of the content feature in addition to contextual input. Additionally, we compare learned embeddings to see if models can learn the similarity between tags effectively. Since vocabulary size is important for the amount of context that is provided, we evaluate the models using varying sizes of vocabulary, from multiple datasets.

1.1. Research Questions

We hypothesize that exploiting the context contained within music tagging datasets combined with audio content in the form of a semi auto-tagger is beneficial for the music auto-tagging problem as a whole. Therefore we formulate two main research questions, driven by our interests in extending the auto-tagging problem.

Research Question 1

RQ1 - How can we build a system that suggests additional tags using contextual input?

We aim to answer this question by looking at three important aspects mentioned in the previous section while taking into account different vocabulary sizes for each subquestion.

SQ1.1 - Given contextual input combined with audio features, can a semi auto-tagger predict more relevant tags compared to a content-only auto-tagger?

Our first sub-question follows from current auto-tagging systems which are heavily based on audio content input and have been successfully able to predict tags in a traditional auto-tagging setting. Therefore a model that is capable of combining content with context should be able to use audio in the form of features extracted from the audio as well. We hypothesize that if we combine contextual input with meaningful audio features we can obtain more relevant tag predictions compared to a baseline model that can only predict based on audio content.

SQ1.2 - For previously unseen songs, can a semi auto-tagging model predict relevant tags?

Second, the system must be able to work for unseen songs. Since human taggers often have to deal with songs that are not yet tagged where only input audio is available and the model can not yet rely on context. By answering this question we verify that a semi auto-tagger has an important property of a traditional auto-tagger: it alleviates the cold-start problem by being able to predict tags for unseen songs directly from the audio content.

Research Question 2

RQ2 - Does a noisy tag vocabulary contain useful information for (auto-)tagging?

This question is motivated by current approaches which are often using only a subset of available auto-tagging datasets, while there is significant unexploited potential. We aim to answer this question by showing two properties. First, we consider a bigger vocabulary than most auto-tagging to show that (semi) auto-taggers can make use of these larger vocabularies. Second, we look at the contextual song-tag relationships learned by our system. Given some strong implicit relationships that are present in the datasets such as between *rock* and *electric guitar* it is expected that a model that considers explicit relationships as input can better grasp implicit relationships compared to a model that does not consider explicit relationships as input. Therefore we look at the tag representations learned by our models to see whether they can be used to find similar concepts within the vocabulary.

1.2. Scientific Contribution

In this work, we provide two contributions to the field. First, we look at the auto-tagging problem from a new angle. By considering human input, we propose a hybrid between human tagging and machine tagging in the form of semi auto-tagging. This opens up new opportunities for researchers to design systems that can better help expert taggers. We provide examples of systems that adhere to the definition of such a system. Second, we give insight into how these systems can be evaluated and provide evidence that they indeed contribute to solving the semi auto-tagging problem.

1.3. Thesis Structure

The thesis is structured as follows. First, we provide the necessary background to understand the auto-tagging problem and the current direction of research in Chapter 2. In this chapter, we also lay out the necessary theory for the models that we use in our experiments. Second, Chapter 3 contains our approach and contains an overview of our model selection. Then, Chapter 4 outlines our experimental setup, implementation details, and evaluation procedure. Finally, we show and discuss the results and limitations of our work in Chapter 5 and conclude with our findings and future work in Chapter 6.

2

Background and Related Work

In this chapter, we discuss the necessary background which forms the basis for our methodology. First, we discuss what music tags are and the different forms in which they exist. Second, we discuss the auto-tagging problem and various solutions that have been proposed over the past twenty years. Finally, we discuss three types of models that form the basis of the models we use in our experiments.

2.1. Music Tags

An important task in the field of Music Information Retrieval (MIR) is obtaining and using information about music to solve a range of music-related problems. Expanding the available contextual knowledge about music is one way to better solve these problems [53]. By making use of this knowledge, systems can be built that have a better understanding of music audio. As problems grow more complex, more and more contextual information about music is required. One way to obtain this context is by labeling music using music tags. Applications that make use of tags can be found in search and discovery, directed search, tag similarity and clustering, item similarity [53] and (supervised) embedding learning [18, 26, 35, 37, 45]. Importantly, they can be used as an alternative to collaborative-filtering based recommendation approaches that suffer from the *cold-start problem* where songs that have not been encountered before cannot be recommended easily [9, 29].

2.1.1. What defines a vocabulary of tags?

A music tag is a textual description that can be used to annotate entities related to music resources such as artists, albums, or songs [53, 74]. A tag can be a word (*rock*, *favourite*, *piano*) or a phrase (*chamber music*, *alternative rock*, *I like this song*) describing the resource. Differing from a genre commonly found in music libraries such as iTunes, multiple tags can be assigned to a single entity and the number of tags assigned to a resource is unlimited. This freedom allows for rich descriptions to be composed of tags which makes tags a highly valued resource in the field. A collection of tags is often referred to as a vocabulary of tags. One way to obtain a vocabulary is via so-called *social-tagging* [53]. In his work on social tagging, Lamere distinguishes between *keywords* and *social tags*. A keyword is a single label assigned to a resource from a fixed vocabulary. A social tag is typically provided by non-expert users of a social platform. On platforms such as Last.fm, users are free to assign tags to music resources without restrictions. A collection of social tags is often referred to as a folksonomy: a vocabulary composed of tags provided by a large group of taggers in a social environment [2]. A (social) tag can be from a wide range of categories and is not necessarily related to audio content, which is a result of the freedom given to users. Examples of tag categories are *genre, locale, mood, opinion, instrumentation, style, personal, organisational* and many more. Table 2.1 shows examples of categories and tags for frequently applied tags at Last.fm as categorized by Lamere [53]. The majority of these social tags are related to audio. A major benefit of social tags is that they can be obtained at scale due to the way they are collected. Turnbull et al. further describe tag vocabularies, differentiating between *fixed* or *extensible* along with *structured* or *unstructured* tag vocabularies [74]. A vocabulary is fixed and structured if the set of tags is predetermined and grouped into consistent semantic categories. On the other hand, a vocabulary is extensible and unstructured if tags can be added freely and there is no restriction on the contents of the

Tag Category	Frequency	Examples
Genre	68%	heavy metal, punk
Locale	12%	French, Seattle, NYC
Mood	5%	chill, party
Opinion	4%	love, favorite
Instrumentation	4%	piano, female vocal
Style	3%	political, humor
Misc	2%	Coldplay, composers
Personal	1%	seen live, I own it
Organisational	1%	check out

Table 2.1: Example tag categories, frequency and tags as given in Table 1 of [53]. Frequency is based on the 500 most frequent tags collected from Last.fm at time of writing. Note that the majority of tags describe audible contents. Table taken from [53]

tag. Social tags can be seen as unstructured and extensible and provide one of the largest vocabularies available [5].

2.1.2. Obtaining tags

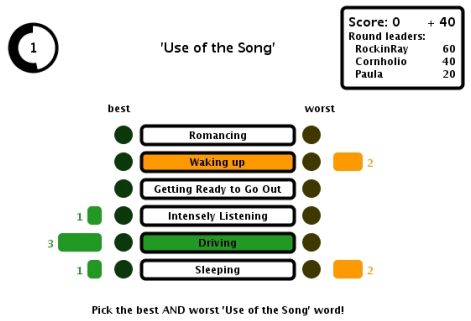
Three human-driven ways to obtain a vocabulary of tags are expert surveys, social tagging, and tagging games. Each way has its strengths and shortcomings which impact their usefulness as a source of tags [74].

Music annotation surveys use experts that annotate music by hand. An example of this is the Music Genome Project (MGP) of Pandora. Pandora is a music and podcast delivery platform and therefore has an interest in understanding music for purposes such as recommendation. The MGP [1] has been gathering music knowledge for over a decade and does so using a team of experts in the form of trained musicologists. A strength of this approach is that it provides high-quality annotations, but on the other hand, it is also a time-consuming process [74]. According to Pandora, it takes about 20 minutes to annotate a single song and between 2000 and 2009 about 700,000 songs have been annotated for 80,000 artists [3, 4]. Their experts discuss audio content and tag from a fixed, structured vocabulary which is beneficial to the quality of the tags obtained. However, given the number of resources allocated by a company such as Pandora, they are unlikely to share this data with the MIR community [74].

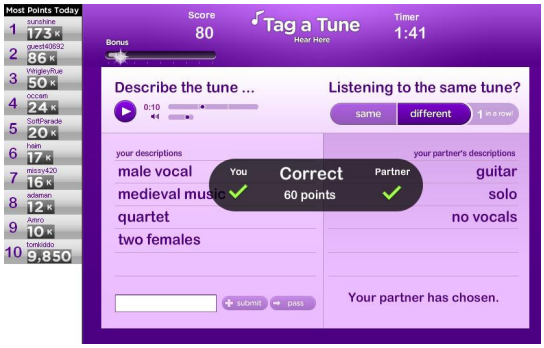
Social (non-expert) tags are tags that are contributed by users of music services such as Last.fm [53, 74]. The unlimited vocabulary users provide can be obtained at scale, depending on the number of users that make use of a social platform. As a result, the datasets obtained from social tagging may be weakly labeled [74], meaning that the absence of a tag does not necessarily mean that the tag does not apply to the resource. Additionally, it may come with noise and the number of tags assigned to a resource may suffer from popularity bias because more popular items are annotated more often [74]. Lamere claims that for their dataset obtained from Last.fm a big majority of artists were not assigned tags [53]. This is even worse when songs are considered. Because of this, social tags may not be suitable for the exploration and recommendation of new or unpopular items. As shown in Table 2.1, a vocabulary obtained from Last.fm the majority of tags however seems to describe the audio [53].

Tagging games such as TagATune [54, 55] attempt to obtain tags for music in a more controlled way, while still making use of a larger group of non-expert taggers. In TagATune, two players are given a fragment of a song and are then asked to describe it. Based on their descriptions they have to decide whether they are listening to the same song or not. This way of tag collection encourages players to describe resources accurately. Figure 2.1 shows example tagging interfaces. There are some risks when collecting data using tagging games. Players can game the system and it may be difficult to create a successful gaming experience [74]. Other examples of tagging games are MajorMiner [59] and ListenGame [73].

Besides human annotation, there are also more machine-driven ways of obtaining tags, such as auto-tagging and web scraping [74]. Auto-tagging is proposed as a solution to a problem similar to the *cold-start problem* in collaborative filtering-based recommendation [74]. Social-tagged songs often come without tags or are sparsely tagged which makes tags unsuitable for the recommendation of new items [53]. Auto-tagging models attempt to predict tags based on the audio content of the song to



(a) User interface of ListenGame. Players are asked to listen to music and select best and worst tags that describe a song. Figure taken from [73]



(b) User interface of TagATune. Two players are asked to describe a song and are then asked if they listen to the same song. They only score points if they correctly guess if they are listening to the same song. Figure taken from [54]

Figure 2.1: Examples of two tagging games.

Dataset	Reference	# Tags	# Songs	Source
AudioSet	[31]	632[1]	2084320[3]	Human Annotators
Ballroom	[33]	8	698[2]	Web Scraping
CAL500	[75]	174	502	Human Annotators
CAL10K	[72]	628	10870	Human Annotators (Pandora)
Free Music Archive (full)	[23]	161	106574	Users (FMA)
Homburg	[43]	8	1886[3]	Users (Garageband.com)[4]
GTZAN Genre Collection	[77]	10	1000	Author
MagnaTagATune	[6, 55]	188	25863[2]	Tagging game
Million Song Dataset (Last.fm)	[5, 11]	522366	505216[5]	Users (Last.fm)
MTG-Jamendo	[13]	195	55609	Users (Jamendo)
Additional datasets[6]	-	-	-	-

Table 2.2: List of commonly used datasets in MIR literature. Note that many come with only a low number of tags, especially compared to MSD.

alleviate this problem. It has been shown that there is merit in improving auto-tagging models [70], as qualitatively higher tagging models are beneficial to the performance of content-based music similarity systems. We will dive deeper into auto-tagging models in Section 2.2.

There exists many different datasets that are obtained via the aforementioned approaches [11, 13, 23, 31, 33, 43, 55, 72, 75, 77]. An overview of the available tag vocabularies that are commonly used in MIR literature and their source is given in Table 2.2.

2.1.3. Issues with tag vocabularies

Several issues arise when dealing with tag vocabularies. The ideal vocabulary is large and contains a diverse set of accurate tags. As discussed in the previous section, collecting tagging datasets is a labor-intensive process, hence it is difficult to obtain such a vocabulary. Popular, relatively clean datasets such as MagnaTagATune [6] come with a small vocabulary and are less diverse than more noisy datasets such as the Last.fm subset of Million Song Dataset (MSD) [5]. Despite being noisy, the vocabulary provided by MSD does however provide the diversity we seek as the dataset contains tags covering a wide range of musical topics such as genre, instrumentation, mood, and more. Being a dataset composed of social tags, it comes with a wide range of issues as described in [53]. Since the users of Last.fm are

[1] Majority is not music related.
[2] 30 second clips or song excerpts.
[3] 10 second song excerpts
[4] Former online community, domain now owned by Apple inc.
[5] Songs that have at least one tag.
[6] Additional MIR datasets can be found at ISMIR: https://ismir.net/resources/datasets/

unrestricted in their tagging behavior. Two tags can have the same meaning such as *r&b* and *r and b*. Tags can be ambiguous such as *Love*. Does the tagger love the song, or is the song romantic? We can also find tags that are likely to be noisy such as single-character tags like *a*. Some taggers are malicious in their tagging intents. For example, people can apply false tags to artists they don't like, tagging a pop song as *Brutal death metal*. These issues can make it difficult to work with these vocabularies often leading to only using a small, more clean subset of these vocabularies [16, 25, 28, 29]. Recent work by Choi et al. however suggests that using bigger subsets of a dataset such as MSD can still be reliable [19].

2.2. Music Auto-Tagging

The music classification task has been part of MIR for more than 20 years. Initial research focused primarily on automatically classifying music genres. Over time research moved from genre classification to music auto-tagging: no longer restricted to genres alone. In this section we will first define the auto-tagging problem, then briefly discuss some common audio input representations. We will then discuss a range of auto-tagging algorithms available.

2.2.1. Defining Music Auto-Tagging

The auto-tagging problem is not clearly defined, most authors have a slightly different view of the problem. Bertin-Mahieux et al. decompose auto-tagging systems into four parts: audio features, tags, a machine learning algorithm, and an evaluation procedure [10]. They define a tag as "a user-generated keyword associated with song resource". Law et al. take a more mathematical approach to the definition, where the music annotation problem is defined starting from the training data, where the goal is to learn a mapping function that maps an audio feature to a set of tag relevance scores [56]. Similarly, Turnbull et al. define the annotation problem as finding a set of semantically meaningful words that describe a query audio track [75]. In this work, we define the auto-tagging problem based on these approaches as follows:

> Given a set of N training songs $S = \{s_1, s_2, \cdots, s_N\}$, annotated using tags from a vocabulary $T = \{t_1, t_2, \cdots, t_V\}$ of length V. Each song is annotated from the set of tags by humans using one of the ways described in section 2.1.2 and is represented by two vectors $s_i = (\mathbf{a_i}, \mathbf{x_i})$. $\mathbf{a_i} \in \{0, 1\}^V$ is the list of ground truth tag annotations, where a tag is marked 1 if it is assigned to song s_i and 0 otherwise. Additionally each song comes with a feature vector of size M, $X = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}\}$ where $\mathbf{x_i} \in \mathbb{R}^M$. The feature is either computed or learned from the audio itself. Finally, the goal of an auto-tagging task is to learn a mapping $\hat{f} = X \times T \to \mathbb{R}$ where each feature is mapped to a score per tag indicating its relevance to the audio, a multi-label classification problem.

2.2.2. Audio input representations

Traditionally features in MIR that are used for auto-tagging depend on domain-specific knowledge rooted in music theory and signal processing [45]. An audio file is processed using an algorithm and converted into a set of features. One of the most common representations is the Mel-Frequency Cepstral Coefficients (MFCC) [21, 46], originally developed for speech recognition. They have been shown to work on other audio inputs such as music as well [29, 75, 76].

MFCCs can be further aggregated to summarize a song or song excerpt. An example of this is the MFCC-Delta feature. First, a set of MFCCs is computed by sliding a short-time window over the song audio. Then, the obtained vectors are concatenated. This set of features can then be extended by the first and second derivatives of the MFCC to capture temporal dependencies between features [75]

More recent auto-tagging works attempt to reduce the amount of manual signal-processing-based feature engineering by learning features directly on the mel-spectrogram or Short-Time Fourier Transform (STFT)[16]. Humphrey et al. argue that the set of hand-crafted features traditionally used by MIR is limited and that the field can benefit from using deep learning-based architectures and automatic feature learning to learn robust audio representations [45]. These learned features can then be used as input to further improve tasks such as auto-tagging. Often, a task such as the auto-tagging task is used to learn these features in a supervised fashion. We dive deeper into these works in **??**.

2.2.3. Traditional Auto-Tagging systems

There are many variations of auto-tagging systems that apply a wide range of algorithms. Early attempts at automatic classification of music mostly focused on genre recognition. Tzanetakis et al. explored genre classification using a collection of features based on music characteristics such as texture, timbre, instrumentation, and rhythm [76]. Additionally, they make use of MFCCs. A gaussian classifier is then learned, which represents each class with a Gaussian distribution estimated from the training data. This classifier shows that genre classification is possible using the selected set of features [76]. Building upon this, a Gaussian Mixture Model (GMM) and a K-Nearest Neighbour (KNN) based approach was tried on a similar set of features [77]. These approaches showed performance similar to

human genre classification but were limited by having only 10 genres available in the dataset[7].

One of the first models to extend beyond genres is a model by Turnbull et al [75]. Their model can both annotate and retrieve songs or sound effects from a database of unlabeled audio. A GMM is learned for each tag using MFCC-Delta features obtained by concatenating the MFCC and its first and second derivatives. Additionally, they contribute the Computer Audition Lab 500 (CAL500) dataset which consists of 500 songs for 500 distinct artists over 50 years. The vocabulary contains 174 tags, and each song contains at least 3 annotations. The songs were annotated by students which were paid to annotate in a similar setting as expert tagging. The authors acknowledge that their data quality may not be optimal, but is still useful for content-based music annotation and retrieval research. Results show that the model is capable of tagging from the entire vocabulary and that it is beneficial to use a relatively clean dataset such as CAL500.

Eck et al. [29] propose a model which learns a model that predicts social tags using AdaBoost [30], also making use of MFCC features. They make use of a subset of the 60 most popular tags obtained by crawling data from the Last.fm Audioscrobbler. Each label is treated as a separate classification problem, where the goal is to predict if an artist has *none*, *some*, or *a lot* of a particular tag. Figure 2.2 shows the architecture of the model. Their approach gives further evidence that auto-tagging music is possible and has merit.

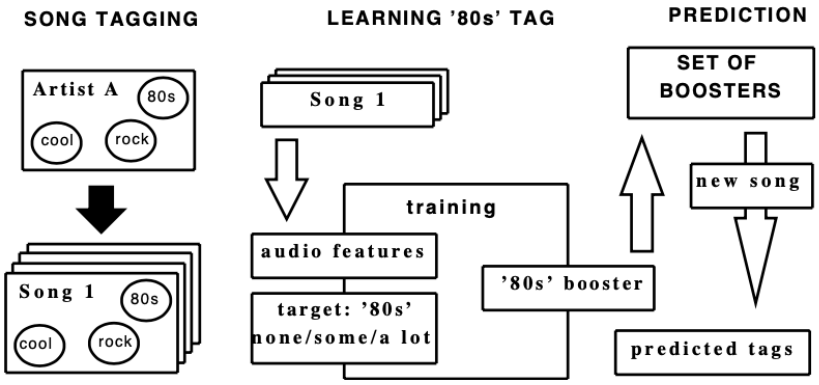


Figure 2.2: The AdaBoost model as used in [29]. Audio features are extracted from songs and then used to learn a booster per tag. Tags can be predicted for new songs using these boosters.

Improving upon this approach, Bertin-Mahieux et al. [9] replace AdaBoost with FilterBoost [14]. Instead of predicting *none*, *some*, or *a lot* they use binary classification per tag. Notably, the authors recognize that there may be correlations between tags. They propose two methods that can incorporate these correlations into the model. First, a second learning stage trains additional classifiers based on the output of the original tagger. Second, a reweighting based on the correlations computed between tags. Their results give evidence that automatically generated tags are of value in a music similarity task. However, they do not outperform the social tags on which they were trained. Additionally, there is some evidence that incorporating relationships between tags can improve model performance.

Hoffman et al. use the CAL500 dataset to train a model called Codeword Bernouilli Average (CBA) which improves upon the results by Turnbull and Bertin-Mahieux [42] by predicting the probability that a tag applies to a song based on a vector quantized representation of the song audio. Wang et al. propose a model that learns a Support Vector Machine (SVM) based model on a hypergraph, taking into account both music style and music tag correlations [82]. Similar to Eck et al. they find that incorporating tag information is beneficial for improving auto-tagging performance.

Law et al. note that most previous work has been using small, fixed, and clean vocabularies and argue that these vocabularies simplify the auto-tagging task and that bigger vocabulary obtained from social-tags or tagging games should be used instead [56]. They list three challenges when doing so: noise (synonymy, misspellings, semantic equivalence), sparsity (long-tail distributed tags), and scalability. To overcome these challenges, a new technique is introduced that can tag at a bigger scale. A Latent

[7] http://marsyas.info/downloads/datasets.html, GTZAN Genre Collection

Dirichlet Allocation (LDA) [12] model is used to group tags in more structured semantic topics. These topics are then used as training labels to predict topics from audio features. The method is efficient and achieves comparable performance to previous methods such as [9, 42, 75].

Weston et al. propose a model that jointly optimizes a set of tasks to map audio, artist names, and tags into a single low-dimensional representation [84]. They find that by optimizing their model using a ranking-based loss, performance improvements in ranked precision can be obtained for example in a tag prediction task. Similarly, Hamel et al. learn a latent representation using genre classification, music similarity, and music auto-tagging tasks.

2.2.4. Deep Learning Based Approaches

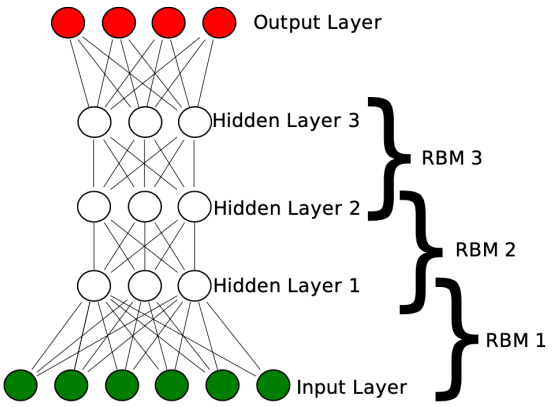
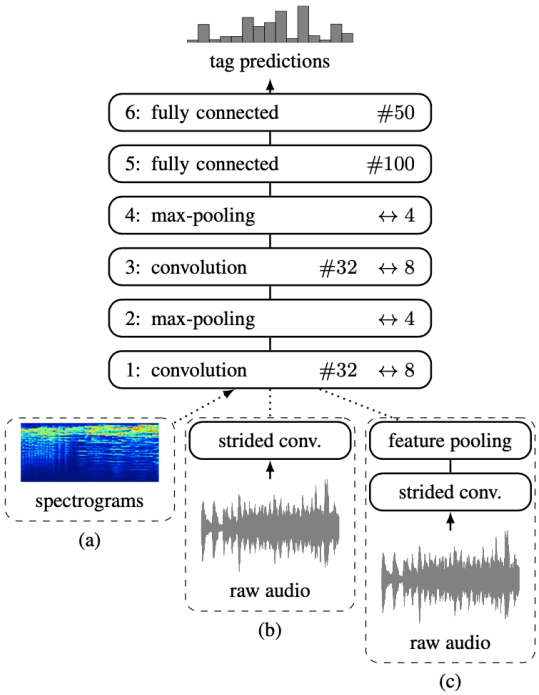


Figure 2.3: Example of a Deep Belief Network. DFT values are computed from the input audio and used as input for unsupervised training. Afterwards, the model is fine-tuned in a supervised way using the 10 classes from GTZAN. After model training, the activations of the hidden layers are extracted as an audio feature which can be used as input for other classifiers. Figure taken from [35]

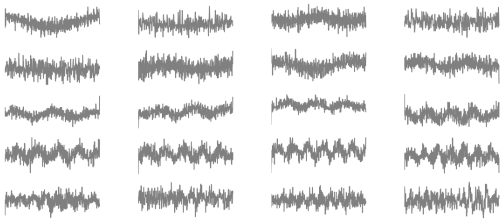
Implementations of this idea showed promising results. Hamel and Eck. perform feature learning using Deep Belief Networks (DBN) [35]. Using the 10 genres in GTZAN [76] and 25 most popular tags collected from the MajorMiner game [59] as training data. The DBN is first trained on Discrete Fourier Transform (DFT) input features in an unsupervised manner. Then, the model was fine-tuned and supervised using the GTZAN genre labels from the same training set. After training, the activations of hidden units in the network can be extracted and used as audio feature input for other classifiers. The robustness of these learned features is then evaluated by training an SVM based auto-tagger on the MajorMiner dataset, using the DBN as a feature extractor. An example of a DBN model is given in Figure 2.3. Results show that for most tags, the learned features outperform signal-processing-based features such as the MFCC-based ones. Similarly, Li et al. [57] use a Convolutional Neural Network (CNN) to learn features based on MFCC features for multiple adjacent frames of input audio from the GTZAN genre dataset. Henaff et al. use the GTZAN dataset to learn audio features via an unsupervised predictive sparse decomposition algorithm [40]. Using these features as input to an SVM, they find that their learned features perform as well as features that are manually computed in a genre classification task.

Dieleman et al. [26] propose a convolutional DBN learned to perform genre, artist, and music key detection using data from MSD. First, a DBN is learned, which is then converted into an MLP to learn the target task. The authors find that the convolutional nature of the network allows features to be summarized over time, improving task accuracy. Similarly, Hamel et al. [36] combine feature learning, time summarization, and tagging into a single model to obtain competitive results on MTAT. First, a set of discrete Fourier transforms (DFT) is preprocessed before the features are summarized over time by a pooling layer. These pooled features are then used as input to an MLP that is used for auto-tagging. They further improve performance by adding an MLP between the features and the pooling. The hidden layers of this MLP serve as feature learners. Further work by Dieleman et al. shows that features can be learned based on multiple timescales [24]. They find that different tags rely on different time scales.

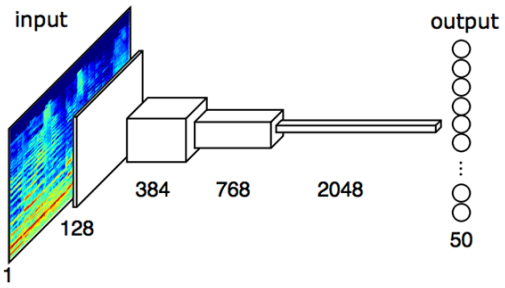
Van den Oord et al. [79] attempt to predict latent factors from music audio. Using Weighted Matrix Factorization (WMF) [44] they first compute latent factors which are then used as training input to deep learning-based models that can predict these latent factors from the music audio. In an auto-tagging task, they find that the learned features outperform the more traditional features. Similarly, in later work [80] a feature extractor is learned using MSD. The feature extractor is then used to obtain features to train an SVM. The features are then shown to work well for other datasets as well in a tag prediction task. In a related approach, Liang et al. [58] use a neural network as a feature extractor, but use this as an addition to the matrix factorization model instead of learning the feature from the matrix factorization model. The feature extractor is learned using an auto-tagging model.



(a) Example of the 1D CNN based architecture proposed by Dieleman and Schrauwen. Two different inputs are considered: a spectrogram and raw audio waveforms. For raw audio both a strided convolution and a strided convolution with feature pooling are considered. Figure taken from [25]



(b) Subset of filters learned in the lowest layers of the on CNN based on raw audio signals. Note that the model can learn frequencies. Figure taken from [25]



(c) Example of the 2D CNN based architecture proposed by Choi et al. Numbers represent the number of feature maps (depth) in each layer. Subsampling is used to decrease the size of the feature maps to 1 by 1 resulting in a feature of size 2048. Output is a prediction probability for each tag (top 50). Figure taken from [16]

Figure 2.4: Architecture and example filter output for CNN-based representation learning models.

Further work on CNN based feature extractors were performed by Dieleman and Schrauwen [25]. They investigate the possibility of learning features directly from audio signals in the form of waveforms and Mel spectrograms. A range of models based on a 1D CNN architecture is learned after which the authors find that these types of architectures can learn useful features from the input. They average predictions of multiple consecutive 3-second audio windows. Notably, the filters within the models can pick up frequencies. If a feature pooling layer is used the learned features are phase and translation invariant as well. The architecture and example filter output is given in Figure 2.4. Later work by Choi et al. further improves upon this idea, but uses 2D convolutions instead and considers a bigger 29.1-second audio signal at once [16]. An example of their architecture is shown in Figure 2.4c. Their model is competitive in auto-tagging and music classification for both MagnaTagATune and MSD and shows that deep neural networks can benefit from large amounts of training data, however, their research is limited by only using the top 50 most frequent tags. Further work by Choi et al. combines Recurrent Neural Network (RNN) with CNN to create a Convolutional Recurrent Neural Network (CRNN). RNN are commonly used to model sequential data [17]. The CRNN model is shown to outperform 1D and 2D CNN based models with the same number of parameters, however, research was again limited by only using 50 tags. Pons et al. look into how CNN based model can be learned while taking into account the different properties of musical aspects [65]. They find that by carefully choosing the dimensions of the CNN filter shapes, they can learn temporal features, frequency features, and both at the same

time. By carefully designing an architecture, they achieve results that are competitive on the Ballroom dataset [33]. Later work by Pons et al. dives deeper into deep music auto-tagging architectures [66] arguing that when a small amount of training data is available, models can be constrained. They use the top 50 tags of both MagnaTagATune and MSD, and a proprietary dataset consisting of 1.2 million songs, annotated using 139 human-expert tags. They find that if a large dataset is available, models that rely directly on waveform input can outperform models that operate on a computed spectrogram, indicating that applying signal processing before learning the model may constrain some models.

2.3. Models

Work on auto-tagging has been influenced by various types of models. In this section, we discuss two of them, Deep Neural Networks (DNN) and Matrix Factorization (MF). Additionally, we discuss a recently introduced class of DNN that can operate on graphs. These models are the basis of our methodology, which will be discussed in the next chapter.

2.3.1. Deep Neural Networks

Deep Learning is a branch of machine learning that focuses on neural network-based models. The majority of deep learning algorithms can be described as a combination of input data, a loss function, an optimization procedure, and a neural network architecture [32].

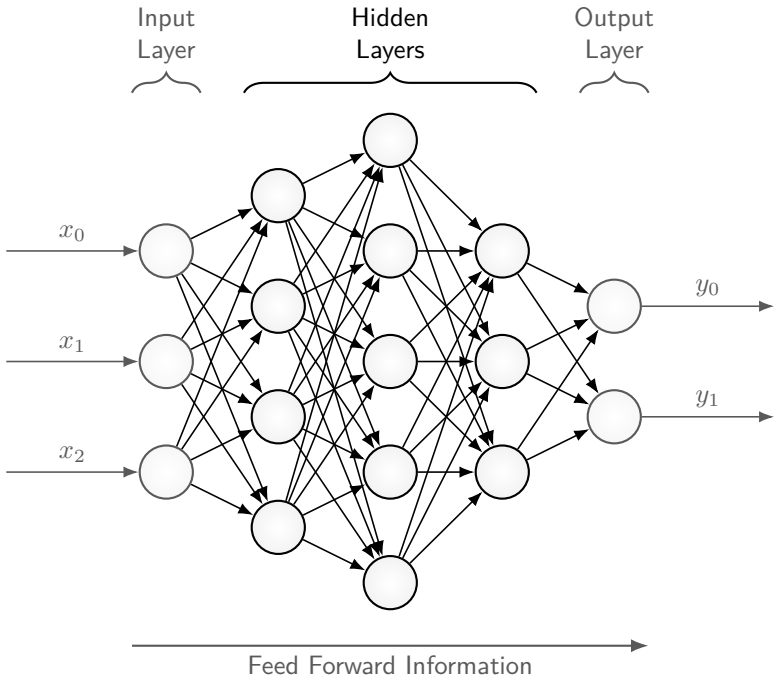


Figure 2.5: Example MLP architecture of a function $\mathbf{y} = f^*(\mathbf{x}; \Theta)$. Features are fed into the input layer and are then fed through the network. Each layer contains a variable number of nodes (width) and the number of hidden layers (depth) can vary. Θ are the learned weights that determine the output of each node.

The most simple form of a deep learning model is a Feed Forward Neural Network (FFNN), or Multi-Layer Perceptron (MLP). The goal of the algorithm is to learn a set of parameters Θ such that the network approximates a function $\mathbf{y} = f^*(\mathbf{x}; \Theta)$ that maps some input $\mathbf{x}$ to an output $\mathbf{y}$. These functions can be composed as a chain such that they form a network of layers. For example, we can chain three functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ to form $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$[8]. Usually, this function applies a nonlinear transformation to the input $\mathbf{x}$, commonly in the form $g(\mathbf{x}\mathbf{W}^\mathbf{T} + \mathbf{b})$ where $\mathbf{W}$ is a matrix of learnable weights, $\mathbf{b}$ is a learnable bias term and g is a nonlinear activation function.

The first function is typically referred to as the input layer, which accepts the input data. The final layer is referred to as the output layer. All layers that do not have an output as dictated by the training data are referred to as the hidden layers because their output is not directly observed. The functions usually take vector data in the form of features as input. The dimensionality of this input determines the number of units and the width of the layer. The optimization procedure should then optimize the weight matrices in the network such that they minimize a loss function. An example of an MLP model can be found in Figure 2.5.

Various deep learning-based architectures exist and have found their way into the MIR field such as the CNN and DBN used in the feature learning algorithms described in Section 2.2.4.

[8]Each function has learnable parameters Θ, but we omitted them for clarity.

2.3.2. Matrix Factorization

Recommender systems can traditionally be divided into two categories: content-based and collaborative filtering [51]. An example of a content-based recommendation can be found in the MGP which was discussed earlier in Section 2.1.2. The tags assigned to songs can be used to build user profiles that can then be used for recommendation. Alternatively, collaborative filtering attempts to model the relationships between users and entities to predict new user-item relationships. Two common methods of collaborative filtering are neighborhood and latent factor models. The former attempts to recommend based on scores for similar (neighboring) items. Alternatively, latent factor models attempt to model relationships by taking both users and items into account by predicting factors that explain the user-entity relationships. An example of a latent factor model is Matrix Factorization (MF), which attempts to learn factors that can be used to reconstruct the input matrix. There are two types of input data that we can consider for MF based recommender systems. The first is explicit feedback where the user has explicitly made clear their preference for the item. An example of this is rating a movie or pressing a like button. The second is implicit feedback, which indirectly reflects user preference for an item. For example, the number of listens to a song or the purchase history in a webshop.

As shown during the famous Netflix Prize[9] competition, latent factor models based on MF have shown to be very successful [51]. During the Netflix Prize participants were asked to improve upon the recommender system used by the movie streaming service Netflix. The goal of an MF model is to map a set of users and items to a joint latent factor space that has dimensionality d. Item i is mapped to a vector $\mathbf{q_i} \in \mathbb{R}^d$ and user u is mapped to a vector $\mathbf{p_u} \in \mathbb{R}^d$ such that the inner product $\hat{r}_{ui} = \mathbf{q_i^T p_u}$ between the user and item vectors models the relationship between those entities. The output $\hat{r}_{ui}$ can then be used to serve recommendations to the user. One way to learn latent vectors is by minimizing a loss function such as the regularized squared error:

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - \mathbf{q_i^T p_u})^2 + \lambda(\|\mathbf{q_i}\|^2 + \|\mathbf{p_u}\|^2) \tag{2.1}$$

Where κ refers to the set of pairs (u,i) for which we have explicit input data r_{ui} in the training dataset, and λ is the hyperparameter controlling the amount of regularization.

Typically, MF models perform well on "in-matrix" predictions, meaning that they can only recommend items that were present in the training set. This means that recommending new "out-of-matrix" items is not possible for these kinds of models [58].

Within the MIR literature there are applications related to music that make use of matrix factorization algorithms. An example is a model discussed earlier in Section 2.2.4, by van den Oord et al. which attempts to predict latent factors based on music audio [79]. They first use Weighted Matrix Factorization (WMF) [44] to obtain latent factors and then use these factors as input to deep learning models to be able to predict them directly from the song audio without requiring access to a full user-item matrix, thus solving the out-of-matrix prediction problem. WMF is designed to take implicit input into account. It minimizes a similar equation as Equation (2.1): Typically, MF models perform well on in-matrix predictions, meaning that they can only recommend items that were present in the training set. This means that recommending new items out-of-matrix is not possible for these kinds of models.

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} c_{ui}(a_{ui} - \mathbf{q_i^T p_u})^2 + \lambda(\|\mathbf{q_i}\|^2 + \|\mathbf{p_u}\|^2) \tag{2.2}$$

Where a_{ui} is a binarized value of r_{ui} where r_{ui} is one if the item has been consumed by the user, and 0 otherwise. c_{ui} is a confidence vector that measures the confidence of observing a_{ui}. An example is $c_{ui} = 1 + \alpha r_{ui}$, which increases the confidence if r_{ui} becomes larger (if a user has listened more to a song for example). α is a constant that controls the rate of confidence increase. Building upon the approach by van den Oord, Liang et al. propose a "content-aware" collaborative music recommendation algorithm [58]. It differs from the model by van den Oord et al. in that it obtains item (song) latent factors by using a feature extractor that is trained on a tagging task, instead of a model that predicts latent factors based on the audio, hence the term "content-aware". Liang et al. argue that learning the latent factors directly from WMF constrains the recommendation model to the performance of WMF and that it is therefore better to use a representation of the audio itself as input to the MF model. The model by Liang et al. is a probabilistic MF model [68] based on the Collaborative Topic Regression

[9] The Netflix Prize, https://www.netflixprize.com

(CTR) model by Wang and Blei [81]. Their model consists of two components: a probabilistic matrix factorization model and Latent Dirichlet Allocation (LDA) [12] for topic modeling. The topics are used as input to the collaborative filtering model. The approach by Liang et al. has a two-stage approach similar to Wang and Blei but differs primarily in how the latent factors for the items are obtained.

We will describe the model by Liang et al. in more detail in Chapter 3. Both the approaches by van den Oord and Liang are not restricted by the cold-start problem in item recommendation because they can use the item content to either approximate a latent factor, or replace the latent factor making them useful for music recommendation. Other models also make use of latent factors to obtain recommendations similar to traditional MF models. The model by Weston et al. discussed in Section 2.2.3 attempts to learn embeddings by training a semantic embedding model on multiple tasks, and uses the learned shared parameters to obtain a ranked list of recommendations [84].

2.3.3. Graph Representation Learning

Graphs are an important data structure in computer science used to model relationships between entities. Naturally, machine learning can be applied to graphs, for example, to perform edge prediction or to classify individual nodes. Similar to auto-tagging literature, researchers have found that manually computed node or edge features such as graph statistics are limited. This has led to modern approaches that attempt to learn representations of graphs directly. Hamilton et al. define these approaches as an encoder-decoder framework sharing similarity with Matrix Factorization models [38]. Their framework consists of the encoder function, decoder function, loss function, and a pairwise node similarity function.

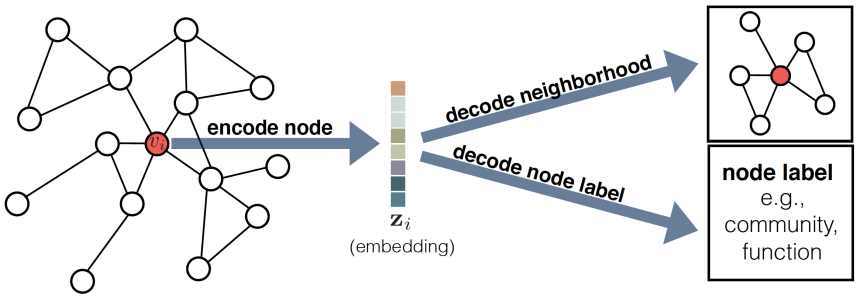


Figure 2.6: Schematic of the encoder-decoder framework described by Hamel et al. The encoder maps each node to an embedding which is then decoded into a similarity metric. By optimizing this system, the encoder learns to compress graph structure into an embedding that can be used as input to other systems. Figure taken from [38]

Input to the framework is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with binary adjacency matrix $\mathbf{A}$ and feature matrix $\mathbf{X} \in \mathbb{R}^{m \times |\mathcal{V}|}$ where m is the dimensionality of a feature associated with each node. The encoder is a (learnable) function that maps each node $v_i \in \mathcal{V}$ to an embedding $\mathbf{z_i} \in \mathbb{R}^d$

$$encoder : \mathcal{V} \rightarrow \mathbb{R}^d \tag{2.3}$$

The decoder function then takes as input these node embeddings and decodes them into a similarity measure that estimates the relationship between two nodes. The type of decoder can vary, but it is often a pairwise decoder similar to the one discussed in Section 2.3.2:

$$decoder : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+ \tag{2.4}$$

Then, similar to Matrix Factorization a loss function is minimized for a set of training node pairs $\mathcal{D}$ such that the pairwise decoder reconstructs the similarity metric $s_{\mathcal{G}}$. An example of such a similarity metric is the graph adjacency matrix $\mathbf{A}$.

$$\mathcal{L} = \sum_{(v_i, v_j) \in D} l(decoder(\mathbf{z_i}, \mathbf{z_j}), s_{\mathcal{G}}(v_i, v_j)) \tag{2.5}$$

where $l : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, a loss function that measures the error between the output of the decoder and the similarity metric. This function is then optimized, and the learned encoder can be used to obtain node

embeddings that can be used as input to further machine-learning tasks such as node classification or edge prediction. A visualization of the framework can be found in Figure 2.6.

Random walk-based approaches such as DeepWalk [63] and node2vec [34] fit into this framework. These approaches traditionally have been able to obtain competitive results on graph datasets. However, they are unable to use node features as information during the encoding process. Additionally, they are transductive: they can only generate embeddings for nodes that have been observed during the training phase. Recently a new class of models was proposed that employ deep learning on graphs outperforming many existing methods on popular graph datasets [48, 49, 69, 78]. These so-called convolutional encoders do not rely on the entire graph but on the node's local neighborhood. They generate node embeddings for each node by summarizing the information from the neighboring nodes. This information can be basic graph statistics such as degrees, or dedicated features that describe the entity. This operation can be seen as a differentiable form of a message-passing framework with a forward pass of the following form:

$$\mathbf{h_i^{(l+1)}} = \sigma\left(\sum_{m \in \mathcal{M}_i} g_m(\mathbf{h_i^{(l)}}, \mathbf{h_j^{(l)}})\right) \tag{2.6}$$

Where σ is a (nonlinear) activation function, $\mathbf{h_i^{(l)}} \in \mathbb{R}^{d^{(l)}}$ is the hidden representation of node v_i in the input graph and g_m is an accumulator for incoming messages $m \in \mathcal{M}_i$ for node i. g_m can be a (basic) neural network such as $W\mathbf{h_j}$

Since these models rely only on the local node neighborhood, they extend beyond the original training graph and can be used to generate embeddings for previously unseen nodes. Examples of successful models are (Variational) Graph Auto-Encoders [48] and GraphSAGE [39].

3

Methodology

In this chapter, we discuss the limitations of auto-tagging and our methodology for contributing to a solution. We start by listing the current limitations that exist in the field of auto-tagging and how we contribute to solving them. We formalize the problem and introduce the semi auto-tagger framework as a possible solution. Finally, we discuss our approach by suggesting models that fit into this framework.

3.1. Limitations of auto-tagging

The cold-start problem in music recommendation [29] is a problem where songs that have not previously been seen by a recommendation system are difficult to recommend. For songs that have not been listened to by system users, only audio content information such as waveforms can be considered for recommendation, limiting how recommendations can be made. Additional (meta) data such as descriptive tags can be used to expand the solution space by putting songs into the right context to similar songs, for example, based on genre. If a song is tagged with tags that describe its content, these tags can be used to connect it to similar songs that the recommender system has seen before such that recommendations can be made. To make good recommendations, we require a rich and detailed description of a song. At the song level, however, we encounter a similar cold-start problem. New songs may not be tagged, or sparsely tagged. Auto-tagging algorithms seek to mitigate this problem by automatically assigning tags to songs based on their audio contents.

Current solutions to the auto-tagging problem in the MIR field can be seen as limited in several ways, primarily from a tag vocabulary perspective. As one of the goals of an auto-tagging algorithm is to alleviate the cold-start problem, one can argue that being able to tag from a large and varied vocabulary makes auto-tagging more useful in a recommendation setting. However, it is difficult to obtain a large and varied vocabulary to learn from which undermines the usefulness of such a system. As noted in Section 2.1.3 the ideal vocabulary is big and contains a diverse set of tags. Current vocabularies, however, are either large and noisy or small. Commonly used datasets such as those listed in Table 2.2 lack the diversity needed to come up with rich descriptions. One big exception to the lack of richness and diversity can be found in the Million Song Dataset (MSD). MSD contains a very big vocabulary of tags compared to the other datasets. The two most common datasets used in auto-tagging literature are MSD and MagnaTagATune (MTAT). While MSD is large and diverse, it is also noisy due to the way it was collected (Last.fm users). MTAT is cleaner as a result of it being obtained via a tagging game, but its vocabulary is much less rich and detailed compared to MSD. It has been shown that there is merit in using big subsets of these datasets [19], nevertheless the majority of current auto-tagging research limits itself to relatively small subsets [16, 25, 28, 29]. Reasons given for limiting the dataset sizes vary: smaller subsets are easier to analyze [28], tag sparsity [29], dataset imbalance [16] or size of the output space [19].

We observe three topics that we think should be improved upon:

- **Clean datasets are scarce**: Annotations provided by domain experts are cleaner as discussed in Section 2.1.2 because the tagging process relies on the musical knowledge and expertise of experts. An issue with experts is that they are an expensive human resource, and they can only tag at a limited rate. Using experts also limits the vocabulary that can be used. Ideally, we have a clean dataset that is obtained by skilled experts such as is attempted at Pandora's MGP [1, 66]. However, such datasets are a valuable resource which makes them often proprietary or not readily available at a scale optimal for MIR research.
- **The size of the vocabularies used**: The majority of works limit their use of the vocabulary. Especially in the case of MSD, more than 99% of the five hundred thousand available tags in the vocabulary are ignored in the majority of works. By doing so, one can ask if such systems can be used in a real-world production environment where a much larger and more diverse set of tags exists. Bigger vocabularies are common in other fields such as image classification [52], which is similar to auto-tagging.
- **Context provided by tags is not considered**: Auto-tagging algorithms traditionally operate on audio content only. However, auto-tagging datasets can be interpreted as a large network of songs and tags in which songs are grouped using the context provided by tags. Such a network contains potentially useful implicit relationships between tags that can then be used to predict additional tags that are close within the graph. Naturally, if the size of the vocabulary is limited, the number of relationships that can be exploited becomes smaller.

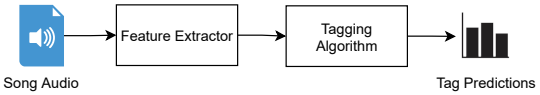
We aim to contribute to reducing the impact of these limitations by extending the auto-tagging problem to a graph prediction problem. By modifying the auto-tagging problem to allow external (expert) input in the form of seed tags, we allow the relationships between tags to be considered. By using the network of songs and tags we may be able to use less clean datasets such as MSD to predict tags, as noisy tags are expected to have less importance in the graph. Additionally, such a system can be used to allow experts to tag at a higher rate while still maintaining their tagging accuracy. They can suggest input tags to the system and will be provided with recommendations. If the expert does not agree with the input, they are free to further complement the output with their expertise. By modeling the task as a completion task, it shares similarities to a recommender system. Recommender systems naturally allow for a higher number of tags to be considered because they have traditionally been able to recommend from large pools of items. Additionally, traditional recommender algorithms such as matrix factorization, model the relationships between items and are therefore able to exploit the relationships that exist between tags. While addressing these limitations we specifically aim to produce solutions that are feasible to be deployed in a production scenario, hence we should make sure proposed models can work for previously unseen songs and do not require a long period of retraining after they have been deployed.

3.1.1. Formalization: Semi Auto-Tagging

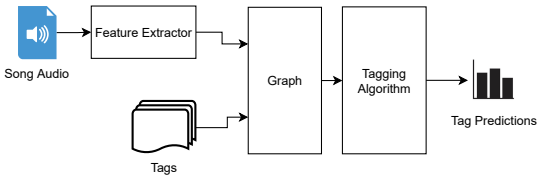
We define the semi auto-tagger as an extension of the auto-tagging problem we defined in Section 2.2.1. We extend it to become a semi auto-tagger which takes a bipartite graph of song and tag nodes as input. For a previously unseen song, the goal of a semi auto-tagger is to complement the list of input edges (tag assignments) by making use of both context and content.

> Given a set of N training songs, nodes $\{s_1, \ldots, s_i, \ldots, s_N\} \in \mathcal{S}$ which are annotated using a vocabulary of V tags $\{t_1, \ldots, t_v, \ldots, t_V\} \in \mathcal{T}$. Each song is associated with a content feature vector $\mathbf{x_i} \in \mathbb{R}^f$ where f is the dimension of the feature. Tag annotations are represented by a set of edges $(s_n, t_m) \in \mathcal{E}$. Together they define a bipartite training graph $\mathcal{G} = (\mathcal{S}, \mathcal{E}, \mathcal{T})$. A semi auto-tagging task then becomes an edge prediction task within this graph. For prediction, U previously unseen songs are added to form $\mathcal{S}' = \{s_1, \ldots, s_j, \ldots, s_U\} \cup \mathcal{S}$, in addition to contextual expert input as $\mathcal{E}' = \{(s_u, t_v), \ldots, (s_U, t_v)\} \cup \mathcal{E}$ edges. Tag prediction then becomes an edge prediction task for songs in $\mathcal{S}'$ in the extended graph $\mathcal{G}' = (\mathcal{S}', \mathcal{E}', \mathcal{T})$.

Naturally, expert input can be provided by adding new song nodes and constructing edges between these song nodes and the existing tags (the expert input). The goal of the semi auto-tagger is then to



(a) Schematic overview of an auto-tagging algorithm. First, features are extracted from the song audio (content), after which a tagging algorithm uses this to predict tags.



(b) Schematic overview of a semi auto-tagging algorithm. The system has two inputs: features extracted from the song audio (content) and a set of tags connected to the song-tag graph of the dataset (context). The tagging algorithm makes use of both inputs via the graph to come with a set of tag predictions.

Figure 3.1: Comparison between a traditional auto-tagging algorithm and the proposed semi auto-tagging framework.

complement the list of edges starting at the new song node with additional relevant song-tag edges. A schematic overview of an auto-tagger and a semi auto-tagger is shown in Figure 3.1.

3.2. Models

We propose two models that fit into the semi auto-tagger framework. The first model is the "Content-Aware" Matrix Factorization model by Liang et al [58]. This model combines MF with a content-based feature extractor to recommend songs to users. Since it has been used successfully for a music-related task before, we think it is a good candidate to implement as a semi auto-tagger. A similar model by van den Oord et al. has been used to predict song latent factors successfully in the MIR field before [79]. Our second model is based on recent advancements in the graph representation learning field, the Graph Convolutional Network (GCN). It shares similarities with MF but is closer to deep-learning-based approaches. Additionally, it has shown competitive results on datasets in the graph learning field [48, 49, 69, 78].

Both models can combine audio content with context provided via tags by modeling similarities between songs in the form of latent factors. We will now discuss the two candidates in the next subsections.

3.2.1. Content-Aware Matrix Factorization

The content-aware recommendation model by Liang et al. consists of a two-stage approach similar to that of Wang and Blei [81] First, a content-based supervised tagging model is learned on a tagging task. This model is then used to extract the feature from the audio (items). This replaces the topic model. Then, this feature is used as input to the collaborative filtering model for prediction.

Supervised pre-training

The supervised model is trained on a tag prediction task. An MLP is trained using 370K tracks from MSD, using 561 tags from the Last.fm subset. As input Echonest timbre features[1] contained in the MSD are used. These features share similarities with MFCC features. The activations of the last layer of the MLP are used as input song latent factors to the MF model. In our implementation, however, we opt to directly input audio features to reduce the number of models that have to be trained for evaluation.

Content-Aware Collaborative Filtering

For clarity, we translate the notation used by Liang et al. to the location we used in Section 2.3.2. The process for training the model is then:

- For each user u, draw a user latent factor:

$$\mathbf{p_u} \sim \mathcal{N}(0,\lambda_p^{-1} I_K) \tag{3.1}$$

- For each song i, draw a song latent factor:

$$\mathbf{q_i} \sim \mathcal{N}(W\mathbf{x_i},\lambda_q^{-1} I_K) \tag{3.2}$$

- For each user-song pair (u, i) draw implicit feedback:

$$r_{ui} \sim \mathcal{N}(\mathbf{p_u^T q_i},c_{ui}^{-1}) \tag{3.3}$$

Where $\mathbf{x_i}$ is the input content feature vector (song latent factor), $\mathcal{N}$ is the normal distribution, K is the latent space dimension, $W \in \mathbb{R}^{K \times F_h}$ is a learnable weight matrix which is used to transform the feature representation of the audio into the collaborative filtering latent space, and λ_q defines how much the song latent factor deviates from the content feature. $c_{ui} = 1 + \alpha r_{ui}$ is a confidence factor which is similar to the heuristic used by Hu et al. [44]. α can be optimized using an optimization procedure (hyperparameter tuning). After optimization three matrices are obtained:

- The concatenated user latent factor matrix

$$P \triangleq [p_1|\cdots|p_U] \in \mathbb{R}^{K \times U}$$

[1] http://millionsongdataset.com/pages/field-list, description of features available in MSD. Note: the original website describing the features from Echonest is no longer available.

- The concatenated song latent factor matrix

$$Q \triangleq [q_1|\cdots|q_I] \in \mathbb{R}^{K\times I}$$

- The learned weight matrix

$$W \in \mathbb{R}^{K\times F_h}$$

For specific optimization details on how to learn these matrices, we kindly refer to [58]. Since the learned song latent factors are dependent on the contents of the audio, we can now easily make both in-matrix and out-of-matrix predictions after training the model. We can make in-matrix predictions by multiplying the user and item factors $\hat{r}_{ui} = \mathbf{p_u^T q_i}$. Since $\mathbf{q_i}$ is derived from the content vector using the learned weight matrix, we can make out-of-matrix predictions for new items as $\hat{r}_{ui} = \mathbf{p_u^T}(\mathbf{W x_i})$.

Adaptation for semi auto-tagging

Two adaptations to the model have to be made to make the model feasible for semi auto-tagging. First, by replacing the user-song matrix with a tag-song matrix, we recommend tags to songs, instead of songs to users. Second, when contextual expert input is given, we adapt the model by rerunning the optimization step for Q, the song latent factors based on the new song-tag matrix with expert input without updating P and W. The updated song latent factors can then be used to make predictions for the new song.

3.2.2. Graph Convolutional Network

The class of graph-based deep learning models introduced in Section 2.3.3 has shown competitive results in node classification and link prediction tasks. It is however not suited for graphs with multiple types of nodes and edges. The Relational Graph Convolutional Network (R-GCN) model proposed by Schlichtkrull et al. [69] proposes a variant of these models that can deal with these graphs. Similarly, Berg et al. propose a bipartite variant that can be used for matrix rating completion [78]. Our approach makes use of the R-GCN model and fits into the encoder-decoder framework we discussed in Section 2.3.3.

Relational Graph Convolutional Networks

The R-GCN model by Schlichtkrull et al. [69] is an extension of the GCN model by Kipf and Welling [49] that enables this model to use large scale multi-partite relational data, such as knowledge graphs. The model operates on a directed multi-graph $G = (\mathcal{V},\mathcal{E},\mathcal{R})$ where $v_i \in \mathcal{V}$ are graph nodes, $(v_i, r, v_j) \in \mathcal{E}$ are relationships between nodes and $r \in \mathcal{R}$ contains bi-directional relationship types. The model operates by aggregating messages passed between neighboring nodes for each relationship type and applying a non-linearity. An encoder-decoder framework is used to learn a link prediction task. The encoder step of an R-GCN model consists of a forward-pass update of the following form, which can be chained:

$$\mathbf{h_i^{(l+1)}} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{n_{i,r}}W_r^{(l)}\mathbf{h_j^{(l)}} + W_0^{(l)}\mathbf{h_i^{(l)}}\right) \tag{3.4}$$

Where σ is a (nonlinear) activation function, $\mathbf{h_i^{(l)}} \in \mathbb{R}^{d^{(l)}}$ is the hidden representation of node v_i in layer l of the neural network, with dimensionality $d^{(l)}$, $\mathcal{N}_i^r$ is the set of neighbor node indices for node i for relationship r, $n_{i,r}$ is a normalization constant that can be learned or pre-computed, $W_r^{(l)}$ is a weight matrix for relationship r and $W_0^{(l)}$ is a weight matrix for a self-loop to the source node. The input to the first layer is a feature vector $\mathbf{h_0} = \mathbf{x}$ per node. The model can then be optimized by running the update (in parallel) for each node in the graph. The output of the encoder step is for each node $v_i \in \mathcal{V}$, a node embedding $\mathbf{e_i} \in \mathbb{R}^d$. The decoder can now predict a score per edge similar to MF based models by scoring (*subj*, *rel*, *obj*) triples using a function $f : \mathbb{R}^d \times \mathcal{R} \times \mathbb{R}^d \to \mathbb{R}^+$. An example of such a function is DistMult [85], where each relation r has a learnable diagonal matrix $R_r \in \mathbb{R}^{d\times d}$. A prediction for relationship triple (*subj*, *rel*, *obj*) can then be made as $f(subj,rel,obj) = \mathbf{e_{subj}^T} R_{rel} \mathbf{e_{obj}}$.

Adaptation for semi auto-tagging

We adapt the R-GCN in two ways for semi auto-tagging. First, our song-tag graph is bipartite and only a single bidirectional relationship type exists: (*song, tagged, tag*). Therefore the decoder can be simplified by simply computing the dot product in the same way as MF by omitting R_r. Second, the R-GCN can operate on input content features per node. However, we do not have input content features for tags. Since in our framework the number of tags is fixed, we replace the tag features with a learnable weight matrix.

4

Experimental Setup

In this chapter, the experimental setup is discussed. We explore the feasibility of the semi auto-tagging framework that was introduced in Section 3.1.1 by implementing the models discussed in Section 3.2 for various hyperparameter configurations and tag vocabulary sizes. We compare these models against a content-based auto-tagging baseline, an MLP. We start by giving an overview of the datasets we use in our experiments. Second, we discuss our experimental design and address the research questions. Third, we give implementation details for the models discussed in Chapter 3. Finally, we revisit the research questions and address how our experiments aim to contribute to solving them.

4.1. Datasets

In this work we make use of two datasets that are frequently used in auto-tagging literature: the Last.fm subset contained within Million Song Dataset (MSD) [5, 11], and MagnaTagATune (MTAT) MTAT [6, 55]. Both datasets are composed of a set of tags, and multiple audio tracks (hereafter referred to as songs). The datasets differ in both vocabulary size and the variation of tags available in the vocabulary. In the next subsections, we will discuss feature selection and the properties of both MSD and MTAT. We also discuss dataset-specific pre-processing and limitations.

4.1.1. Features

To see if the selection of semi auto-tagging models can efficiently make use of content feature information, we compare an audio content feature against a random feature. If the random feature is given to the model, its performance is expected to be lower compared to a feature that is based on the meaningful audio content information. For our audio content feature, we compute delta-MFCCs by extracting 20 MFCC [21, 46] coefficients[1] for which we compute the first and second derivatives. We then take the average and standard deviation over the time dimension and concatenate them to obtain a 120-dimensional feature. For our random feature, we sample a 120-dimensional vector from a uniform distribution in the range $[0, 1)$. Both feature sets were standardized.

[1] MFCC coefficients were computed using Librosa[60]

4.1.2. Million Song Dataset (Last.fm)

The Last.fm subset contained within MSD consists of 943,347 songs matched between MSD and the Last.fm API. From this set of tags, 505,216 tracks have at least one tag assigned from a set of 522,366 unique tags. The dataset contains 8,598,630 relationships between tracks and tags [5] which were obtained from real users of Last.fm. Track audio files are not readily available and need to be obtained separately from the dataset. We have been able to obtain a subset of 370,710 song preview MP3 files with a maximum length of 30 seconds from 7Digital[2] using the identifiers available in MSD. The large vocabulary of available tags, combined with the fact that they are crowdsourced from real users, make that tags are distributed in a long-tail fashion. Due to this method of obtaining, the data is also relatively noisy as tags are based on unrestricted text input from Last.fm users. Figure 4.1a shows the distribution of the 1024 most frequent tags in MSD. Note that more than half of the tags are in the top list of 50% least frequent tags, but still have a relatively high frequency. The availability of many song(track)-tag relationships makes the dataset a good candidate for training a semi auto-tagger due to the context that is provided via relationships in this network. A subgraph of the dataset, showing the network for 4 songs from a similar genre, can be found in Figure 4.2a. We applied the following dataset-specific preprocessing steps:

- Tracks for which we were unable to obtain an audio sample were excluded.
- Tracks shorter than 10 seconds, or with corrupted audio files that could not be read were excluded.
- 7 subsets were created based on the number of top n most frequent tags. For each n, we select the set of top n tags from the vocabulary and the songs that have been assigned a tag from this set. We then include the tags, songs, and relationships between them into our subset. We used the following values for n: $n \in \{16, 32, 62, 128, 256, 512, 1024\}$.
- Tracks that do not have tags assigned, or that do not have tags assigned after creating a subset were excluded.

The number of songs, tags, and unique relationships contained in our subsets are given in Table 4.1. Note that even though the subset sizes are limited, the number of relationships increases rapidly as the subset size increases. Due to the limited subset size and the exclusion of tracks, not all song previews we obtained can be used in experiments.

dataset	# unique tags	# songs	# relationships
msd16	16	199,219	529,758
msd32	32	229,580	821,239
msd64	64	261,534	1,193,315
msd128	128	286,911	1,639,166
msd256	256	303,817	2,072,286
msd512	512	318,526	2,498,481
msd1024	1,024	327,831	2,924,849

Table 4.1: Properties of the Million Song Dataset subsets.

[2] 7Digital, http://www.7digital.com

4.1.3. MagnaTagATune

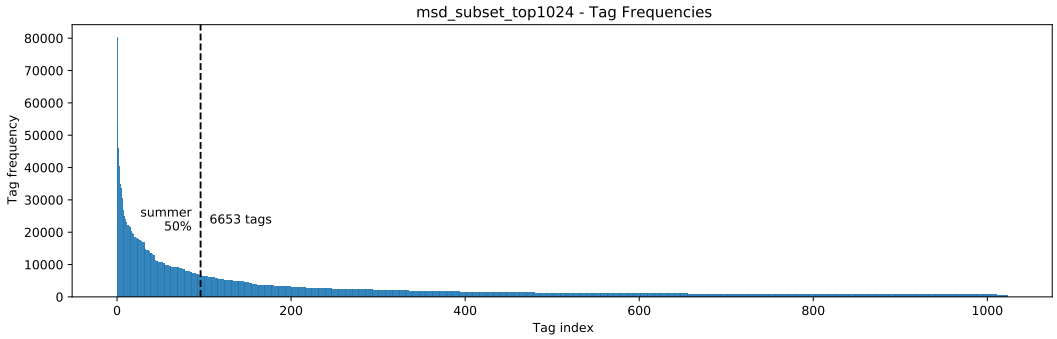
The MagnaTagATune (MTAT) dataset [6, 55], contains 188 tags collected using the TagATune tagging game for a collection of 25,863 clips from 5,404 mp3 files. The dataset contains 89,395 relationships between songs (clips) and tags. 4,221 of the clips contained in the dataset do not have tags assigned, and can therefore not be used in our semi auto-tagging task. Figure 4.1b shows the distribution of all tags in MTAT. Note that the distribution of tags follows a similar long-tail (power-law) like distribution as is found in MSD. Figure 4.2b shows how relationships between songs and tags within MTAT can be used to find tags that can be shared among songs. The following dataset-specific pre-processing steps were applied:

- Clips that do not have tags assigned were excluded.
- 5 subsets were created based on the number of top n most frequent tags. For each n, we select the set of top n tags from the vocabulary, and the clips that have been assigned a tag from this set. We then include the tags, clips, and relationships between them in our subset. We used the following values for n: $n \in \{16, 32, 62, 128, 188\}$.

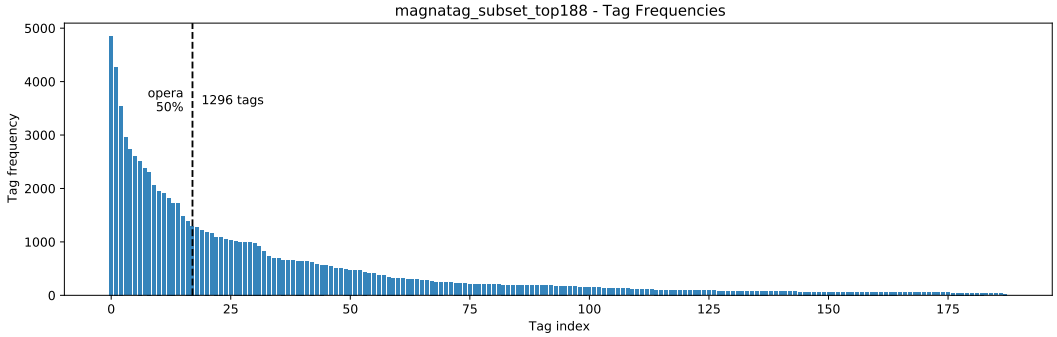
The number of clips, tags, and unique relationships are given in Table 4.2. Note that clips that do not have tags assigned were omitted.

dataset	# tags	# clips	# relationships
mtat16	16	18,585	40,812
mtat32	32	20,639	58,513
mtat64	64	21,257	75,069
mtat128	128	21,546	85,774
mtat188	188	21,642	89,395

Table 4.2: Properties of the MagnaTagATune subsets.

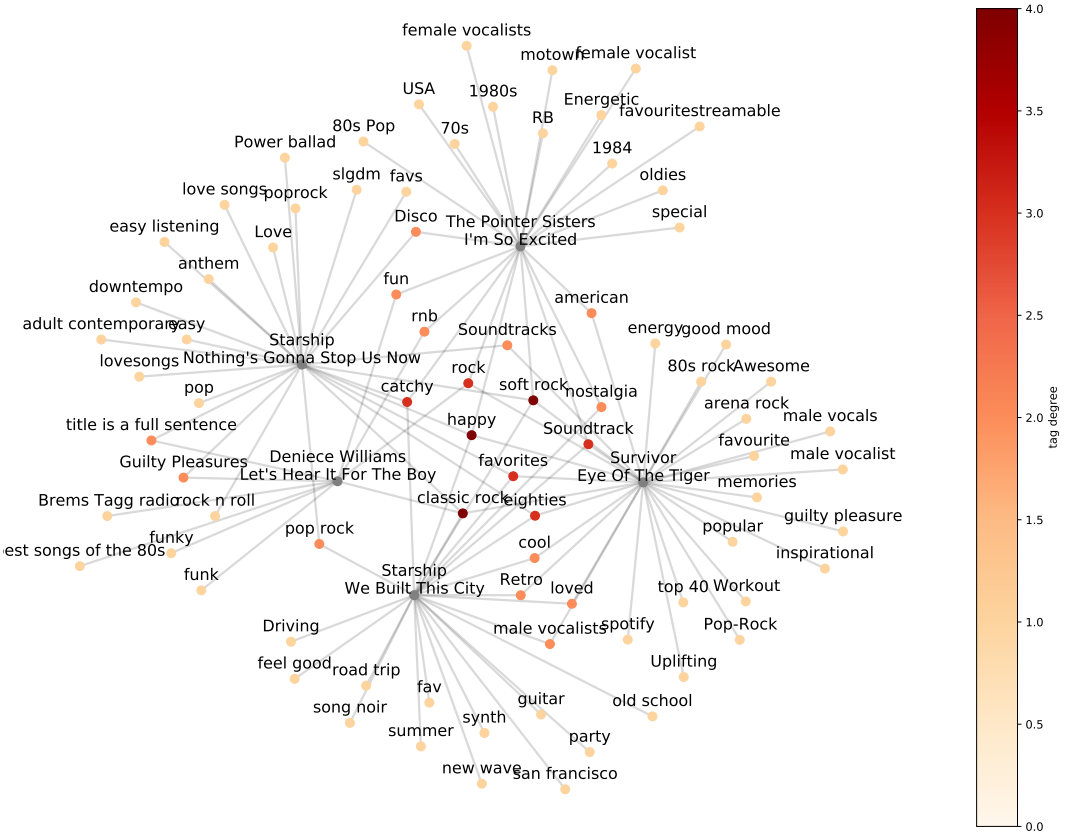


(a) Tag frequencies for the top 1024 tags contained within MSD. Note that more than 50% of tags can be found in the long tail. Here, we define the long tail as all tags that have a frequency lower than 6,653.

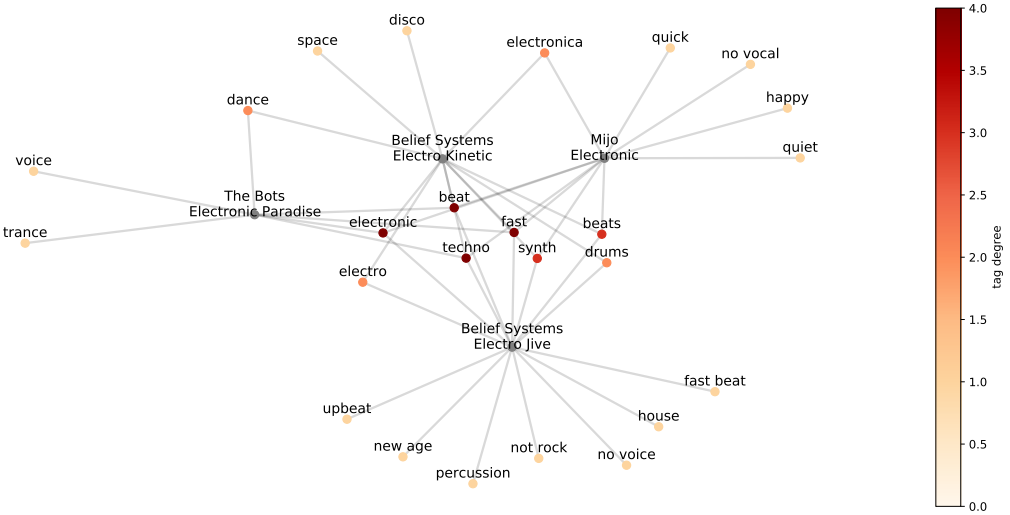


(b) Tag frequencies for the top 188 (all) tags contained within MTAT. Note that similar to MSD, the majority of tags can be found in the long tail. Here, we define the long tail as all tags that have a frequency lower than 1,296.

Figure 4.1: Tag frequencies for both MSD and MTAT. Note that for both datasets the majority of distinct tags can be found in the long tail.



(a) Subgraph of MSD showing the relationship between 5 songs using tags from the top 1024 subset. Only 50% of the assigned tags are shown for clarity. Node color indicates tag degree. Note that these similar songs share some of their tags.



(b) Subgraph of MTAT showing the relationship between 4 songs using all (top 188) tags. Node color indicates tag degree. Note that similar to MSD, these similar songs share many of their tags.

Figure 4.2: Graph representation of MSD and MTAT. Note how for both datasets tags can be shared between similar songs.

4.2. Experimental Design

In this section, we discuss the experimental design. We begin by briefly discussing the model selection and their respective hyperparameters. Then, we discuss how these hyperparameters are selected. Third, we discuss the metrics used to evaluate the performance of the models from various angles. An overview of the full experimental design, including selected hyperparameters, can be found in Appendix A.

4.2.1. Model and Hyperparameter selection

For our experiments we use the two models that implement the semi auto-tagging framework as outlined in Section 3.2: the content-aware matrix factorization model by Liang et al. and the Relational Graph Convolutional Network (R-GCN) model by Schlichtkrull et al. Additionally, we implement a simple Multi-Layer Perceptron (MLP) baseline which represents a simple auto-tagger without contextual input. We will now briefly discuss parameter selection for each model. We select hyperparameters using scikit-optimize[3], a Gaussian process-based optimization framework. To simplify the search of hyperparameters, for all models, we run hyperparameter optimization on the first cross-validation fold only to maximize the validation performance based on ranking accuracy. We then re-use these parameters for the remaining folds. The hyperparameter search ranges can be found in Table 4.3.

Model	**# Hidden**	$\mathbf{D_{hidden}}$	**Learning rate***	**L2 coefficient***
MLP	1, 2, 3	32, 64, 128	[1E-6, 1.0]	[1E-8, 1.0]
MF	-	32, 64, 128	-	[1E-7, 1E7]
RGCN	1, 2, 3	32, 64, 128	[1E-7, 1.0]	[1E-7, 1.0]
	$\mathbf{p_{dropout}}$	λ*	α*	**# Iterations**
MLP	[0.0, 0.5]	-	-	-
MF	-	[1E-7, 1E7]	[1E-7, 1E7]	[5, 20]
RGCN	[0.0, 0.5]	-	-	-

Table 4.3: Hyperparameter optimization ranges for the model selection. Brackets indicate a continuous range of values is considered. *The hyperparameter search grid is sampled from a log-uniform grid within the specified range.

Baseline

The baseline is a simple MLP with as input one of the content features mentioned in the previous section. We use ReLU as the activation function and use L2 and dropout [71] regularization with probability $p_{dropout}$. We use a fixed mini-batch size of 256. Additionally we use early stopping if the validation performance does not improve by 10^{-6} within 50 iterations, a threshold that we found empirically. Additionally, we run two baseline "dummy" classifiers from sklearn[4] to predict based on dataset *prior* (empirical class distribution) and *uniform* (uniformly random prediction) to assess if models learn.

Matrix Factorization

For the content-aware matrix factorization model, we use an Alternating Least Squares (ALS) optimizer to solve the equations listed in Section 3.2. We treat the number of solving iterations and the λ and α parameters as part of the hyperparameter optimization procedure.

RGCN

For the R-GCN we use ReLU as the activation function of the convolutional layers. Similar to the MLP, we make use of L2 and dropout regularization with probability $p_{dropout}$. We do not use mini-batches but update using full-batch gradient descent. We stop training if the validation performance does not improve by 10^{-6} within 25 iterations, which we found empirically.

4.2.2. Loss function

Due to the importance of ranking in our evaluation scheme, we opt to use Bayesian Personalized Ranking (BPR)[67] as our loss function for the deep-learning based models, the MLP and R-GCN. The objective

[3]scikit-optimize,
https://scikit-optimize.github.io/stable/
[4]sklearn.dummy.DummyClassifier,
https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html

of BPR is scoring pair of items (tags) i, j where i is preferred over j. For each tag i in the ground truth, we use negative sampling to obtain a tag j that does not appear in the ground truth, effectively training the model to rank the ground truth above the untagged entries. BPR for song u between positive tag i and negative tag j is defined as:

$$\mathcal{L}_{bpr} = -\sigma(\hat{x}_{uij}) \tag{4.1}$$

Where σ is the logistic sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$

and $\hat{x}_{uij}$ is a function specifying the relationship between tag i and j which we define as

$$\hat{x}_{uij} = x_i - x_j \tag{4.3}$$

Implying that tag i being higher ranked than tag j contributes to minimizing the loss.

4.2.3. Evaluation

We evaluate the models on each dataset subset using 10-fold cross-validation. For each subset, we consider different amounts of contextual input seed tags $n_{tag} = \{0, 1, 2, 5\}$. First, we split the pool of songs in the subset into approximately equally sized subsets for each value of n_{tag}. For each song within this subset, we hold out a corresponding number of tags as seeds and use the remainder as test entries. If the total number of tags in a set is smaller than the number of seeds, we filter those entries[5]. The distributions of the resulting datasets for the first fold can be found in respectively Figure 5.2a and Figure 5.2a of Appendix F.

For each fold, we make use of three metrics to gain insight into the behavior of the different model configurations. They are used to measure three different properties: classification accuracy, ranking accuracy, and embedding similarity to highlight different aspects of the model configurations.

Classification Accuracy

To measure the classification accuracy of the models, we consider the Receiver Operating Characteristic Area Under Curve (ROC-AUC). Due to the definition of the metrics, we only consider the ROC-AUC in a scenario where no seed input tags are given. The ROC plots the True Positive Rate (TPR) versus the False Positive Rate (FPR) at various classification thresholds. TPR and FPR are defined as:

$$\text{True Positive Rate (TPR)} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4.4}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}} \tag{4.5}$$

The area under this curve is the AUC, a metric for prediction accuracy independent of the chosen classification threshold. It is typically used for link prediction tasks and is used frequently in auto-tagging literature. The AUC has a value of 0.5 if models recommend randomly, e.g. they are unable to discriminate between classes. The closer the value is to 1, the better the model performs. To highlight classification performance at different levels, we distinguish between *AUC-micro* (element-wise), *AUC-macro* (average per tag, column-level) and *AUC-samples* (average per song, row-level) as implemented[6] in Scikit-learn[62].

Ranking Accuracy

To examine how well models perform in terms of prediction relevance we consider the Normalized Discounted Cumulative Gain (NDCG) [47]. Which is defined as:

$$\text{NDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p} \tag{4.6}$$

[5] https://github.com/jochem725/semi-auto-tagging/blob/master/autotag_graph/data/seed_split.py
[6] sklearn.metrics.roc_auc_score,
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

Where

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2 (i+1)} \tag{4.7}$$

$$\mathrm{IDCG}_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i} - 1}{\log_2 (i+1)} \tag{4.8}$$

Where $|REL_p|$ is the list of tags ordered by relevance. It expresses how accurate the predicted ranking of tags is per song based on the order of prediction. In our experiments, we use cutoff $p = 10$, which we think is a realistic amount of tags to be recommended to a user of a tag recommender as it is representative of a real-world graphical user interface. Additionally, the cutoff is smaller than our smallest dataset subset, if we take a maximum number of 5 seed tags into account. This metric gives insight into whether giving contextual input tags is beneficial to predicting more relevant tags when considering the final list of predictions as output to humans. We used the same cutoff as part of the validation metric in the model training loop.

Embedding Similarity
To gain an understanding of how well models can learn which tags have a similar context, we use a heuristic to compare the learned embeddings with the textual representation of the tag. This heuristic is based on the Levenshtein distance, which measures the difference in characters between two strings. Under the assumption that syntactically similar tags can have a similar meaning, first, we compute a similarity matrix of Levenshtein distances between tags using the TheFuzz[7] library. By doing so we obtain a matrix of scores in $[0, 1]$ that indicate the similarity of tags based on text. Second, we compute a cosine-similarity matrix of the embedding distances. Using the cosine-similarity matrix as input and the Levenshtein matrix as ground truth, we then compute the NDCG with cutoff $p = 10$. By doing so we get an indication of how well the model performs in embedding similar textural descriptions. We do note that this heuristic is suboptimal in several ways. Primarily, while tags may appear to be similar textually, they may not be similar by meaning. For example, *female voice* and *male voice* differ substantially in meaning when tagging is considered (different kinds of voice). However, we do believe that in many cases it is still useful as a proxy given that there exist many tags that are similar in both text and meaning.

4.3. Implementation Details

The code that was used to run the experiments can be found on Github[8] and was written using Python 3.7. We used Deep Graph Library (DGL)[83][9] to implement the model based on the GCN. Both the GCN and MLP were implemented using Pytorch [61]. Part of the code for the MF model was obtained from the Attribute-Aware Recommender ModelS (AARMS) library by Jaehun Kim [10]. A full list of dependencies and their versions can be found on GitHub.

4.3.1. Hardware

Experiments were performed on the TUDelft Intelligent Systems HPC cluster. The HPC consists of multiple nodes with a varying range of CPU and GPU configurations. For fairness, the experiments for the MLP and MF models were executed on the same GPU series, an NVIDIA Tesla P100 with 16GB of memory.

[7] TheFuzz,
https://github.com/seatgeek/thefuzz
[8] https://github.com/jochem725/semi-auto-tagging
[9] DGL, https://github.com/dmlc/dgl
[10] AARMS, https://github.com/eldrin/aarms

4.4. Research Questions

In this section, we discuss how our experimental design aims to answer the research questions listed in Chapter 1.

4.4.1. RQ1: Effectiveness of a Semi Auto-tagging system

Our first research question - *How can we build a system that suggests additional tags based on expert input?* - is addressed by the introduction of the semi auto-tagger and the two models that we propose to implement it. However, the quality of the system defines its usefulness in the context of auto-tagging.

Subquestion 1.1

The first subquestion for RQ 1 is - *Given song-tag relationships combined with audio features, can a semi auto-tagger predict more relevant tags compared to a feature-only auto-tagger?*. As shown in Section 2.2, auto-tagging is based primarily on audio feature input. Models that consider relationships, such as MF models typically operate on just relationships between entities. For a semi auto-tagging system to work well for audio tag prediction, audio must be incorporated into the model in a way that allows the model to benefit from both relationships and the audio feature. To address this question two properties are important:

- Can the semi auto-tagging model make use of the input feature information?
- Does the addition of contextual information result in more relevant tag predictions?

We address the first property by evaluating our models using two sets of input features: one meaningful feature derived from the audio, and one meaningless random noise feature. To address the second property, we compare the semi auto-tagging models with a baseline, the MLP-based auto-tagger which can only make use of input features. A comparison using an evaluation metric that measures relevance then shows if a semi auto-tagger with feature input can indeed predict more relevant tags. As mentioned in Chapter 1, we hypothesize that combining song-tag relationships with meaningful audio features allows us to obtain more relevant tag predictions compared to the baseline.

Subquestion 1.2

The second subquestion for RQ 1 is - *For previously unseen songs, can a semi auto-tagging model predict relevant tags?*. Since the goal of auto-tagging is alleviating the cold-start problem for unseen songs, it is vital that the semi auto-tagger also works for new songs. This question is partially addressed by the model selection and partially by the evaluation metrics. Our selection of models was chosen carefully, to allow previously unseen input. The model by Liang et al. addresses this by allowing out-of-matrix prediction. The model by Schlichtkrull et al. can operate on arbitrary graphs. The evaluation subsets of our datasets were prepared such that they only contain previously unseen songs. We hypothesize that a semi auto-tagging model can successfully predict tags for previously unseen songs, as is shown by the models on which our implementation is based.

4.4.2. RQ2: Effective use of the vocabulary

To answer the second research question - *Does a noisy tag vocabulary contain useful information for (auto-)tagging?* - We make use of larger vocabulary sizes than are traditionally used by auto-tagging works. By using bigger vocabularies, we make it possible to learn more possible relationships between tags. We then look at two properties of models. First, we look at ranking accuracy in the long tail of each dataset subset. We define the long-tail as the 50% least frequent tags in the set. By measuring this we hope to show that our choice of models can recommend from the long tail better when compared to the baseline MLP auto-tagger. We hypothesize that for the semi auto-tagging models, the classification accuracy will be higher for tags in the long tail of our vocabulary as compared to the MLP baseline. Second, we compare tag latent factors and embeddings learned by the models to see if they can capture similar concepts. We do this by comparing the embeddings for tags with distance-based metrics that define their similarity in the graph. If similar topics have similar embeddings, the model has successfully identified similar topics and has therefore learned implicit relationships between tags. We compare latent factors for the MF model and learned embeddings for the graph and baseline models. Due to the nature of the model selection, we hypothesize that the semi auto-tagging models can capture similar tag string representations better than the baseline, especially if the size of the input dataset grows.

5
Results

In this chapter, we discuss the results obtained by running the experiments outlined in the experimental design. We look at the results from three perspectives. First, we discuss the classification accuracy by focusing on the ROC-AUC. Second, we discuss the ranking accuracy by looking at the NDCG. Finally, we look at the quality of the learned embeddings. We conclude with a brief discussion to show how the experiment results contribute to answering the research questions.

5.1. Classification Accuracy

The results from a classification accuracy perspective can be found in Figure 5.1. We split out classification accuracy[1] (ROC-AUC) in three different levels (AUC-micro, AUC-macro, and AUC-samples). Each level highlights a different aspect of the models, respectively considering individual tag assignments (element-wise), the average AUC per tag (column-level), and the average AUC per song (row-level). Note that the accuracy metrics are only computed for cases where no seed tags are supplied to the models which means they simulate the scenario where the models run in the absence of contextual information in the form of tags.

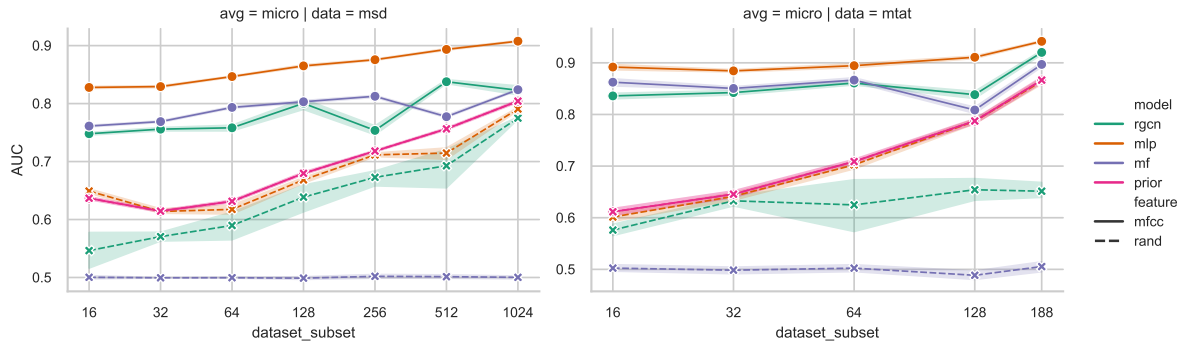
5.1.1. AUC-micro

Looking at the element-wise classification metric in Figure 5.1a we see that the MLP baseline outperforms the other models for all variations of the experiments when a meaningful input feature (MFCC) is given. Notably, for both datasets, if the number of tags in the subset increases, the classification score increases as well even when a random feature is used. We believe this can be explained by the distribution of the dataset. As the size of the vocabulary grows, tags are generally assigned more sparsely if they are infrequent. This means there are more zeros in the ground truth and prediction, leading to a higher AUC score as the number of true negatives goes up lowering the false positive rate. Note that with a random feature, none of the models can improve upon the prior baseline because with a random feature the prediction is effectively random. The MF model, however, does not perform with a random feature. The random feature in combination with no seeds given results in numerical instability which causes predictions to be random. Learned embeddings are close to zero and as a result, the output prediction scores (logits) are zero as well. Supplying a random feature, therefore, leads to uniformly random scores, which equals an AUC value of 0.5. For MTAT, supplying a random feature to the R-GCN consistently performs worse than the baseline while it does seem to give a more reasonable prediction than MF with the random feature. We were unable to identify the root cause of this, and do not observe this behavior when considering AUC-samples.

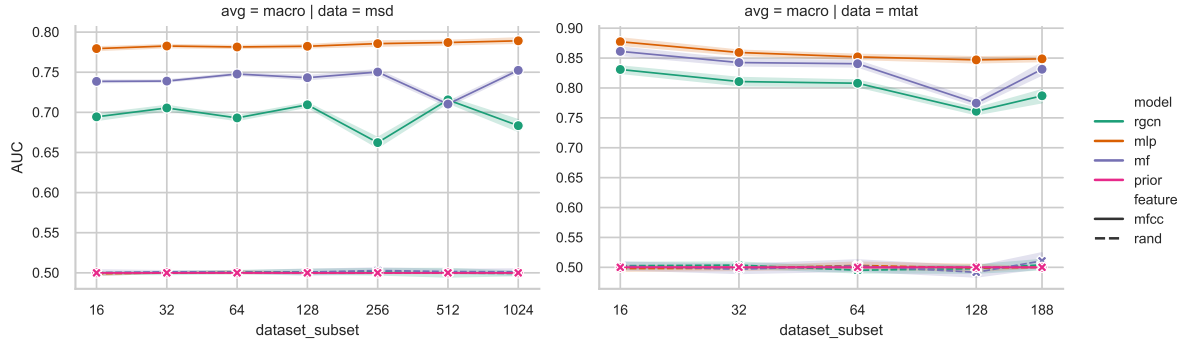
5.1.2. AUC-macro

If we consider the AUC by averaging at a tag level as shown in Figure 5.1b we find that once again the MLP baseline outperforms the other models. By averaging at the tag level, the effect of tag imbalance in the dataset is not considered. Each tag is treated equally important by the metric. We observe

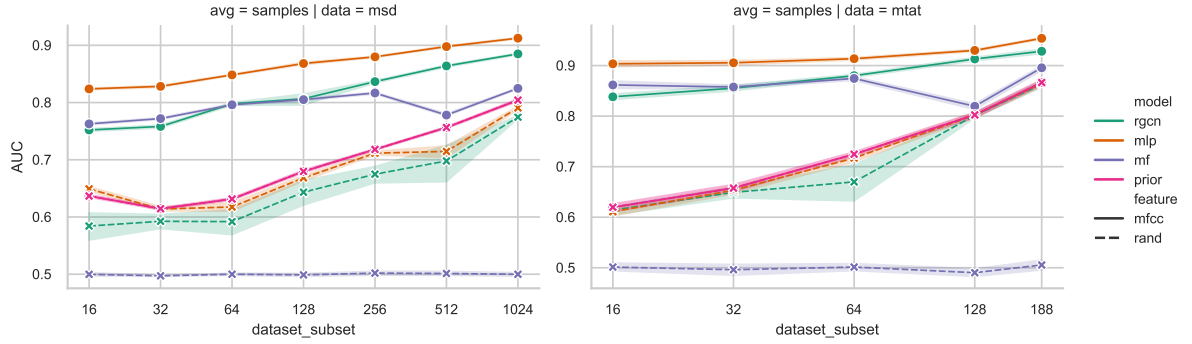
[1]The raw numerical values behind the accuracy figures can be found in Appendix B.



(a) AUC Micro level



(b) AUC Macro (tag level)



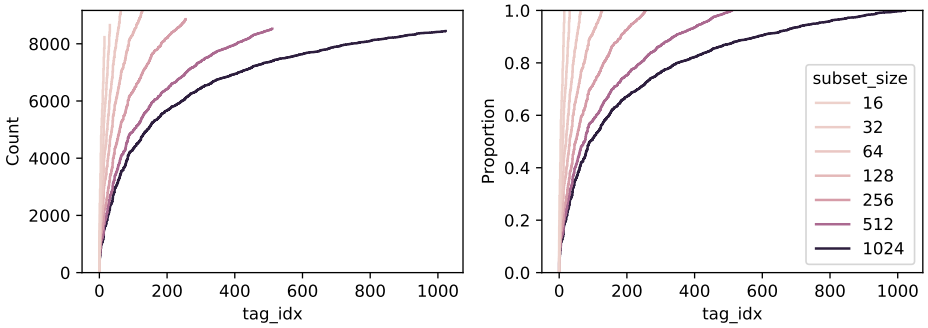
(c) AUC Samples (song level)

Figure 5.1: ROC-AUC metrics on various levels. On the horizontal axis, the size of the respective dataset subset is given, incrementally. The vertical axis shows the output score. Metrics are computed for the case where no contextual input (seeds) is given to the model. Note: all models consistently reach a higher accuracy when a meaningful feature (MFCC) is given as compared to a random feature. Models can make use of the content feature if there is no access to contextual input (via an input seed tag). In general, there is a clear distinction between samples (aggregated at the song level) and macro (aggregated at the tag level).

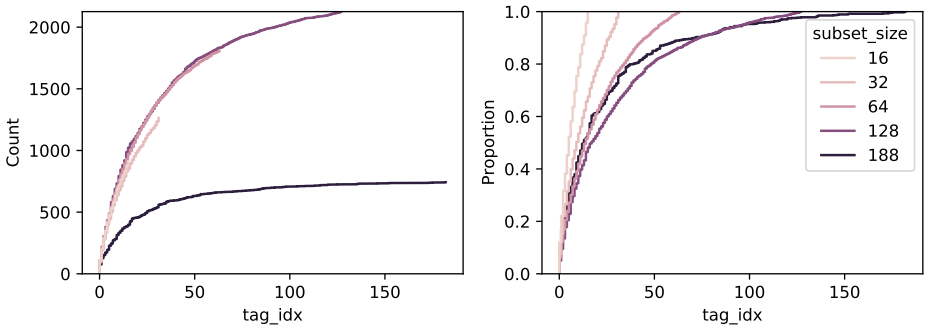
that if the vocabulary size increases, for MSD we see a small, upward trend in performance. For MF we observe a small downward trend. These trends are however small; indicating that using a larger vocabulary does not impact the performance of the models in this case. If we consider the models we do however see a large impact. Notably, the R-GCN performs substantially worse than other models. As expected, supplying a random feature leads to an AUC-macro of 0.5 for all models, as a random feature will lead to uniformly random tag assignments because of how the metric is averaged.

5.1.3. AUC-samples

Looking at song level performance in Figure 5.1c, we observe roughly the same behavior as the micro level. Notably, for the MTAT dataset, a random feature given to the R-GCN does not lead to degraded performance when compared to the prior baseline. Figure 5.2 shows the tag distribution of the first fold of each dataset subset, for the case where zero seed tags are given. The horizontal axis shows the index of each tag ordered by frequency, where the tag with index 0 is the most frequent. On the vertical axis, we find the count and fraction of tag occurrence. For subset sizes 512 and 128 of respectively MSD and MTAT we see data points that do not follow the general trend observed when increasing the dataset subset size. The dataset distribution for these subset sizes however does not seem to be irregular compared to the other subset sizes. In the case of MTAT we see that there are fewer tags in the dataset with subset size 188, this can be attributed to the procedure that was used to create the dataset splits as described in the experimental setup.



(a) MSD



(b) MTAT

Figure 5.2: Distribution of tag assignments for the first fold of each dataset subset when 0 seeds are given. Other folds are distributed similarly. *Count* refers to the cumulative number of tags with the given *tag_idx*. Similarly, *Proportion* refers to the cumulative percentage.

5.2. Ranking Accuracy

The results[2] from a ranking accuracy perspective can be found in Figure 5.3[3]. We will now discuss the results from a ranking perspective by first discussing observations when ranking the full (subset) vocabulary. Second, we will discuss observations by looking only at recommendation performance using tags from the long-tail (last 50%) of each subset vocabulary.

5.2.1. NDCG (full)

Ranking accuracy for each subset, varying the number of seed tags given can be found in Figure 5.3a. When focusing on the scenario where zero seeds are given, we observe similar performance compared to the accuracy results for both MSD and MTAT. Specifically, when providing a meaningful feature (the MFCC) to the model, we see an improved ranking score for all subsets. However, if a random feature is given, two things can be observed. First, all models have similar performance compared to the prior baseline, giving evidence that the model simply learns to assign tags in a way that matches the distribution of the dataset. Second, when comparing the prior and uniform baselines, there is an effect of the tag distribution on the results.

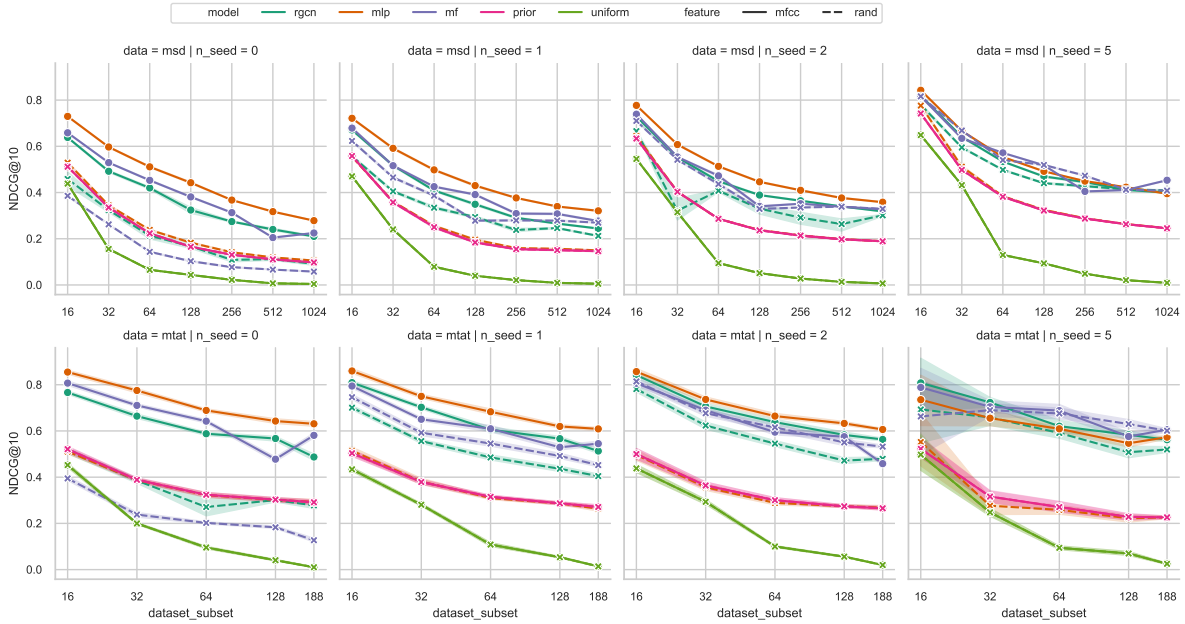
When seeds are given we see a clear change in ranking score. Both semi auto-tagger models, MF and the R-GCN greatly benefit from the addition of a single seed tag, even when a random feature is given. Their ranking score is now far above that of the prior, indicating the models are no longer outputting scores based on the distribution of the dataset and that they can predict relevant tags, even in the absence of the MFCC. We observe this effect for both MSD and MTAT. As more seeds are given to models with a random feature as input, the score gap compared to models that have been given the MFCC becomes smaller. For MTAT in the case where 5 seeds are given as input we observe high variance in the score. This can be attributed to a decrease in the nubmre of songs per fold if more seed tags are held out, leading to high variance in the results. A similar phenomenon occurs in MSD, but since there are many more songs available in this dataset, there is no visible effect in variance.

5.2.2. NDCG (50% tail)

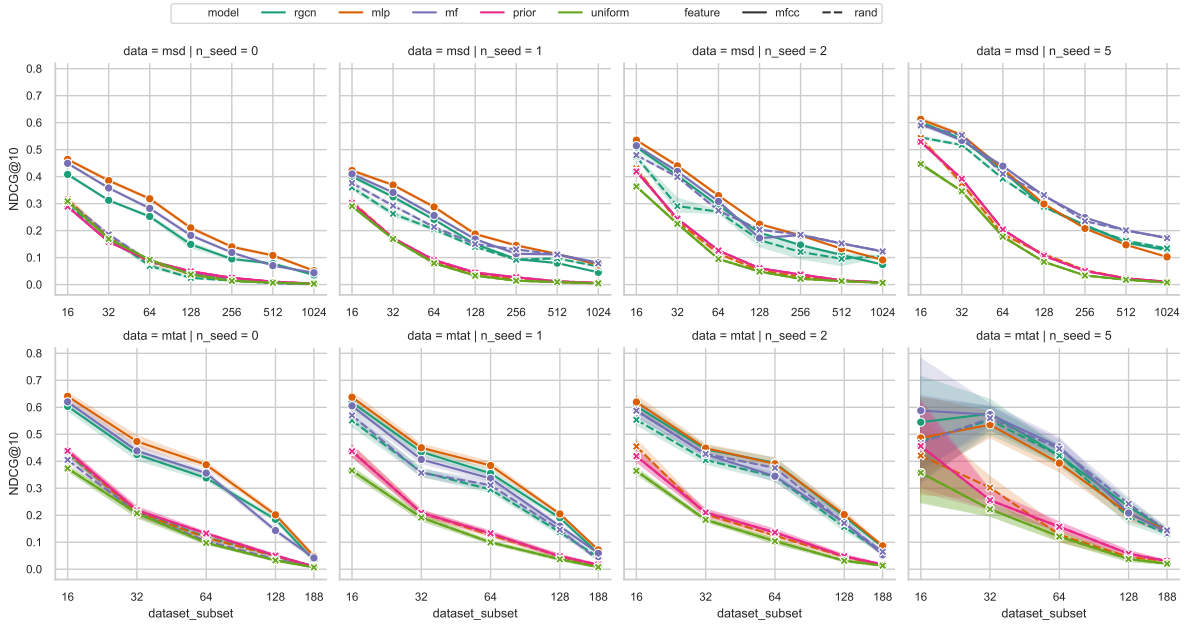
To dive deeper into the behavior of models when only tail tags are considered, ranking metrics have been computed excluding the 50% most frequently occurring tags (the head). These scores can be found in Figure 5.3b. Note that models have not been retrained and given the same seeds considered in the full dataset ranking scores. Firstly, we still observe that giving a single seed improves the ranking score for semi autotagger models when only a random feature is considered. This provides evidence that it is beneficial for long-tail tag prediction to supply a seed tag to models. Note that due to the distribution of the tags in both MSD and MTAT, it is likely that a seed tag is a popular tag. As hypothesized in Figure 1.1 such a tag can indicate the model to consider less popular seed tags, which we observe by the increase in ranking score.

We see that in the tail the scores for models are much closer to the MLP baseline for both the MFCC and random feature. But when more seeds are given, the semi auto-tagger models start outperforming the MLP indicating that they have a better performance on the tail overall. Notably, this might indicate that when a larger vocabulary with more seeds is used, these models may outperform the MLP even more.

[2]The numerical data behind the ranking accuracy figures can be found in Appendix C.
[3]Larger versions of these figures can be found in Appendix F



(a) NDCG score, including all tags in the subset vocabulary.



(b) NDCG score, top 50% of the most frequent tags are omitted, leaving only tags from the long-tail.

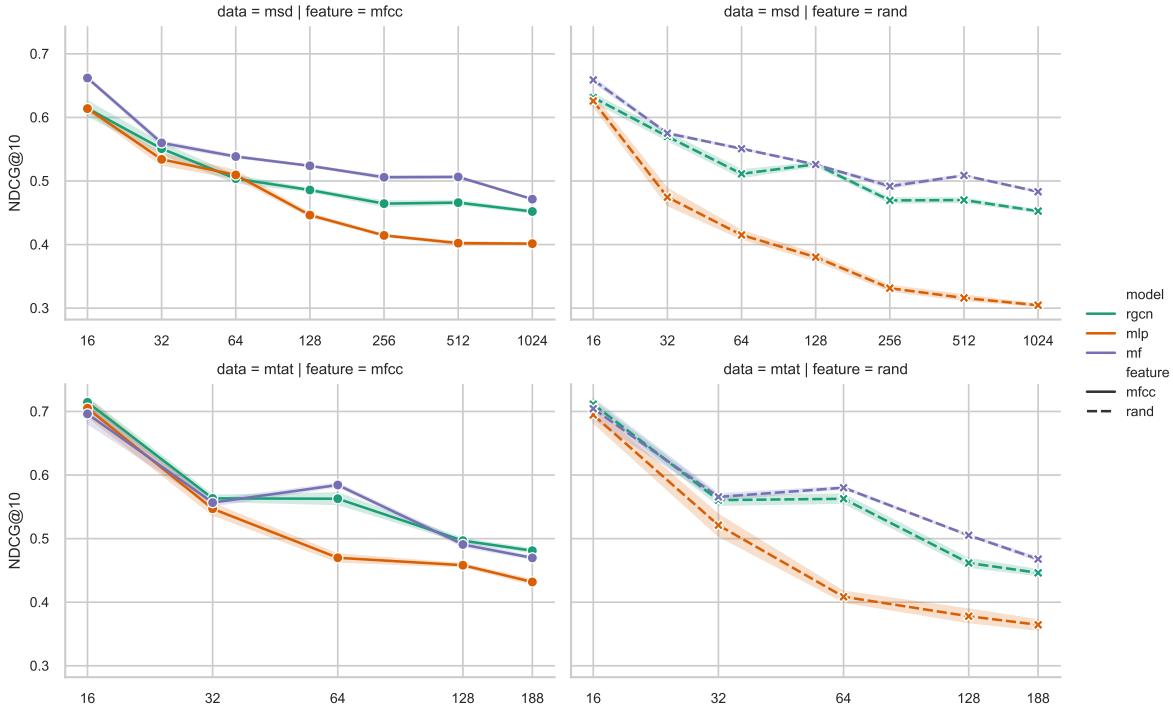
Figure 5.3: NDCG scores for both the full vocabulary and the long-tail.

5.3. Embedding Similarity

Figure 5.4a shows the output[4] of the Levenshtein similarity metric[5]. Note that the Levenshtein distance is used as the heuristic ground truth here.

First looking at the case where a meaningful input feature, the MFCC, is given we observe that the embeddings of semi auto-tagger models, on average, have a higher score. This means that tags that are similar considering their text, also learn similar embeddings in the network models. This indicates that the network models are better able to group tags that may have a similar meaning based on their string representation, something we can observe when we visualize in Figure 5.5. The figure shows how similar embeddings are compared to the reference tag *female vocalists* when considering the Levenshtein distance (horizontal axis) and the cosine similarity of the embeddings (vertical axis). Note that for text with a similar string representation such as *female vocals*, the cosine similarity between the embeddings for *female vocalists* and *female vocals* is also high for semi auto-tagging models, whereas for the MLP this is not the case.

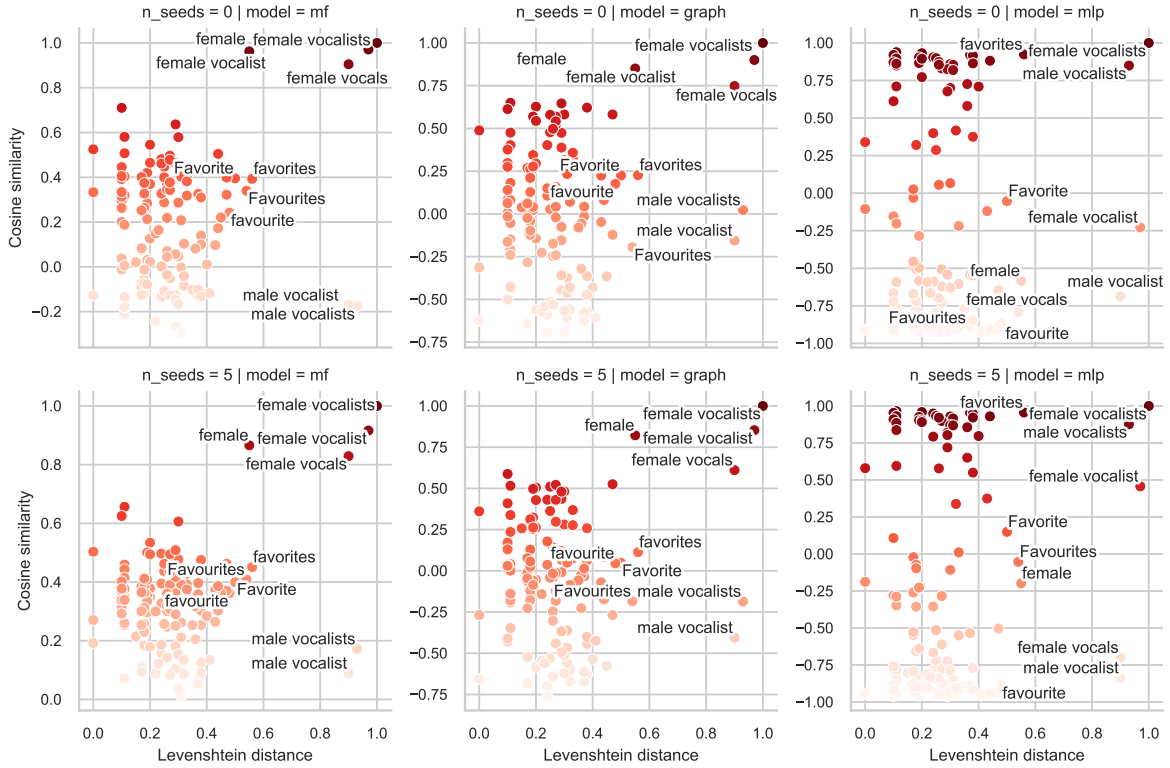
Second, if we consider only the random feature we see that the score of the MLP is much lower compared to the semi auto-tagging models as it starts learning random embeddings to approximate the distribution of the dataset. Looking at the network models, we see that even with a random feature the score remains high, meaning that the models benefit from the given seeds and can capture tags that have a similar string representation.



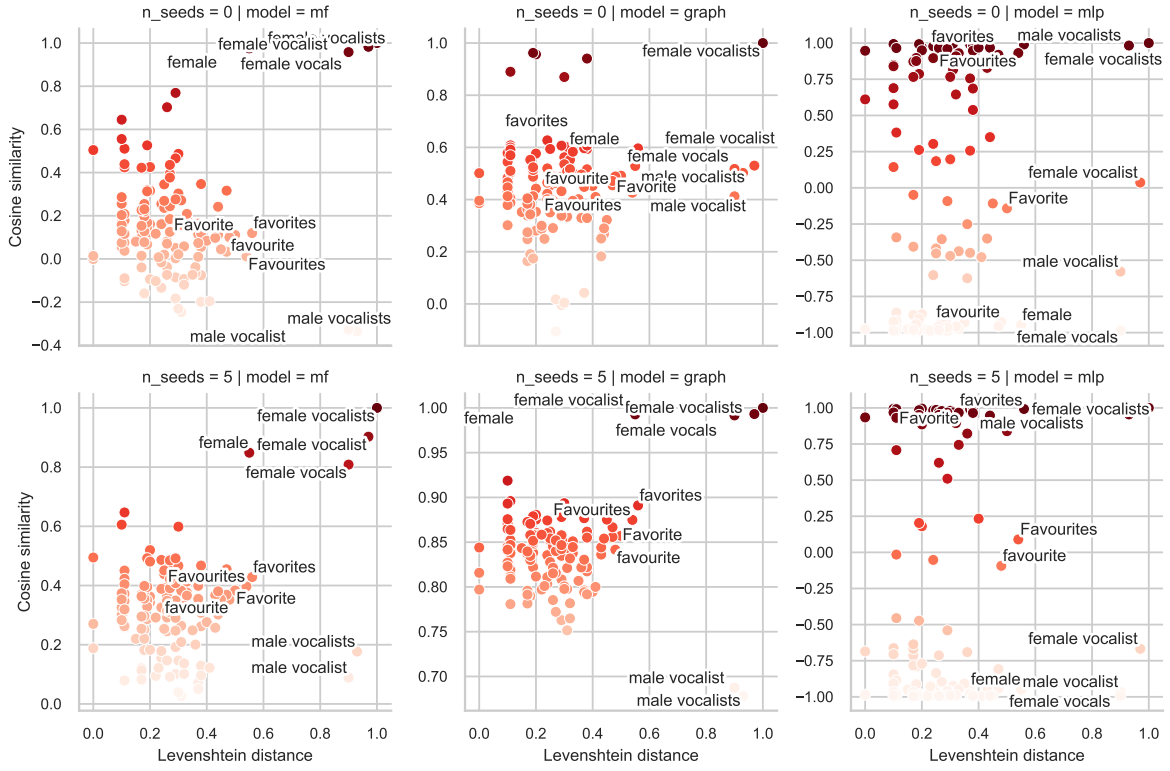
(a) NDCG score using a tag-tag similarity matrix as ground truth input (Levenshtein distances) and the cosine similarity between tag embeddings from the model as prediction input. The output is a heuristic that approximates how well a model can distinguish tags. The score is shown for models that have been given 5 seed tags as input.

Figure 5.4: Embeding similarity

[4]The numerical data composing the figures can be found in Appendix D.
[5]Metrics have been computed for the case where 5 seeds are given to the model.



(a) Embedding comparison when an MFCC feature is given. Note that the network-based models are better at grouping tags that are similar by string representation.



(b) Embedding comparison when a random feature is given. Note that even in the absence of a meaningful input feature the network-based models can still separate tags based on similar string representation, while the MLP fails to do so.

Figure 5.5: Scatterplot of embedding cosine similarity versus Levenshtein similarity for the tag *female vocalists* of the MSD dataset with subset size 128. We compare the case where 0 and 5 tags are given as input to the model. Note: the metric was computed with a cutoff of 10 tags, meaning that only the 10 tags with the highest cosine similarity between embeddings are considered in the score.

5.4. Discussion and limitations

In this section, we discuss high-over findings and follow by discussing the limitations of our work.

5.4.1. Discussion

Results from a model accuracy perspective (ROC-AUC) show that when a meaningful input feature is given all models consistently reach a higher classification accuracy when compared to a random feature. This shows that the MF and R-GCN can learn from a meaningful feature in the absence of contextual input data (a network of songs and tags). This indicates that models can provide a reasonable prediction if no context is available for a particular song. The MF and R-GCN are however not able to reach the same performance as the MLP. Notably, if the dataset subset size increases the accuracy score does as well for the micro and macro levels due to the effect the tag sparseness has on the evaluation metric.

Considering results from a ranking perspective (NDCG), similarly, as for the accuracy results we observe there is merit in providing a meaningful input feature. If meaningful content information in the form of input seed tags is given, we see that the gap between the MFCC and the random feature becomes smaller while the ranking score increases. A similar effect can still be observed when looking at the results calculated on the long-tail (50% least frequent tags) indicating that models can predict from the long-tail. Note that for MTAT the variance between folds appears to increase as more seed tags are given. This can again be attributed to the way the dataset was split. Holding out seed tags causes fewer tags to be available for the ground truth, especially on small subset sizes, eliminating a large number of songs from the set (because no ground truth remains for those songs). Distributions shown in Figure F.2 show that when 2 and 5 seeds are given, the number of tags available in the ground truth is substantially lower than when 0 and 1 seeds are given leading to high variance in the results. Calculating metrics on the tail amplifies this even further because even fewer tags from the ground truth can effectively be used to calculate the metric.

Looking at the embedding similarities, we find that semi auto-tagging models are better able to learn similar embeddings for tags that have a similar text string representation. Notably, this effect also occurs when a random feature is given to the model indicating that the addition of context information is very beneficial to learning these relationships.

5.4.2. Limitations

It is however important to note that our work is limited in several ways. Firstly, the input datasets, being noisy, contain many incorrect tag assignments. By listening to the songs we hear that in many cases the ground truth is either lacking or wrong. If we take as an example Table 5.1, The *R&B/Soul* song *Rosie* by Joan Armatrading of models trained with dataset subset size 128, the tags selected in the ground truth contain *rock* and *soft-rock*, but when listening to the song there are no rock elements present in the song. Additionally, this song can be seen as a *pop* song, but it is not part of the ground truth. Semi auto-tagger models seem to suggest better tags subjectively in ranking, but they are not present in the ground truth. If the *rnb*, *female vocalist*, and *pop* tags would have been present in the ground truth the NDCG score of the MF model would have exceeded the MLP.

ground truth	input seed	mf (0.458)	graph (0.410)	mlp (0.411)
rock	Love	pop	**soul**	pop
female vocalists	80s	**female vocalists**	rnb	**female vocalists**
singer-songwriter	70s	**soul**	*spanish*	**rock**
soul	female	**oldies**	**oldies**	dance
oldies	reggae	**rock**	pop	00s
happy		classic rock	classic rock	*spanish*
latin		favorites	*60s*	female vocalist
smooth		female vocalist	***latin***	***latin***
soft rock		rnb	country	**singer-songwriter**
		dance	male vocalists	favorites

Table 5.1: **Rosie - Joan Armatrading** For each model the NDCG score with cutoff 10 is given.
Spotify: https://open.spotify.com/track/4yMdCBVURkXYYNtY53H1FY.

Additional spot-check examples can be found in Appendix E which show that the semi auto-taggers are often not wrong, but ground truth is lacking. The quality of the ground truth is as expected and is a side effect of the way the dataset tags are collected (either via crowdsourcing or a tagging game). A less noisy ground truth will not only enable models to learn better but will also enable evaluation metrics to judge models more fairly. Alternatively, the ground truth of existing datasets could be filtered or validated before using them to train models; but this may be a resource-intensive task given the scale of both datasets. Second, as mentioned by Law et al. human evaluation is essential for measuring the true performance of music taggers[56]. An evaluation performed by skilled human experts, or a validation of the ground truth by experts can be used to better assess the usefulness and accuracy of the system in a real-world scenario. Additionally, since the output subsets are large there can be many tags that have a similar meaning, but simply don't appear in the ground truth which skews the evaluation metrics. Merging or expanding the ground truth based on similar tags might further improve the fairness of evaluation metrics.

Our selection of models made it challenging to consider a larger variation in vocabulary size and number of seeds assigned due to computational resource constraints such as (GPU) memory, but the analysis of the long-tail ranking metrics gives evidence that a larger vocabulary combined with more input seed tags may be crucial to achieving better tagging results. Adapting other matrix factorization or deep learning models to fit in the semi auto-tagger framework might unlock the usage of even bigger vocabularies further increasing the tagging diversity. Additionally, our choice of input feature, the MFCC may have been limiting as previous work as discussed in Section 2.2.4 as shown there is merit in using features obtained from deep-learning models. We chose not to make use of those features to not complicate the research any further as we would have had to train these models ourselves using our dataset for fairness.

6

Conclusion

In this chapter, we briefly summarize the motivation and hypothesis for our research and answer the research questions. We finish with a short conclusion and discuss future work.

Contextual information can be obtained from various sources, both human- and machine-driven. Human expert tagging is accurate but time-consuming. Crowd-sourced social tagging allows collecting tags at scale, but the tags are noisy, so they may not be correct. Automatic tagging (auto-tagging) is a way of obtaining tags using a machine. The goal of such an algorithm is to predict tags directly from song audio, without human interference at scale, to solve the problem of sparsely tagged songs in a dataset. Building an auto-tagger is however difficult because the size of the vocabulary of popular tagging datasets such as MagnaTagATune (MTAT) or Million Song Dataset (MSD) varies and often has a long-tail distribution.

In this work, we proposed an adaptation to the classic auto-tagging problem, called *semi auto-tagging*. In this framework, we allow tags as contextual input to the auto-tagger, in addition to an audio content feature. This combination allows for stags to be predicted based on audio (content) and the position of the song in the vocabulary of tags (context). Such context could be provided by a human expert, which could potentially improve their tagging efficiency, therefore reducing their cost.

We hypothesize that using this approach, comparable or better performance can be achieved when compared to a traditional content-based auto-tagging baseline. To validate this hypothesis, we implemented two candidate models. The Relational Graph Convolutional Network (R-GCN), a deep-learning based graph model [69] and a content-aware matrix factorization model[81]. Both models allow combining an input feature, with contextual information in the form of external input tags. We compared the performance of both models by looking at both ranking accuracy and classification accuracy for varying vocabulary sizes of MTAT and MSD.

6.1. Contribution

We find that it is possible to build a system that can suggest additional tags based on expert input in the form of seed tags given to a model as contextual information combined with an audio content feature. Our selection of models however does not exceed a simple MLP baseline. Models can predict tags for previously unseen songs, allowing them to alleviate the cold-start problem. Models created within the semi auto-tagger framework seem to be able to better learn relationships between tags that have a similar text string representation. Based on our results we believe there is evidence that a noisy tag vocabulary contains useful information for tagging, however, the ground truth for models needs to be improved to better assess this. We will now proceed by answering the research questions stated in chapter 1 in more detail.

Research Question 1

RQ1 - How can we build a system that suggests additional tags using contextual input?

SQ1.1 - Given contextual input combined with audio features, can a semi auto-tagger predict more relevant tags compared to a content-only auto-tagger?

To answer this question two properties are important, as outlined in Section 4.4. First, we address the possibility of semi auto-tagging models to make use of input content feature information. We find that for our selection of models, it is possible to make use of input feature information. By comparison with a random feature, we find that if a meaningful input feature (the MFCC) is given, the model score improves substantially as can be found in both accuracy and ranking metrics in respectively Figure 5.1 and Figure 5.3. Where a random feature results in models predicting randomly from the dataset, not exceeding a prior classifier supplying a meaningful feature results in an improved score. This indicates models can make use of the input feature for learning.

Second, we look at the addition of contextual information by supplying seed tags to the model and assess if they result in more relevant tag predictions. By comparison with an MLP that is unable to make use of contextual information, we find that if more contextual information is given to the semi auto-tagging models, model scores improve towards the baseline. Our results give evidence that if more seed tags are supplied, models may start to exceed the MLP baseline, something which occurs in the long-tail as seen in Figure 5.3b.

Additionally, a comparison of the embeddings in Figure 5.4 using a heuristic based on the Levenshtein distance show that the semi auto-tagging models are better able to cluster together tags with a similar string text representation as compared to the baseline.

SQ1.2 - For previously unseen songs, can a semi auto-tagging model predict relevant tags?

To be able to alleviate the cold start problem for unseen songs, a semi auto-tagger should work for previously unseen songs. Our selection of models, the Content-Aware Matrix Factorization model by Liang et a. [58] and the Relational Graph Convolutional Network (R-GCN) model by Schlichtkrull et al. [69] are both capable of allowing "out-of-graph" prediction, in this case meaning they can predict for song nodes that were not part of the training data. The R-GCN does this by extending a node to the training graph and making a prediction for that node. Similarly, the MF model predicts by rerunning part of the optimization step of the model during prediction time. Additionally, our evaluation subsets only contain previously unseen songs. We find that all models can predict relevant tags for previously unseen songs as shown in Figure 5.3 when a meaningful input feature is given. Therefore a semi-autotagger can still alleviate the cold-start problem for new songs, which is one of the main goals of a traditional auto-tagger.

Research Question 2

RQ2 - Does a noisy tag vocabulary contain useful information for (auto-)tagging?

By training and evaluating our models on dataset subsets that are bigger than the sizes used by traditional auto-tagging approaches we find that both the baseline and semi auto-tagging models can make use of larger vocabularies. If we look at the performance in the long-tail, the top 50% least frequent tag in each subset, we see that if sufficient contextual input seed tags are given, semi auto-tagger models can exceed the performance of the MLP baseline, indicating predictions from the long tail benefit from contextual input. By spot-checking prediction output, however, we find that our ranking metric may not be optimal to answer this question. We find that the ground truth is often incomplete or lacking, leading to lower scores on the semi auto-tagger models when they predict correctly (from the long tail). As tags become less frequent, they are more likely to be unavailable in the ground truth.

By comparing embeddings in Figure 5.4 using a heuristic based on the Levenshtein distance and cosine similarity between respectively tag string representations and tag embedding representations,

we confirmed that the semi auto-tagger models can better learn implicit relationships between tags compared to the MLP baseline. Figure 5.5 makes this visual by showing an example where tags with a high Levenshtein distance, also have a similar cosine similarity and are therefore clustered together. Semi auto-tagging models are better able to capture similar tags by string representation, even if the vocabulary size grows, giving evidence that a tag vocabulary such as MSD contains useful information for auto-tagging despite being noisy.

6.2. Future Work

The discoveries made during this research provide various points that can be used to drive future research. We will address future work by discussing three issues that impact our work. First, the quality of the ground truth which was shown to be lacking might have hurt the evaluation metrics. As described in Section 2.1.2, it is known within the MIR field that input datasets are often lacking reliable ground truth which is something that we have observed in Table 5.1 of Section 5.4.2. While we have shown that the semi auto-tagging models can learn relationships between similar string representations from the noisy vocabulary, it is difficult to access the quality of the recommended tags. Improving the quality of the existing ground truth, or collecting new, more clean datasets for music auto-tagging instead of relying on Million Song Dataset (MSD) and MagnaTagATune (MTAT) might lead to a more fair evaluation of auto-tagging systems. Another way to better access the performance of models is by using real human experts as part of the evaluation process. Previous research has indicated that it is essential for humans to be involved in determining the performance of auto-tagging models [56]. While in this work we have shown evidence that external tag input may be useful, we did not verify this with actual human experts.

Second, our selection of evaluation metrics and dataset split was suboptimal. The answer to research question 2 can be elaborated upon if ranking metrics were computed at a tag level, to better be able to compare scores between the popular tags and tags from the long-tail. Alternatively, we could have used a weighting scheme that penalizes more frequent tags to better access the performance in the tail. Figure 5.3 shows that if the dataset subset size increases and more tags are given, the semi auto-tagging models start to approximate the performance of the MLP baseline. Extending the study to even larger subset sizes and tag seed assignments might show that semi auto-taggers can exceed the performance of the baseline.

Finally, in this work, we have relied on a relatively simple MLP baseline and feature selection. A better baseline however would be one of the deep-learning-based models discussed in Section 2.2.4, as they have been evaluated on MSD and MTAT before. Similarly, there exist more models that fit in the semi auto-tagging framework, specifically those that scale well to larger datasets implemented using libraries such as DGL[83] to be more computationally efficient. Previous work as described in Section 2.2.2 has shown that features extracted using deep-learning models often exceed the performance of the MFCC. To make use of these features, however, the experimental design has to be adjusted to train these models on the same subsets used for the semi auto-tagging models.

Summarizing we believe the following three topics are relevant for future study: improving the ground truth used to evaluate models, adjusting the evaluation scheme to gain deeper insight into the performance of tags from the long-tail, and extending the study to consider models that are better able to process larger dataset vocabularies.

A

Experimental Design

This appendix contains the full experimental design of our study.

Table A.1: Multilayer Perceptron - MagnaTagATune

Dataset	Subset	Feature	Seed	# Hidden	Hidden	Lr	L2	$p_{dropout}$
magnatag	16	mfcc	0	3	128	1.62e-03	1.14e-05	1.97e-01
magnatag	16	mfcc	1	2	128	6.28e-03	6.63e-08	2.78e-01
magnatag	16	mfcc	2	1	32	1.39e-02	1.73e-07	0.00e+00
magnatag	16	mfcc	5	1	128	1.00e+00	2.35e-04	0.00e+00
magnatag	32	mfcc	0	3	128	2.61e-03	2.22e-06	2.44e-01
magnatag	32	mfcc	1	3	128	2.37e-03	1.61e-04	0.00e+00
magnatag	32	mfcc	2	2	64	3.12e-03	1.31e-03	0.00e+00
magnatag	32	mfcc	5	1	128	7.69e-03	1.00e-08	2.64e-01
magnatag	64	mfcc	0	1	64	7.63e-03	2.14e-04	0.00e+00
magnatag	64	mfcc	1	1	128	9.26e-03	9.26e-08	5.00e-01
magnatag	64	mfcc	2	3	64	9.44e-03	2.63e-04	0.00e+00
magnatag	64	mfcc	5	1	64	5.66e-02	1.16e-04	0.00e+00
magnatag	128	mfcc	0	3	64	1.14e-02	3.26e-08	8.79e-02
magnatag	128	mfcc	1	1	128	6.92e-03	3.07e-06	5.00e-01
magnatag	128	mfcc	2	2	128	3.47e-03	2.26e-07	2.18e-01
magnatag	128	mfcc	5	1	32	1.65e-02	8.44e-05	0.00e+00
magnatag	188	mfcc	0	3	128	6.17e-03	5.25e-08	1.62e-01
magnatag	188	mfcc	1	2	128	4.95e-03	1.00e-08	2.38e-01
magnatag	188	mfcc	2	2	128	7.97e-03	2.85e-05	2.37e-01
magnatag	188	mfcc	5	2	128	2.44e-03	2.78e-07	3.04e-01
magnatag	16	rand	0	1	64	1.00e+00	1.00e-08	0.00e+00
magnatag	16	rand	1	3	32	1.00e+00	1.00e+00	0.00e+00
magnatag	16	rand	2	3	64	1.00e+00	1.00e+00	0.00e+00
magnatag	16	rand	5	3	128	2.30e-02	1.00e-08	9.92e-02
magnatag	32	rand	0	2	128	1.40e-01	6.00e-02	3.12e-01
magnatag	32	rand	1	2	64	9.96e-01	3.66e-05	4.33e-01
magnatag	32	rand	2	3	128	4.99e-02	1.00e+00	5.00e-01
magnatag	32	rand	5	1	32	1.00e+00	1.00e+00	0.00e+00
magnatag	64	rand	0	3	64	2.27e-04	4.88e-02	1.69e-01
magnatag	64	rand	1	1	64	1.98e-01	1.00e-08	5.00e-01
magnatag	64	rand	2	1	64	2.00e-01	1.00e-08	2.01e-01
magnatag	64	rand	5	3	32	1.00e+00	1.00e+00	0.00e+00
magnatag	128	rand	0	3	128	8.48e-04	2.16e-07	5.00e-01
magnatag	128	rand	1	2	128	1.33e-03	2.70e-03	3.60e-01
magnatag	128	rand	2	3	32	3.16e-03	1.00e-08	5.00e-01
magnatag	128	rand	5	1	128	1.00e-03	1.00e-03	2.00e-01
magnatag	188	rand	0	3	32	1.00e+00	1.00e-08	2.32e-01
magnatag	188	rand	1	1	128	2.07e-05	8.93e-02	7.06e-02
magnatag	188	rand	2	1	32	3.90e-02	1.00e+00	5.00e-01
magnatag	188	rand	5	2	128	8.30e-03	9.20e-05	5.00e-01

Table A.2: Multilayer Perceptron - Million Song Dataset

Dataset	Subset	Feature	Seed	# Hidden	Hidden	Lr	L2	$p_{dropout}$
msd	16	mfcc	0	3	128	2.99e-03	2.74e-08	1.88e-01
msd	16	mfcc	1	3	32	1.77e-03	1.00e-08	0.00e+00
msd	16	mfcc	2	3	64	1.03e-02	1.43e-04	0.00e+00
msd	16	mfcc	5	3	64	2.55e-03	1.93e-04	0.00e+00
msd	32	mfcc	0	2	128	2.49e-04	8.41e-06	1.92e-01
msd	32	mfcc	1	3	128	1.44e-03	1.44e-05	1.70e-01
msd	32	mfcc	2	3	64	3.48e-03	1.00e-08	4.94e-02
msd	32	mfcc	5	3	128	3.62e-03	1.33e-04	0.00e+00
msd	64	mfcc	0	3	128	4.56e-03	7.25e-08	3.48e-02
msd	64	mfcc	1	3	128	4.63e-03	6.89e-06	1.55e-01
msd	64	mfcc	2	3	128	8.02e-04	3.44e-05	1.91e-01
msd	64	mfcc	5	3	128	2.88e-03	1.03e-04	0.00e+00
msd	128	mfcc	0	3	128	1.70e-03	1.68e-05	6.68e-02
msd	128	mfcc	1	3	128	1.75e-03	1.37e-05	5.57e-02
msd	128	mfcc	2	3	128	5.93e-04	1.00e-08	4.75e-02
msd	128	mfcc	5	3	128	1.52e-03	3.43e-05	1.54e-01
msd	256	mfcc	0	3	128	2.42e-03	2.01e-05	0.00e+00
msd	256	mfcc	1	3	128	1.41e-03	2.27e-05	7.75e-02
msd	256	mfcc	2	3	128	9.44e-04	4.46e-06	1.38e-01
msd	256	mfcc	5	3	128	2.73e-03	1.49e-05	2.00e-01
msd	512	mfcc	0	3	128	2.01e-03	2.59e-05	0.00e+00
msd	512	mfcc	1	3	128	1.12e-03	6.19e-06	0.00e+00
msd	512	mfcc	2	3	128	1.92e-03	1.32e-05	0.00e+00
msd	512	mfcc	5	3	128	5.68e-04	6.70e-06	1.62e-01
msd	1024	mfcc	0	3	128	1.89e-03	1.00e-05	0.00e+00
msd	1024	mfcc	1	3	128	7.70e-04	1.00e-08	8.87e-02
msd	1024	mfcc	2	3	128	1.16e-03	3.13e-06	1.27e-01
msd	1024	mfcc	5	3	128	2.24e-04	1.64e-06	2.65e-02
msd	16	rand	0	3	32	9.41e-01	3.09e-01	1.18e-01
msd	16	rand	1	1	64	9.11e-01	5.92e-01	4.85e-01
msd	16	rand	2	1	64	9.12e-01	1.20e-08	3.40e-01
msd	16	rand	5	3	128	7.65e-02	1.00e+00	5.00e-01
msd	32	rand	0	1	32	1.00e+00	1.00e+00	3.22e-01
msd	32	rand	1	1	128	8.91e-01	1.07e-08	4.17e-01
msd	32	rand	2	3	32	8.84e-01	2.90e-08	9.06e-03
msd	32	rand	5	3	128	1.51e-02	1.00e+00	5.00e-01
msd	64	rand	0	1	128	3.70e-02	1.00e+00	5.00e-01
msd	64	rand	1	3	64	1.00e+00	1.67e-03	1.14e-02
msd	64	rand	2	1	128	4.14e-01	1.09e-05	0.00e+00
msd	64	rand	5	3	32	1.70e-02	1.00e+00	4.06e-01
msd	128	rand	0	3	128	1.00e+00	1.00e-08	0.00e+00
msd	128	rand	1	1	32	6.87e-03	1.00e+00	0.00e+00
msd	128	rand	2	3	32	4.09e-03	1.00e+00	1.84e-01
msd	128	rand	5	3	32	9.13e-02	5.50e-07	0.00e+00
msd	256	rand	0	3	128	8.34e-01	3.77e-08	4.10e-01
msd	256	rand	1	1	64	1.00e+00	1.00e+00	5.00e-01
msd	256	rand	2	1	32	7.00e-02	1.26e-01	2.65e-01
msd	256	rand	5	3	64	9.84e-04	1.00e-08	5.00e-01
msd	512	rand	0	3	64	8.28e-01	4.59e-01	4.76e-01
msd	512	rand	1	1	64	1.55e-02	4.18e-01	1.72e-02
msd	512	rand	2	1	32	1.13e-03	1.00e+00	0.00e+00
msd	512	rand	5	1	32	7.19e-02	3.93e-05	0.00e+00
msd	1024	rand	0	3	128	1.00e+00	1.00e-08	5.00e-01
msd	1024	rand	1	2	128	1.80e-03	1.00e+00	0.00e+00
msd	1024	rand	2	3	32	1.11e-02	6.70e-02	5.00e-01
msd	1024	rand	5	3	32	1.32e-01	1.56e-08	1.94e-03

Table A.3: Matrix Factorization - MagnaTagATune

Dataset	Subset	Feature	Seed	# Hidden	# Iter	λ	L2	α
magnatag	16	mfcc	0	128	20	1.00e+07	1.00e-07	2.97e-02
magnatag	16	mfcc	1	128	20	1.36e+06	5.36e+01	6.57e+02
magnatag	16	mfcc	2	32	20	1.65e-02	9.83e+02	7.04e+02
magnatag	16	mfcc	5	32	8	1.00e-07	1.00e-07	1.99e+05
magnatag	32	mfcc	0	128	15	1.21e+05	2.08e-03	1.00e-07
magnatag	32	mfcc	1	128	12	1.00e+07	1.33e+05	7.98e+04
magnatag	32	mfcc	2	64	8	1.21e+02	1.42e+01	1.00e-07
magnatag	32	mfcc	5	128	12	1.00e-07	2.75e+01	1.07e+03
magnatag	64	mfcc	0	128	20	1.00e+07	2.73e-02	6.64e+00
magnatag	64	mfcc	1	128	20	1.00e+07	2.13e-01	3.62e-04
magnatag	64	mfcc	2	128	20	1.00e+07	4.92e-07	8.07e-06
magnatag	64	mfcc	5	128	14	2.31e-06	1.10e+02	1.68e+01
magnatag	128	mfcc	0	128	5	1.00e+07	5.49e+01	1.00e-07
magnatag	128	mfcc	1	64	5	1.00e+07	1.17e-04	1.00e-07
magnatag	128	mfcc	2	128	7	1.71e+04	1.00e-07	1.69e-05
magnatag	128	mfcc	5	128	17	2.43e+06	1.00e-07	3.11e+01
magnatag	188	mfcc	0	128	20	1.00e+07	2.88e-03	4.57e-04
magnatag	188	mfcc	1	128	20	1.00e+07	5.84e-07	4.87e-04
magnatag	188	mfcc	2	32	5	1.00e+07	2.26e-02	1.59e-04
magnatag	188	mfcc	5	128	8	1.00e-07	7.43e-02	8.36e+01
magnatag	16	rand	0	32	5	1.00e+07	1.00e+07	1.00e+07
magnatag	16	rand	1	128	20	1.00e-07	5.85e+02	9.07e+01
magnatag	16	rand	2	128	20	2.11e-05	3.05e+02	1.86e+05
magnatag	16	rand	5	64	10	5.45e+02	1.00e-07	2.74e+05
magnatag	32	rand	0	64	20	2.57e+06	9.79e+06	9.37e+06
magnatag	32	rand	1	128	20	5.84e-06	7.56e+04	1.00e+07
magnatag	32	rand	2	128	20	1.00e-07	4.39e+03	1.66e+06
magnatag	32	rand	5	64	16	1.32e-07	2.40e+02	2.72e+04
magnatag	64	rand	0	32	5	2.68e+03	7.38e-06	2.73e-03
magnatag	64	rand	1	64	17	2.60e+02	2.80e+02	2.43e+01
magnatag	64	rand	2	64	11	1.08e+00	2.32e+02	4.59e+01
magnatag	64	rand	5	128	20	1.00e-07	4.76e-01	8.75e+02
magnatag	128	rand	0	128	5	1.04e-01	2.71e+01	1.00e-07
magnatag	128	rand	1	128	5	9.49e+02	7.40e+04	1.00e+07
magnatag	128	rand	2	128	5	4.23e-03	1.43e+03	8.90e+05
magnatag	128	rand	5	128	20	1.00e-07	4.69e-02	3.10e+01
magnatag	188	rand	0	32	20	1.00e-07	1.00e+07	1.00e-07
magnatag	188	rand	1	128	16	3.57e-02	6.56e+04	9.32e+05
magnatag	188	rand	2	128	14	4.47e-03	1.66e+04	1.00e+07
magnatag	188	rand	5	128	19	3.66e+02	9.05e-01	1.59e+02

Table A.4: Matrix Factorization - Million Song Dataset

Dataset	Subset	Feature	Seed	# Hidden	# Iter	λ	L2	α
msd	16	mfcc	0	128	20	1.00e+07	1.29e-05	2.14e-03
msd	16	mfcc	1	128	20	4.39e+04	3.62e-01	2.62e-06
msd	16	mfcc	2	64	18	1.10e+04	2.65e+02	4.53e+01
msd	16	mfcc	5	64	5	1.00e+07	1.00e+07	4.07e+05
msd	32	mfcc	0	128	20	1.15e+06	1.09e-04	2.62e-05
msd	32	mfcc	1	64	5	4.14e+03	1.45e+00	1.00e-07
msd	32	mfcc	2	64	16	1.26e+02	1.48e+02	5.35e+00
msd	32	mfcc	5	64	19	2.71e-03	1.64e+02	1.85e+02
msd	64	mfcc	0	128	20	1.76e+06	6.33e-03	6.29e-03
msd	64	mfcc	1	128	20	1.11e+06	1.80e-05	7.48e-03
msd	64	mfcc	2	128	20	4.54e+04	1.00e-07	5.99e-07
msd	64	mfcc	5	64	12	3.91e-01	4.94e+02	8.50e+01
msd	128	mfcc	0	128	20	1.00e+07	1.00e-07	1.48e-05
msd	128	mfcc	1	128	20	1.97e+04	1.77e+00	6.00e-01
msd	128	mfcc	2	32	11	1.10e+04	1.58e-02	7.31e-01
msd	128	mfcc	5	32	20	1.00e-07	9.87e+02	2.80e+01
msd	256	mfcc	0	128	20	1.13e+06	8.37e-02	5.14e+00
msd	256	mfcc	1	128	20	1.34e+06	1.00e-07	1.00e-07
msd	256	mfcc	2	32	5	1.00e-07	1.46e+02	4.70e+00
msd	256	mfcc	5	32	20	6.71e-04	1.00e-07	5.81e+00
msd	512	mfcc	0	32	5	1.00e+07	2.48e-03	2.46e-05
msd	512	mfcc	1	32	20	1.00e-07	2.88e+03	1.70e+02
msd	512	mfcc	2	32	20	1.00e-07	7.26e-01	9.22e+00
msd	512	mfcc	5	32	20	1.00e-07	1.07e+00	1.44e+01
msd	1024	mfcc	0	128	20	1.00e+07	7.89e-05	3.71e-05
msd	1024	mfcc	1	32	20	6.94e+02	1.00e-07	1.68e+01
msd	1024	mfcc	2	32	20	2.16e+00	2.05e-01	2.22e+01
msd	1024	mfcc	5	64	19	8.45e+00	3.06e+03	2.96e+02
msd	16	rand	0	32	18	5.08e+03	1.83e-07	1.03e-07
msd	16	rand	1	128	20	1.00e-07	3.46e+03	3.35e+03
msd	16	rand	2	128	17	1.05e-02	7.90e+02	8.45e+00
msd	16	rand	5	64	5	2.89e-04	8.99e+02	8.45e+01
msd	32	rand	0	128	19	1.59e+01	1.00e+07	4.27e+04
msd	32	rand	1	128	20	5.89e-01	7.58e+02	7.66e+00
msd	32	rand	2	64	19	7.70e-01	1.24e+03	3.29e+01
msd	32	rand	5	64	20	4.20e-03	3.34e+02	7.24e+01
msd	64	rand	0	32	10	3.23e+03	3.23e+01	3.55e-03
msd	64	rand	1	64	19	1.00e-07	6.81e+02	6.43e+00
msd	64	rand	2	128	20	1.04e-01	3.60e+03	6.62e+04
msd	64	rand	5	128	20	8.22e-05	1.07e+02	2.62e-05
msd	128	rand	0	64	20	2.49e+03	3.95e-04	1.83e-05
msd	128	rand	1	32	20	1.56e-02	8.78e-06	6.20e+00
msd	128	rand	2	32	20	1.00e-07	1.00e-07	6.05e+00
msd	128	rand	5	128	20	5.78e-07	1.04e+03	3.22e+01
msd	256	rand	0	64	13	4.17e+03	3.04e-06	4.27e-01
msd	256	rand	1	32	20	1.00e-07	1.00e-07	3.98e+00
msd	256	rand	2	32	20	1.00e-07	1.44e-04	5.53e+00
msd	256	rand	5	128	20	5.16e-01	5.25e+00	1.27e+02
msd	512	rand	0	32	5	2.82e+03	7.62e-05	4.96e-05
msd	512	rand	1	32	20	1.99e-06	8.72e-01	1.02e+01
msd	512	rand	2	32	20	1.94e-06	6.16e-01	1.10e+01
msd	512	rand	5	32	20	4.39e-04	8.82e-01	1.21e+01
msd	1024	rand	0	32	19	1.03e-06	1.83e+02	1.06e-07
msd	1024	rand	1	32	20	6.06e-02	2.86e-02	9.80e+00
msd	1024	rand	2	32	20	9.39e-06	6.33e-01	1.98e+01
msd	1024	rand	5	32	20	1.21e-02	1.96e+00	1.39e+01

Table A.5: Relational Graph Convolutional Network - MagnaTagATune

Dataset	Subset	Feature	Seed	# Hidden	Hidden	Lr	L2	$p_{dropout}$
magnatag	16	mfcc	0	1	128	1.22e-03	1.00e-07	5.00e-01
magnatag	16	mfcc	1	1	128	3.05e-03	5.41e-03	3.12e-01
magnatag	16	mfcc	2	1	64	6.82e-03	1.00e-07	5.00e-01
magnatag	16	mfcc	5	2	128	4.40e-04	5.65e-03	3.60e-01
magnatag	32	mfcc	0	2	128	1.16e-03	1.34e-06	3.22e-01
magnatag	32	mfcc	1	2	128	1.15e-03	5.37e-05	2.57e-01
magnatag	32	mfcc	2	1	128	1.43e-03	2.19e-04	5.00e-01
magnatag	32	mfcc	5	1	64	1.37e-02	1.00e-07	5.00e-01
magnatag	64	mfcc	0	2	128	8.86e-04	1.25e-06	3.04e-01
magnatag	64	mfcc	1	1	128	1.74e-03	6.79e-07	5.00e-01
magnatag	64	mfcc	2	1	64	1.80e-02	1.80e-03	1.63e-01
magnatag	64	mfcc	5	1	64	1.65e-03	1.00e-07	3.09e-02
magnatag	128	mfcc	0	1	128	9.91e-03	5.97e-03	0.00e+00
magnatag	128	mfcc	1	1	128	5.49e-03	2.54e-03	6.24e-03
magnatag	128	mfcc	2	1	128	3.28e-03	1.32e-03	0.00e+00
magnatag	128	mfcc	5	1	64	6.53e-03	7.87e-06	1.83e-01
magnatag	188	mfcc	0	2	64	8.34e-04	1.27e-03	1.10e-01
magnatag	188	mfcc	1	2	64	2.48e-03	2.20e-03	1.32e-01
magnatag	188	mfcc	2	1	128	1.03e-02	1.52e-03	0.00e+00
magnatag	188	mfcc	5	1	64	5.93e-03	2.75e-04	0.00e+00
magnatag	16	rand	0	3	32	4.03e-03	1.99e-02	1.84e-02
magnatag	16	rand	1	1	64	1.00e-02	3.82e-07	4.50e-01
magnatag	16	rand	2	1	64	2.65e-02	8.19e-04	0.00e+00
magnatag	16	rand	5	1	64	2.68e-02	4.52e-03	1.80e-01
magnatag	32	rand	0	2	64	2.63e-03	8.12e-07	3.33e-01
magnatag	32	rand	1	1	64	7.97e-03	4.22e-05	4.05e-01
magnatag	32	rand	2	1	32	5.17e-02	2.04e-05	5.00e-01
magnatag	32	rand	5	1	128	2.59e-02	1.82e-05	1.87e-01
magnatag	64	rand	0	3	64	2.57e-03	1.00e-07	2.08e-01
magnatag	64	rand	1	1	32	7.42e-03	2.31e-05	4.58e-01
magnatag	64	rand	2	1	128	1.72e-02	8.46e-04	0.00e+00
magnatag	64	rand	5	1	128	1.32e-02	2.01e-07	4.21e-01
magnatag	128	rand	0	2	128	7.19e-03	2.29e-02	7.02e-02
magnatag	128	rand	1	1	64	6.64e-02	1.09e-03	0.00e+00
magnatag	128	rand	2	2	32	9.85e-03	4.11e-05	1.74e-01
magnatag	128	rand	5	2	64	2.04e-02	5.51e-07	2.37e-01
magnatag	188	rand	0	3	128	1.69e-02	2.37e-02	3.24e-02
magnatag	188	rand	1	1	32	3.49e-02	1.00e-07	5.00e-01
magnatag	188	rand	2	1	32	2.82e-02	2.01e-05	5.00e-01
magnatag	188	rand	5	2	32	2.20e-02	2.21e-06	1.80e-01

Table A.6: Relational Graph Convolutional Network - Million Song Dataset

Dataset	Subset	Feature	Seed	# Hidden	Hidden	Lr	L2	$p_{dropout}$
msd	16	mfcc	0	2	128	1.51e-03	1.16e-03	4.29e-01
msd	16	mfcc	1	2	128	6.31e-04	5.81e-04	3.01e-01
msd	16	mfcc	2	2	128	1.07e-03	1.65e-03	4.23e-01
msd	16	mfcc	5	1	128	5.07e-03	3.95e-03	2.23e-01
msd	32	mfcc	0	2	128	7.57e-04	4.70e-03	2.09e-01
msd	32	mfcc	1	2	128	8.31e-04	1.99e-03	2.23e-01
msd	32	mfcc	2	2	128	1.89e-03	9.23e-04	2.63e-01
msd	32	mfcc	5	1	128	5.69e-04	1.00e-07	4.80e-01
msd	64	mfcc	0	1	128	7.26e-03	9.01e-03	1.92e-01
msd	64	mfcc	1	2	64	1.10e-03	1.40e-06	1.36e-01
msd	64	mfcc	2	2	128	2.80e-03	2.37e-04	3.49e-01
msd	64	mfcc	5	1	128	6.15e-04	1.14e-05	5.00e-01
msd	128	mfcc	0	2	128	2.00e-03	1.39e-03	2.87e-01
msd	128	mfcc	1	2	64	2.80e-03	3.36e-04	2.13e-01
msd	128	mfcc	2	2	128	2.29e-03	6.17e-07	2.76e-01
msd	128	mfcc	5	1	128	1.07e-03	3.37e-04	5.00e-01
msd	256	mfcc	0	1	64	2.78e-04	4.84e-03	0.00e+00
msd	256	mfcc	1	1	64	2.27e-03	3.21e-03	0.00e+00
msd	256	mfcc	2	2	128	6.38e-04	1.00e-07	2.20e-01
msd	256	mfcc	5	1	32	2.51e-02	1.00e-07	5.00e-01
msd	512	mfcc	0	1	128	7.97e-03	1.94e-03	1.68e-01
msd	512	mfcc	1	1	128	2.22e-03	1.27e-03	9.79e-02
msd	512	mfcc	2	2	128	9.18e-04	1.14e-04	1.99e-01
msd	512	mfcc	5	1	128	6.69e-03	1.16e-07	5.00e-01
msd	1024	mfcc	0	1	64	3.54e-04	1.87e-03	0.00e+00
msd	1024	mfcc	1	1	128	3.11e-03	1.52e-03	0.00e+00
msd	1024	mfcc	2	2	128	9.00e-04	2.25e-06	1.82e-01
msd	1024	mfcc	5	1	128	2.40e-03	1.00e-07	3.46e-01
msd	16	rand	0	3	64	1.31e-03	1.00e-07	4.90e-01
msd	16	rand	1	1	128	1.89e-02	1.16e-04	5.00e-01
msd	16	rand	2	1	128	3.43e-03	1.76e-04	5.00e-01
msd	16	rand	5	2	128	1.28e-02	1.04e-02	0.00e+00
msd	32	rand	0	2	64	1.44e-02	1.26e-02	3.67e-01
msd	32	rand	1	1	128	1.87e-02	4.23e-03	0.00e+00
msd	32	rand	2	1	32	1.00e+00	3.72e-04	5.00e-01
msd	32	rand	5	2	128	7.86e-03	1.70e-03	0.00e+00
msd	64	rand	0	2	64	4.07e-03	1.00e-07	5.00e-01
msd	64	rand	1	2	64	2.63e-03	3.40e-05	8.80e-02
msd	64	rand	2	2	64	4.30e-03	1.28e-04	1.01e-01
msd	64	rand	5	1	32	1.38e-02	1.02e-05	5.00e-01
msd	128	rand	0	2	32	4.52e-03	2.01e-04	5.00e-01
msd	128	rand	1	1	32	5.19e-01	3.10e-05	5.00e-01
msd	128	rand	2	2	32	3.37e-03	6.24e-06	1.98e-02
msd	128	rand	5	1	64	9.19e-02	2.66e-04	0.00e+00
msd	256	rand	0	2	64	2.04e-02	5.51e-07	2.37e-01
msd	256	rand	1	2	32	2.43e-03	1.00e-07	0.00e+00
msd	256	rand	2	2	64	6.11e-03	3.22e-06	8.86e-02
msd	256	rand	5	1	32	1.82e-02	1.00e-07	4.05e-01
msd	512	rand	0	3	64	7.65e-02	2.26e-03	1.53e-01
msd	512	rand	1	1	64	3.65e-02	4.70e-07	5.00e-01
msd	512	rand	2	2	128	1.50e-02	6.80e-05	5.46e-02
msd	512	rand	5	1	64	2.63e-02	2.22e-07	5.00e-01
msd	1024	rand	0	2	64	2.58e-03	2.52e-06	5.00e-01
msd	1024	rand	1	1	32	3.73e-02	9.06e-07	1.83e-01
msd	1024	rand	2	1	128	5.38e-02	4.11e-06	5.00e-01
msd	1024	rand	5	1	64	3.13e-02	3.86e-07	5.00e-01

B

Classification Accuracy Results

This appendix contains the classification accuracy results for both MSD and MTAT. Each table contains per dataset subset the average and standard deviation of the 10 folds. The model with the highest score per subset is marked in bold.

Million Song Dataset

This subsection contains the classification accuracy results for MSD.

AUC-Micro

	mf	mlp	prior	rgcn
16	0.7612 ± 0.0025	**0.8278 ± 0.0029**	0.6367 ± 0.0041	0.7480 ± 0.0036
32	0.7689 ± 0.0026	**0.8293 ± 0.0027**	0.6145 ± 0.0030	0.7559 ± 0.0050
64	0.7934 ± 0.0027	**0.8466 ± 0.0019**	0.6314 ± 0.0035	0.7582 ± 0.0083
128	0.8032 ± 0.0033	**0.8650 ± 0.0033**	0.6797 ± 0.0038	0.8005 ± 0.0163
256	0.8126 ± 0.0030	**0.8757 ± 0.0021**	0.7181 ± 0.0024	0.7537 ± 0.0132
512	0.7774 ± 0.0033	**0.8935 ± 0.0025**	0.7564 ± 0.0029	0.8377 ± 0.0055
1024	0.8240 ± 0.0021	**0.9077 ± 0.0019**	0.8042 ± 0.0034	0.8230 ± 0.0127

Table B.1: AUC micro (element-wise) averaged over all folds for MSD when the MFCC feature is given.

	mf	mlp	prior	rgcn
16	0.5003 ± 0.0048	**0.6491 ± 0.0065**	0.6367 ± 0.0041	0.5464 ± 0.0518
32	0.4995 ± 0.0004	0.6139 ± 0.0069	**0.6145 ± 0.0030**	0.5705 ± 0.0120
64	0.4996 ± 0.0013	0.6175 ± 0.0127	**0.6314 ± 0.0035**	0.5898 ± 0.0409
128	0.4988 ± 0.0038	0.6684 ± 0.0082	**0.6797 ± 0.0038**	0.6386 ± 0.0384
256	0.5020 ± 0.0052	0.7115 ± 0.0055	**0.7181 ± 0.0024**	0.6728 ± 0.0223
512	0.5014 ± 0.0051	0.7146 ± 0.0172	**0.7564 ± 0.0029**	0.6928 ± 0.0599
1024	0.5002 ± 0.0042	0.7907 ± 0.0045	**0.8042 ± 0.0034**	0.7750 ± 0.0032

Table B.2: AUC micro (element-wise) averaged over all folds for MSD when the random feature is given.

AUC-Macro

	mf	mlp	prior	rgcn
16	0.7386 ± 0.0023	**0.7795 ± 0.0037**	0.5000 ± 0.0000	0.6944 ± 0.0065
32	0.7390 ± 0.0018	**0.7827 ± 0.0032**	0.5000 ± 0.0000	0.7055 ± 0.0054
64	0.7477 ± 0.0022	**0.7815 ± 0.0028**	0.5000 ± 0.0000	0.6931 ± 0.0049
128	0.7432 ± 0.0040	**0.7824 ± 0.0030**	0.5000 ± 0.0000	0.7094 ± 0.0043
256	0.7502 ± 0.0049	**0.7857 ± 0.0054**	0.5000 ± 0.0000	0.6623 ± 0.0106
512	0.7103 ± 0.0049	**0.7870 ± 0.0043**	0.5000 ± 0.0000	0.7156 ± 0.0061
1024	0.7524 ± 0.0041	**0.7891 ± 0.0052**	0.5000 ± 0.0000	0.6834 ± 0.0122

Table B.3: AUC macro (tag level) averaged over all folds for MSD when the MFCC feature is given.

	mf	mlp	prior	rgcn
16	**0.5003 ± 0.0057**	0.4988 ± 0.0030	0.5000 ± 0.0000	0.4997 ± 0.0018
32	**0.5013 ± 0.0020**	0.5000 ± 0.0000	0.5000 ± 0.0000	0.4997 ± 0.0027
64	**0.5010 ± 0.0020**	0.5006 ± 0.0027	0.5000 ± 0.0000	0.5009 ± 0.0030
128	0.5009 ± 0.0055	0.5000 ± 0.0000	0.5000 ± 0.0000	**0.5012 ± 0.0050**
256	**0.5029 ± 0.0056**	0.5000 ± 0.0000	0.5000 ± 0.0000	0.5013 ± 0.0063
512	**0.5016 ± 0.0041**	0.5001 ± 0.0006	0.5000 ± 0.0000	0.4998 ± 0.0088
1024	**0.5013 ± 0.0060**	0.5000 ± 0.0000	0.5000 ± 0.0000	0.4992 ± 0.0042

Table B.4: AUC macro (tag level) averaged over all folds for MSD when the random feature is given.

AUC-Samples

	mf	mlp	prior	rgcn
16	0.7628 ± 0.0029	**0.8239 ± 0.0024**	0.6367 ± 0.0041	0.7521 ± 0.0038
32	0.7720 ± 0.0028	**0.8283 ± 0.0029**	0.6145 ± 0.0030	0.7582 ± 0.0045
64	0.7961 ± 0.0027	**0.8484 ± 0.0019**	0.6314 ± 0.0035	0.7977 ± 0.0040
128	0.8051 ± 0.0031	**0.8683 ± 0.0036**	0.6797 ± 0.0038	0.8066 ± 0.0171
256	0.8167 ± 0.0028	**0.8799 ± 0.0021**	0.7181 ± 0.0024	0.8366 ± 0.0035
512	0.7782 ± 0.0038	**0.8979 ± 0.0026**	0.7564 ± 0.0029	0.8640 ± 0.0032
1024	0.8248 ± 0.0020	**0.9125 ± 0.0019**	0.8042 ± 0.0034	0.8850 ± 0.0025

Table B.5: AUC samples (song level) averaged over all folds for MSD when the MFCC feature is given.

	mf	mlp	prior	rgcn
16	0.4997 ± 0.0046	**0.6491 ± 0.0064**	0.6367 ± 0.0041	0.5842 ± 0.0435
32	0.4972 ± 0.0015	0.6139 ± 0.0069	**0.6145 ± 0.0030**	0.5925 ± 0.0223
64	0.4999 ± 0.0031	0.6175 ± 0.0126	**0.6314 ± 0.0035**	0.5919 ± 0.0374
128	0.4988 ± 0.0043	0.6684 ± 0.0082	**0.6797 ± 0.0038**	0.6433 ± 0.0354
256	0.5019 ± 0.0054	0.7115 ± 0.0055	**0.7181 ± 0.0024**	0.6748 ± 0.0244
512	0.5013 ± 0.0060	0.7146 ± 0.0172	**0.7564 ± 0.0029**	0.6979 ± 0.0524
1024	0.4998 ± 0.0050	0.7907 ± 0.0045	**0.8042 ± 0.0034**	0.7745 ± 0.0044

Table B.6: AUC samples (song level) averaged over all folds for MSD when the random feature is given.

MagnaTagATune

This subsection contains the classification accuracy results for MTAT.

AUC-Micro

	mf	mlp	prior	rgcn
16	0.8625 ± 0.0113	**0.8918 ± 0.0089**	0.6115 ± 0.0114	0.8359 ± 0.0091
32	0.8504 ± 0.0052	**0.8844 ± 0.0039**	0.6456 ± 0.0114	0.8424 ± 0.0074
64	0.8666 ± 0.0071	**0.8945 ± 0.0065**	0.7088 ± 0.0068	0.8608 ± 0.0062
128	0.8086 ± 0.0102	**0.9108 ± 0.0041**	0.7873 ± 0.0080	0.8384 ± 0.0117
188	0.8969 ± 0.0076	**0.9416 ± 0.0029**	0.8665 ± 0.0095	0.9202 ± 0.0064

Table B.7: AUC micro (element-wise) averaged over all folds for MTAT when the MFCC feature is given.

	mf	mlp	prior	rgcn
16	0.5024 ± 0.0110	0.6014 ± 0.0144	**0.6115 ± 0.0114**	0.5761 ± 0.0182
32	0.4984 ± 0.0108	0.6414 ± 0.0120	**0.6456 ± 0.0114**	0.6327 ± 0.0165
64	0.5024 ± 0.0115	0.7024 ± 0.0124	**0.7088 ± 0.0068**	0.6248 ± 0.0856
128	0.4885 ± 0.0145	0.7870 ± 0.0078	**0.7873 ± 0.0080**	0.6542 ± 0.0355
188	0.5054 ± 0.0160	0.8642 ± 0.0099	**0.8665 ± 0.0095**	0.6513 ± 0.0244

Table B.8: AUC micro (element-wise) averaged over all folds for MTAT when the random feature is given.

AUC-Macro

	mf	mlp	prior	rgcn
16	0.8613 ± 0.0117	**0.8774 ± 0.0103**	0.5000 ± 0.0000	0.8309 ± 0.0100
32	0.8425 ± 0.0092	**0.8593 ± 0.0069**	0.5000 ± 0.0000	0.8107 ± 0.0114
64	0.8405 ± 0.0088	**0.8520 ± 0.0071**	0.5000 ± 0.0000	0.8079 ± 0.0087
128	0.7746 ± 0.0087	**0.8470 ± 0.0084**	0.5000 ± 0.0000	0.7610 ± 0.0088
188	0.8313 ± 0.0178	**0.8487 ± 0.0070**	0.5000 ± 0.0000	0.7869 ± 0.0178

Table B.9: AUC macro (tag level) averaged over all folds for MTAT when the MFCC feature is given.

	mf	mlp	prior	rgcn
16	0.5022 ± 0.0107	0.4980 ± 0.0080	**0.5000 ± 0.0000**	**0.5027 ± 0.0092**
32	0.4969 ± 0.0125	0.4983 ± 0.0052	**0.5000 ± 0.0000**	**0.5037 ± 0.0083**
64	0.5026 ± 0.0162	**0.5032 ± 0.0087**	**0.5000 ± 0.0000**	0.4952 ± 0.0078
128	0.4917 ± 0.0150	0.4990 ± 0.0098	**0.5000 ± 0.0000**	0.4977 ± 0.0118
188	**0.5108 ± 0.0208**	**0.5000 ± 0.0000**	**0.5000 ± 0.0000**	0.5042 ± 0.0146

Table B.10: AUC micro (tag level) averaged over all folds for MTAT when the random feature is given.

AUC-Samples

	mf	mlp	prior	rgcn
16	0.8617 ± 0.0120	**0.9035 ± 0.0092**	0.6197 ± 0.0129	0.8382 ± 0.0094
32	0.8576 ± 0.0058	**0.9056 ± 0.0069**	0.6576 ± 0.0119	0.8553 ± 0.0091
64	0.8745 ± 0.0062	**0.9136 ± 0.0042**	0.7246 ± 0.0075	0.8800 ± 0.0051
128	0.8195 ± 0.0093	**0.9300 ± 0.0035**	0.8024 ± 0.0087	0.9131 ± 0.0060
188	0.8956 ± 0.0077	**0.9539 ± 0.0029**	0.8665 ± 0.0095	0.9283 ± 0.0074

Table B.11: AUC samples (song level) averaged over all folds for MTAT when the MFCC feature is given.

	mf	mlp	prior	rgcn
16	0.5014 ± 0.0128	0.6109 ± 0.0126	**0.6197 ± 0.0129**	0.6138 ± 0.0141
32	0.4961 ± 0.0177	0.6533 ± 0.0116	**0.6576 ± 0.0119**	0.6492 ± 0.0175
64	0.5014 ± 0.0117	0.7171 ± 0.0127	**0.7246 ± 0.0075**	0.6699 ± 0.0627
128	0.4902 ± 0.0128	0.8021 ± 0.0085	**0.8024 ± 0.0087**	0.8015 ± 0.0088
188	0.5055 ± 0.0164	0.8642 ± 0.0099	**0.8665 ± 0.0095**	0.8629 ± 0.0095

Table B.12: AUC samples (song level) averaged over all folds for MTAT when the random feature is given.

C

Ranking Accuracy Results

This appendix contains the ranking accuracy results for both MSD and MTAT. Each table contains per dataset subset the average and standard deviation of the 10 folds. The model with the highest score per subset is marked in bold.

Million Song Dataset

This subsection contains the ranking accuracy results for MSD.

0 Seed tags given

	mf	mlp	prior	rgcn	uniform
16	0.6586 ± 0.0027	**0.7295 ± 0.0032**	0.5117 ± 0.0056	0.6383 ± 0.0049	0.4385 ± 0.0032
32	0.5294 ± 0.0034	**0.5971 ± 0.0043**	0.3353 ± 0.0044	0.4921 ± 0.0075	0.1557 ± 0.0021
64	0.4528 ± 0.0050	**0.5112 ± 0.0057**	0.2233 ± 0.0036	0.4199 ± 0.0090	0.0655 ± 0.0011
128	0.3811 ± 0.0046	**0.4425 ± 0.0059**	0.1654 ± 0.0037	0.3243 ± 0.0223	0.0435 ± 0.0014
256	0.3130 ± 0.0037	**0.3674 ± 0.0032**	0.1307 ± 0.0031	0.2743 ± 0.0047	0.0222 ± 0.0011
512	0.2050 ± 0.0034	**0.3173 ± 0.0042**	0.1108 ± 0.0024	0.2404 ± 0.0084	0.0070 ± 0.0009
1024	0.2249 ± 0.0030	**0.2784 ± 0.0025**	0.0975 ± 0.0018	0.2101 ± 0.0049	0.0043 ± 0.0006

Table C.1: NDCG score with cutoff 10 (top-10 tags) for MSD when no seed tags are given for the MFCC feature.

	mf	mlp	prior	rgcn	uniform
16	0.3858 ± 0.0048	**0.5293 ± 0.0069**	0.5117 ± 0.0056	0.4587 ± 0.0497	0.4385 ± 0.0032
32	0.2622 ± 0.0038	**0.3445 ± 0.0047**	0.3353 ± 0.0044	0.3237 ± 0.0144	0.1557 ± 0.0021
64	0.1435 ± 0.0031	**0.2385 ± 0.0040**	0.2233 ± 0.0036	0.2136 ± 0.0233	0.0655 ± 0.0011
128	0.1029 ± 0.0040	**0.1830 ± 0.0052**	0.1654 ± 0.0037	0.1673 ± 0.0143	0.0435 ± 0.0014
256	0.0771 ± 0.0039	**0.1414 ± 0.0034**	0.1307 ± 0.0031	0.1087 ± 0.0139	0.0222 ± 0.0011
512	0.0665 ± 0.0018	**0.1194 ± 0.0023**	0.1108 ± 0.0024	0.1119 ± 0.0091	0.0070 ± 0.0009
1024	0.0579 ± 0.0028	**0.1055 ± 0.0032**	0.0975 ± 0.0018	0.0913 ± 0.0095	0.0043 ± 0.0006

Table C.2: NDCG score with cutoff 10 (top-10 tags) for MSD when no seed tags are given for the random feature.

1 Seed tag given

	mf	mlp	prior	rgcn	uniform
16	0.6788 ± 0.0060	**0.7214 ± 0.0044**	0.5581 ± 0.0044	0.6708 ± 0.0064	0.4702 ± 0.0029
32	0.5162 ± 0.0062	**0.5914 ± 0.0060**	0.3577 ± 0.0052	0.5171 ± 0.0073	0.2405 ± 0.0047
64	0.4255 ± 0.0051	**0.4981 ± 0.0060**	0.2503 ± 0.0073	0.4087 ± 0.0071	0.0787 ± 0.0034
128	0.3912 ± 0.0046	**0.4301 ± 0.0066**	0.1840 ± 0.0047	0.3492 ± 0.0079	0.0397 ± 0.0013
256	0.3090 ± 0.0065	**0.3766 ± 0.0036**	0.1550 ± 0.0038	0.2905 ± 0.0066	0.0210 ± 0.0016
512	0.3076 ± 0.0056	**0.3396 ± 0.0033**	0.1509 ± 0.0034	0.2657 ± 0.0041	0.0091 ± 0.0008
1024	0.2767 ± 0.0041	**0.3208 ± 0.0040**	0.1462 ± 0.0032	0.2442 ± 0.0046	0.0052 ± 0.0004

Table C.3: NDCG score with cutoff 10 (top-10 tags) for MSD when 1 seed tag is given for the MFCC feature.

	mf	mlp	prior	rgcn	uniform
16	**0.6232 ± 0.0061**	0.5548 ± 0.0055	0.5581 ± 0.0044	0.5568 ± 0.0093	0.4702 ± 0.0029
32	**0.4651 ± 0.0069**	0.3610 ± 0.0041	0.3577 ± 0.0052	0.4052 ± 0.0077	0.2405 ± 0.0047
64	**0.3867 ± 0.0067**	0.2557 ± 0.0061	0.2503 ± 0.0073	0.3345 ± 0.0098	0.0787 ± 0.0034
128	0.2783 ± 0.0042	0.1960 ± 0.0049	0.1840 ± 0.0047	**0.2941 ± 0.0061**	0.0397 ± 0.0013
256	**0.2794 ± 0.0058**	0.1594 ± 0.0046	0.1550 ± 0.0038	0.2381 ± 0.0120	0.0210 ± 0.0016
512	**0.2785 ± 0.0048**	0.1566 ± 0.0035	0.1509 ± 0.0034	0.2461 ± 0.0043	0.0091 ± 0.0008
1024	**0.2696 ± 0.0033**	0.1497 ± 0.0031	0.1462 ± 0.0032	0.2123 ± 0.0035	0.0052 ± 0.0004

Table C.4: NDCG score with cutoff 10 (top-10 tags) for MSD when 1 seed tag is given for the random feature.

2 Seed tags given

	mf	mlp	prior	rgcn	uniform
16	0.7389 ± 0.0024	**0.7772 ± 0.0032**	0.6337 ± 0.0032	0.7288 ± 0.0261	0.5454 ± 0.0030
32	0.5539 ± 0.0030	**0.6077 ± 0.0048**	0.4033 ± 0.0042	0.5524 ± 0.0067	0.3148 ± 0.0036
64	0.4724 ± 0.0086	**0.5134 ± 0.0054**	0.2862 ± 0.0031	0.4490 ± 0.0085	0.0945 ± 0.0020
128	0.3401 ± 0.0060	**0.4463 ± 0.0041**	0.2365 ± 0.0021	0.3890 ± 0.0038	0.0513 ± 0.0017
256	0.3516 ± 0.0029	**0.4097 ± 0.0036**	0.2136 ± 0.0039	0.3646 ± 0.0037	0.0277 ± 0.0009
512	0.3389 ± 0.0014	**0.3765 ± 0.0043**	0.1977 ± 0.0032	0.3398 ± 0.0033	0.0133 ± 0.0005
1024	0.3269 ± 0.0040	**0.3582 ± 0.0034**	0.1890 ± 0.0029	0.3183 ± 0.0039	0.0067 ± 0.0003

Table C.5: NDCG score with cutoff 10 (top-10 tags) for MSD when 2 seed tags are given for the MFCC feature.

	mf	mlp	prior	rgcn	uniform
16	**0.7100 ± 0.0040**	0.6448 ± 0.0036	0.6337 ± 0.0032	0.6656 ± 0.0059	0.5454 ± 0.0030
32	**0.5417 ± 0.0029**	0.4035 ± 0.0050	0.4033 ± 0.0042	0.3215 ± 0.0814	0.3148 ± 0.0036
64	**0.4367 ± 0.0077**	0.2863 ± 0.0032	0.2862 ± 0.0031	0.4075 ± 0.0083	0.0945 ± 0.0020
128	0.3292 ± 0.0050	0.2380 ± 0.0022	0.2365 ± 0.0021	**0.3305 ± 0.0355**	0.0513 ± 0.0017
256	**0.3353 ± 0.0021**	0.2145 ± 0.0036	0.2136 ± 0.0039	0.2915 ± 0.0433	0.0277 ± 0.0009
512	**0.3388 ± 0.0028**	0.1986 ± 0.0030	0.1977 ± 0.0032	0.2635 ± 0.0441	0.0133 ± 0.0005
1024	**0.3293 ± 0.0031**	0.1892 ± 0.0029	0.1890 ± 0.0029	0.3013 ± 0.0060	0.0067 ± 0.0003

Table C.6: NDCG score with cutoff 10 (top-10 tags) for MSD when 2 seed tags are given for the random feature.

5 Seed tags given

	mf	mlp	prior	rgcn	uniform
16	0.8161 ± 0.0057	**0.8426 ± 0.0053**	0.7420 ± 0.0075	0.8185 ± 0.0034	0.6486 ± 0.0048
32	0.6346 ± 0.0060	**0.6667 ± 0.0042**	0.4980 ± 0.0033	0.6412 ± 0.0068	0.4320 ± 0.0034
64	**0.5716 ± 0.0029**	0.5537 ± 0.0038	0.3819 ± 0.0036	0.5342 ± 0.0036	0.1300 ± 0.0015
128	**0.5180 ± 0.0033**	0.4919 ± 0.0034	0.3222 ± 0.0032	0.4694 ± 0.0052	0.0935 ± 0.0022
256	0.4048 ± 0.0021	**0.4494 ± 0.0033**	0.2873 ± 0.0033	0.4454 ± 0.0089	0.0487 ± 0.0010
512	0.4107 ± 0.0033	**0.4232 ± 0.0027**	0.2630 ± 0.0018	0.4114 ± 0.0041	0.0208 ± 0.0006
1024	**0.4533 ± 0.0028**	0.3949 ± 0.0033	0.2454 ± 0.0031	0.4001 ± 0.0053	0.0092 ± 0.0005

Table C.7: NDCG score with cutoff 10 (top-10 tags) for MSD when 5 seed tags are given for the MFCC feature.

	mf	mlp	prior	rgcn	uniform
16	**0.8161 ± 0.0042**	0.7761 ± 0.0068	0.7420 ± 0.0075	0.7798 ± 0.0112	0.6486 ± 0.0048
32	**0.6683 ± 0.0031**	0.5118 ± 0.0111	0.4980 ± 0.0033	0.5949 ± 0.0057	0.4320 ± 0.0034
64	**0.5406 ± 0.0029**	0.3840 ± 0.0038	0.3819 ± 0.0036	0.4980 ± 0.0045	0.1300 ± 0.0015
128	**0.5195 ± 0.0033**	0.3251 ± 0.0031	0.3222 ± 0.0032	0.4403 ± 0.0047	0.0935 ± 0.0022
256	**0.4730 ± 0.0042**	0.2887 ± 0.0035	0.2873 ± 0.0033	0.4278 ± 0.0069	0.0487 ± 0.0010
512	0.4122 ± 0.0026	0.2637 ± 0.0020	0.2630 ± 0.0018	**0.4135 ± 0.0032**	0.0208 ± 0.0006
1024	0.4083 ± 0.0031	0.2458 ± 0.0033	0.2454 ± 0.0031	**0.4093 ± 0.0049**	0.0092 ± 0.0005

Table C.8: NDCG score with cutoff 10 (top-10 tags) for MSD when 5 seed tags are given for the random feature.

MagnaTagATune

This subsection contains the ranking accuracy results for MTAT.

0 Seed tags given

	mf	mlp	prior	rgcn	uniform
16	0.8070 ± 0.0167	**0.8548 ± 0.0139**	0.5211 ± 0.0205	0.7664 ± 0.0128	0.4524 ± 0.0159
32	0.7106 ± 0.0119	**0.7751 ± 0.0157**	0.3886 ± 0.0142	0.6642 ± 0.0163	0.1997 ± 0.0067
64	0.6423 ± 0.0125	**0.6889 ± 0.0115**	0.3238 ± 0.0163	0.5880 ± 0.0120	0.0956 ± 0.0090
128	0.4778 ± 0.0168	**0.6428 ± 0.0164**	0.3028 ± 0.0149	0.5673 ± 0.0171	0.0406 ± 0.0053
188	0.5810 ± 0.0172	**0.6309 ± 0.0176**	0.2913 ± 0.0177	0.4873 ± 0.0239	0.0102 ± 0.0031

Table C.9: NDCG score with cutoff 10 (top-10 tags) for MTAT when no seed tags are given for the MFCC feature.

	mf	mlp	prior	rgcn	uniform
16	0.3945 ± 0.0157	0.5108 ± 0.0165	**0.5211 ± 0.0205**	0.5146 ± 0.0200	0.4524 ± 0.0159
32	0.2380 ± 0.0162	0.3874 ± 0.0123	**0.3886 ± 0.0142**	0.3853 ± 0.0165	0.1997 ± 0.0067
64	0.2021 ± 0.0080	0.3219 ± 0.0175	**0.3238 ± 0.0163**	0.2708 ± 0.0650	0.0956 ± 0.0090
128	0.1831 ± 0.0087	0.3014 ± 0.0135	**0.3028 ± 0.0149**	0.3008 ± 0.0141	0.0406 ± 0.0053
188	0.1270 ± 0.0136	**0.2933 ± 0.0176**	0.2913 ± 0.0177	0.2779 ± 0.0207	0.0102 ± 0.0031

Table C.10: NDCG score with cutoff 10 (top-10 tags) for MTAT when no seed tags are given for the random feature.

1 Seed tag given

	mf	mlp	prior	rgcn	uniform
16	0.7945 ± 0.0157	**0.8596 ± 0.0172**	0.5036 ± 0.0231	0.8091 ± 0.0139	0.4338 ± 0.0147
32	0.6505 ± 0.0119	**0.7499 ± 0.0140**	0.3790 ± 0.0168	0.7025 ± 0.0118	0.2807 ± 0.0097
64	0.6095 ± 0.0244	**0.6830 ± 0.0189**	0.3149 ± 0.0117	0.6052 ± 0.0185	0.1083 ± 0.0136
128	0.5294 ± 0.0189	**0.6196 ± 0.0126**	0.2868 ± 0.0108	0.5666 ± 0.0145	0.0536 ± 0.0051
188	0.5451 ± 0.0161	**0.6089 ± 0.0102**	0.2710 ± 0.0148	0.5131 ± 0.0159	0.0139 ± 0.0025

Table C.11: NDCG score with cutoff 10 (top-10 tags) for MTAT when 1 seed tag is given for the MFCC feature.

	mf	mlp	prior	rgcn	uniform
16	**0.7459 ± 0.0188**	0.5144 ± 0.0261	0.5036 ± 0.0231	0.7008 ± 0.0159	0.4338 ± 0.0147
32	**0.5923 ± 0.0206**	0.3812 ± 0.0208	0.3790 ± 0.0168	0.5560 ± 0.0141	0.2807 ± 0.0097
64	**0.5455 ± 0.0203**	0.3096 ± 0.0115	0.3149 ± 0.0117	0.4854 ± 0.0185	0.1083 ± 0.0136
128	**0.4916 ± 0.0198**	0.2862 ± 0.0112	0.2868 ± 0.0108	0.4363 ± 0.0201	0.0536 ± 0.0051
188	**0.4523 ± 0.0127**	0.2638 ± 0.0180	0.2710 ± 0.0148	0.4047 ± 0.0157	0.0139 ± 0.0025

Table C.12: NDCG score with cutoff 10 (top-10 tags) for MTAT when 1 seed tag is given for the random feature.

2 Seed tags given

	mf	mlp	prior	rgcn	uniform
16	0.8062 ± 0.0197	**0.8568 ± 0.0243**	0.4999 ± 0.0385	0.8415 ± 0.0230	0.4380 ± 0.0291
32	0.6897 ± 0.0175	**0.7364 ± 0.0179**	0.3640 ± 0.0249	0.7061 ± 0.0237	0.2934 ± 0.0175
64	0.5949 ± 0.0222	**0.6643 ± 0.0184**	0.3003 ± 0.0189	0.6382 ± 0.0210	0.0999 ± 0.0063
128	0.5751 ± 0.0188	**0.6327 ± 0.0136**	0.2739 ± 0.0110	0.5829 ± 0.0204	0.0560 ± 0.0042
188	0.4584 ± 0.0115	**0.6063 ± 0.0090**	0.2653 ± 0.0133	0.5633 ± 0.0105	0.0198 ± 0.0030

Table C.13: NDCG score with cutoff 10 (top-10 tags) for MTAT when 2 seed tags are given for the MFCC feature.

	mf	mlp	prior	rgcn	uniform
16	**0.8143 ± 0.0265**	0.4960 ± 0.0392	0.4999 ± 0.0385	0.7813 ± 0.0227	0.4380 ± 0.0291
32	**0.6767 ± 0.0216**	0.3563 ± 0.0232	0.3640 ± 0.0249	0.6235 ± 0.0262	0.2934 ± 0.0175
64	**0.6156 ± 0.0272**	0.2887 ± 0.0156	0.3003 ± 0.0189	0.5454 ± 0.0217	0.0999 ± 0.0063
128	**0.5504 ± 0.0241**	0.2744 ± 0.0104	0.2739 ± 0.0110	0.4718 ± 0.0224	0.0560 ± 0.0042
188	**0.5322 ± 0.0103**	0.2685 ± 0.0138	0.2653 ± 0.0133	0.4795 ± 0.0127	0.0198 ± 0.0030

Table C.14: NDCG score with cutoff 10 (top-10 tags) for MTAT when 2 seed tags are given for the random feature.

5 Seed tags given

	mf	mlp	prior	rgcn	uniform
16	0.7885 ± 0.1547	0.7353 ± 0.1847	0.5224 ± 0.0725	**0.8080 ± 0.1953**	0.4974 ± 0.1374
32	0.7043 ± 0.0427	0.6553 ± 0.0579	0.3159 ± 0.0429	**0.7234 ± 0.0366**	0.2476 ± 0.0287
64	**0.6881 ± 0.0401**	0.6092 ± 0.0261	0.2707 ± 0.0381	0.6206 ± 0.0466	0.0943 ± 0.0135
128	0.5759 ± 0.0283	0.5467 ± 0.0281	0.2282 ± 0.0234	**0.5836 ± 0.0271**	0.0697 ± 0.0143
188	**0.6056 ± 0.0168**	0.5731 ± 0.0126	0.2261 ± 0.0097	0.5620 ± 0.0138	0.0251 ± 0.0047

Table C.15: NDCG score with cutoff 10 (top-10 tags) for MTAT when 5 seed tags are given for the MFCC feature.

	mf	mlp	prior	rgcn	uniform
16	0.6629 ± 0.2639	0.5522 ± 0.1864	0.5224 ± 0.0725	**0.6928 ± 0.2263**	0.4974 ± 0.1374
32	**0.6899 ± 0.0481**	0.2764 ± 0.0610	0.3159 ± 0.0429	0.6588 ± 0.0545	0.2476 ± 0.0287
64	**0.6757 ± 0.0406**	0.2586 ± 0.0324	0.2707 ± 0.0381	0.5913 ± 0.0351	0.0943 ± 0.0135
128	**0.6299 ± 0.0305**	0.2221 ± 0.0278	0.2282 ± 0.0234	0.5079 ± 0.0409	0.0697 ± 0.0143
188	**0.6010 ± 0.0145**	0.2260 ± 0.0097	0.2261 ± 0.0097	0.5198 ± 0.0209	0.0251 ± 0.0047

Table C.16: NDCG score with cutoff 10 (top-10 tags) for MTAT when 5 seed tags are given for the random feature.

D

Embedding Similarity Results

This appendix contains the embedding similarity results for both MSD and MTAT. Each table contains per dataset subset the average and standard deviation of the 10 folds. The model with the highest score per subset is marked in bold.

Million Song Dataset

	mf	mlp	rgcn
16	**0.6620 ± 0.0028**	0.6139 ± 0.0065	0.6143 ± 0.0198
32	**0.5600 ± 0.0056**	0.5340 ± 0.0148	0.5509 ± 0.0146
64	**0.5385 ± 0.0029**	0.5096 ± 0.0103	0.5037 ± 0.0094
128	**0.5239 ± 0.0013**	0.4464 ± 0.0044	0.4858 ± 0.0037
256	**0.5059 ± 0.0037**	0.4144 ± 0.0026	0.4644 ± 0.0069
512	**0.5064 ± 0.0026**	0.4023 ± 0.0034	0.4660 ± 0.0048
1024	**0.4714 ± 0.0004**	0.4015 ± 0.0020	0.4520 ± 0.0035

Table D.1: NDCG score with cutoff 10 for embedding similarity of models trained with the MSD dataset when an MFCC feature is given.

	mf	mlp	rgcn
16	**0.6590 ± 0.0064**	0.6259 ± 0.0148	0.6313 ± 0.0106
32	**0.5751 ± 0.0023**	0.4746 ± 0.0230	0.5700 ± 0.0111
64	**0.5508 ± 0.0017**	0.4151 ± 0.0107	0.5113 ± 0.0098
128	0.5259 ± 0.0012	0.3803 ± 0.0087	**0.5263 ± 0.0045**
256	**0.4918 ± 0.0055**	0.3314 ± 0.0077	0.4695 ± 0.0068
512	**0.5088 ± 0.0015**	0.3161 ± 0.0069	0.4700 ± 0.0052
1024	**0.4831 ± 0.0019**	0.3046 ± 0.0029	0.4526 ± 0.0027

Table D.2: NDCG score with cutoff 10 for embedding similarity of models trained with the MSD dataset when a random feature is given.

MagnaTagATune

	mf	mlp	rgcn
16	0.6962 ± 0.0262	0.7052 ± 0.0258	**0.7144 ± 0.0130**
32	0.5567 ± 0.0079	0.5470 ± 0.0153	**0.5633 ± 0.0075**
64	**0.5844 ± 0.0042**	0.4699 ± 0.0108	0.5628 ± 0.0151
128	0.4908 ± 0.0059	0.4582 ± 0.0038	**0.4968 ± 0.0048**
188	0.4697 ± 0.0037	0.4318 ± 0.0061	**0.4810 ± 0.0050**

Table D.3: NDCG score with cutoff 10 for embedding similarity of models trained with the MTAT dataset when an MFCC feature is given.

	mf	mlp	rgcn
16	0.7044 ± 0.0273	0.6946 ± 0.0194	**0.7114 ± 0.0080**
32	**0.5656 ± 0.0085**	0.5212 ± 0.0293	0.5605 ± 0.0124
64	**0.5801 ± 0.0030**	0.4086 ± 0.0144	0.5627 ± 0.0118
128	**0.5052 ± 0.0018**	0.3779 ± 0.0173	0.4617 ± 0.0113
188	**0.4678 ± 0.0043**	0.3645 ± 0.0137	0.4462 ± 0.0074

Table D.4: NDCG score with cutoff 10 for embedding similarity of models trained with the random dataset when a random feature is given.

E

Song predictions

This appendix contains the predicted tags for a selection of songs of dataset subset size 128 for both MSD and MTAT. Seed tags have been excluded from the output. Tags in *italic* are tags from the 50% least frequent tags in the set. Tags in **bold** are tags that occur in the ground truth and are therefore correctly predicted. Behind each model, the NDCG value with cutoff 10 for that particular column is given.

E.1. MSD 128 5 seeds

Table E.1: **Stockholm Syndrome - Muse**
Spotify: https://open.spotify.com/track/5VVWgWH8HFLAtM8lbttvn9

ground truth	input seed	mf (0.863)	graph (0.931)	mlp (0.936)
rock	indie	**alternative**	**alternative**	**rock**
pop	electronic	**experimental**	**experimental**	**alternative**
alternative	indie rock	**rock**	**alternative rock**	**alternative rock**
favorites	Progressive rock	**alternative rock**	**oos**	**favorites**
Love	amazing	**oos**	**Awesome**	**oos**
oos		***psychedelic***	**rock**	**Awesome**
alternative rock		indie pop	**electronica**	**pop**
dance		**Awesome**	seen live	**hard rock**
beautiful		**electronica**	**favorites**	**metal**
metal		seen live	**british**	american
Awesome				
male vocalists				
electronica				
chill				
british				
hard rock				
cool				
Favorite				
experimental				
Favourites				
punk rock				
hardcore				
psychedelic				
favorite songs				
male vocalist				
epic				
emo				
2000s				
Favourite Songs				
Love it				
nice				
UK				
britpop				
Progressive				
great				
Progressive metal				
FUCKING AWESOME				

Table E.2: **Tainted Love - Soft Cell**
Spotify: https://open.spotify.com/track/58E1XVmZTODC67YNjneuXM

ground truth	input seed	mf (0.934)	graph (0.845)	mlp (0.931)
rock	indie	**electronic**	**dance**	**pop**
pop	Love	**electronica**	**electronic**	**dance**
alternative	new wave	**dance**	**pop**	**rock**
electronic	techno	**pop**	**alternative**	**male vocalists**
favorites	synthpop	**electro**	00s	**party**
alternative rock		**alternative**	**party**	**electronic**
dance		**80s**	**catchy**	**favorites**
Awesome		**british**	90s	00s
male vocalists		indie rock	**fun**	**80s**
electronica		***post-punk***	**electronica**	**catchy**
classic rock				
80s				
british				
cool				
Favorite				
Favourites				
oldies				
sexy				
party				
catchy				
electro				
favourite				
fun				
Soundtrack				
loved				
cover				
classic				
good				
Favourite Songs				
post-punk				
UK				
britpop				
dark				

Table E.3: **Runnin' From The Law - L7**
Spotify: https://open.spotify.com/track/6VXBffSoy4QrgwOhINLH3A

ground truth	input seed	mf (0.699)	graph (0.727)	mlp (0.874)
rock	alternative	**rock**	**rock**	**rock**
female vocalists	indie	**punk**	**punk**	**punk**
favorites	alternative rock	**indie rock**	**indie rock**	90s
metal	hard rock	seen live	seen live	**metal**
indie rock	punk rock	hardcore	**metal**	**80s**
80s		*emo*	**favorites**	**heavy metal**
punk		**metal**	Awesome	**favorites**
heavy metal		90s	*emo*	**indie rock**
		favorites	*post-punk*	*post-punk*
		00s	british	seen live

Table E.4: **Rosie - Joan Armatrading**
Spotify: https://open.spotify.com/track/4yMdCBVURkXYYNtY53H1FY

ground truth	input seed	mf (0.458)	graph (0.410)	mlp (0.411)
rock	Love	pop	**soul**	pop
female vocalists	80s	**female vocalists**	rnb	**female vocalists**
singer-songwriter	70s	**soul**	*spanish*	**rock**
soul	female	**oldies**	**oldies**	dance
oldies	reggae	**rock**	pop	00s
happy		classic rock	classic rock	*spanish*
latin		favorites	*60s*	female vocalist
smooth		female vocalist	***latin***	***latin***
soft rock		rnb	country	**singer-songwriter**
		dance	male vocalists	favorites

Table E.5: **How We Became - Jeremy Warmsley**
Youtube: https://www.youtube.com/watch?v=rW_DbW6TD84

ground truth	input seed	mf (0.296)	graph (0.141)	mlp (0.765)
electronic	rock	alternative	alternative	**electronic**
singer-songwriter	pop	**singer-songwriter**	Love	**electronica**
electronica	indie	indie rock	favorites	dance
	folk	female vocalists	00s	alternative
	indie pop	acoustic	british	00s
		alternative rock	female vocalists	electro
		00s	beautiful	chillout
		seen live	male vocalists	Love
		Mellow	**singer-songwriter**	favorites
		british	indie rock	experimental

Table E.6: **Must Be Doin' Somethin' Right - Billy Currington**
Spotify: https://open.spotify.com/track/6TefPQDooc88BNALpu4U87

ground truth	input seed	mf (0.229)	graph (0.107)	mlp (0.252)
beautiful	Love	favorites	male vocalists	rock
sexy	oldies	male vocalists	pop	pop
country	easy listening	pop	rock	male vocalists
Favourite Songs	loved	american	singer-songwriter	favorites
love songs	favorite songs	**beautiful**	alternative	**beautiful**
		Favorite	acoustic	**country**
		favourite	Mellow	alternative
		60s	**beautiful**	*Ballad*
		Mellow	favorites	00s
		country	indie	sad

E.2. MTAT 128 5 seeds

Table E.7: **Rattling Sabers - Rapoon**
Spotify: https://open.spotify.com/track/2HLxryL3H39oDkmaFMnjn7

ground truth	input seed	mf (0.264)	graph (0.000)	mlp (0.798)
ambient	guitar	slow	classical	**ambient**
no vocal	strings	classical	piano	slow
	quiet	soft	classic	soft
	solo	**ambient**	slow	electronic
	foreign	*noise*	soft	new age
		opera	cello	synth
		vocal	*classical guitar*	*dark*
		piano	harp	weird
		female	*lute*	**no vocal**
		dark	opera	*low*

Table E.8: **Goodbye Caramel - Norine Braun**
Spotify: https://open.spotify.com/track/49fooWM4zWKpGnXtVoEKUi

ground truth	input seed	mf (0.422)	graph (0.441)	mlp (0.593)
guitar	female	singing	**female vocal**	rock
pop	woman	vocal	singing	**pop**
female vocal	female vocals	**guitar**	female voice	**guitar**
country	singer	rock	vocal	vocals
modern	female singer	**female vocal**	vocals	vocal
		pop	*woman singing*	singing
		vocals	*women*	male
		folk	voice	**country**
		male	**pop**	**female vocal**
		fast	*foreign*	loud

Table E.9: **Vdol Po Rechenke (Russia) - Kitka**
Spotify: https://open.spotify.com/track/5kN4hfZoaNih6OQJBZDvpi

ground truth	input seed	mf (0.807)	graph (0.807)	mlp (0.807)
female	opera	**female**	**female**	**female**
woman singing	singing	female vocal	female vocals	female vocals
	woman	vocal	vocal	female vocal
	choir	choral	female vocal	vocal
	female voice	female vocals	vocals	*women*
		vocals	*women*	vocals
		women	choral	voice
		woman singing	***woman singing***	***woman singing***
		chorus	*chorus*	*female singer*
		singer	voice	choral

Table E.10: **Spinning - Atomic Opera**
Spotify: https://open.spotify.com/track/1SkNhPrk8NeLctN3BsDN3R

ground truth	input seed	mf (0.414)	graph (0.264)	mlp (0.501)
vocal	rock	*hard rock*	*hard rock*	guitar
male	male vocal	loud	loud	loud
	male voice	drums	*heavy metal*	drums
	metal	guitar	**male**	**vocal**
	hard	**male**	*heavy*	**male**
		male vocals	male vocals	vocals
		man	*punk*	singing
		electric guitar	man	fast
		heavy	*man singing*	techno
		vocal	*electric guitar*	electronic

Table E.11: **Sonata 2 (Minuetto I) - Ensemble Mirable**
Spotify: https://open.spotify.com/track/4iOyWIpSKUQncdVTKfRpll

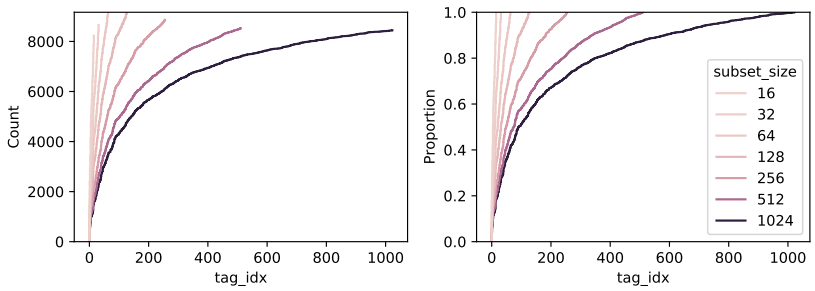
ground truth	input seed	mf (0.832)	graph (0.832)	mlp (0.832)
classical	guitar	**violin**	**classical**	**classical**
slow	strings	**classical**	**violin**	**violin**
violin	soft	**slow**	**slow**	**slow**
low	classic	harpsichord	solo	piano
	cello	violins	harpsichord	violins
		piano	string	harpsichord
		string	piano	string
		no vocal	violins	indian
		solo	harp	sitar
		instrumental	no vocals	solo

Table E.12: **Pauls Galiarde - Jacob Heringman**
Spotify: https://open.spotify.com/album/72xGnfZFlATZLZKy3on1l8

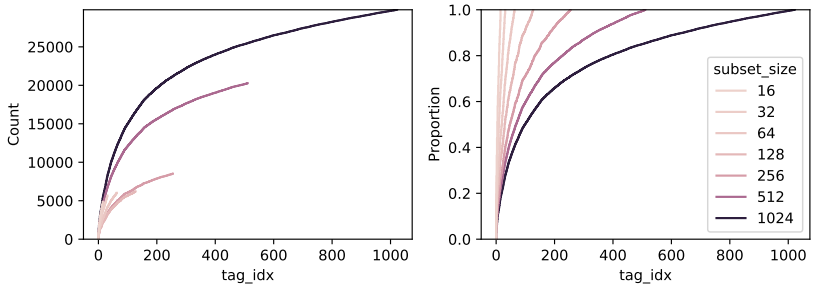
ground truth	input seed	mf (0.967)	graph (0.894)	mlp (0.967)
guitar	strings	**guitar**	**classical**	**guitar**
classical	quiet	**slow**	***classical guitar***	**slow**
slow	solo	**classical**	harp	**classical**
soft	string	harp	piano	piano
classical guitar	lute	***classical guitar***	classic	**soft**
		piano	**soft**	harp
		soft	**slow**	***classical guitar***
		no vocal	**guitar**	*acoustic*
		classic	*acoustic*	flute
		acoustic	cello	classic

F

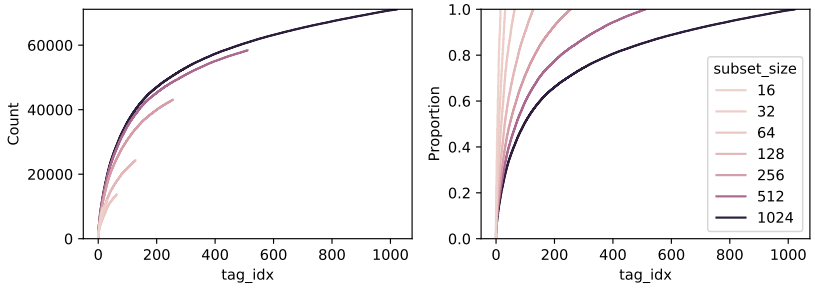
Supporting figures



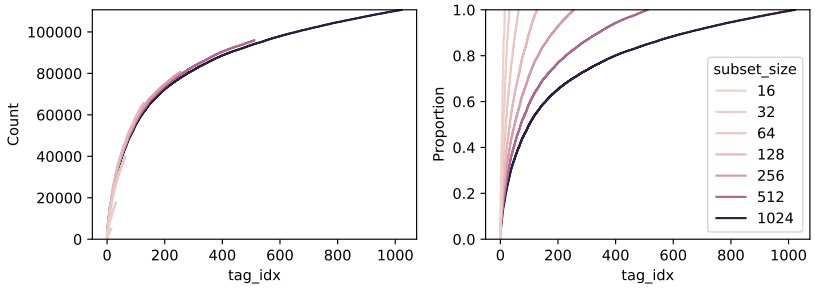
(a) 0 seeds



(b) 1 seed

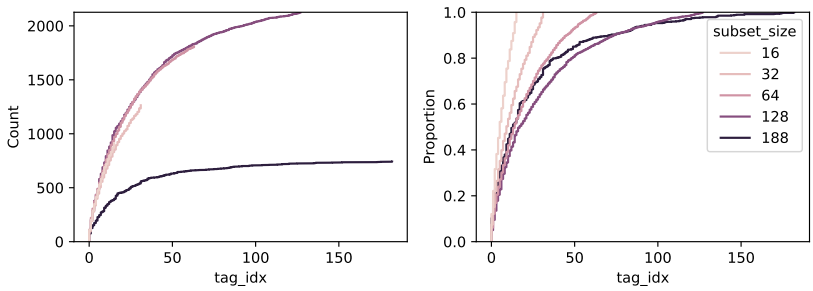


(c) 2 seeds

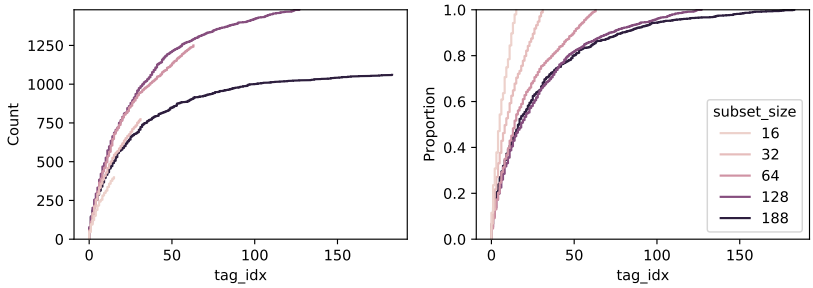


(d) 5 seeds

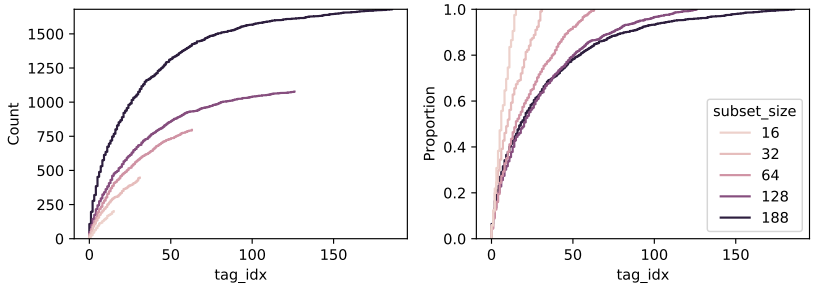
Figure F.1: Distribution of tag assignments for the first fold of MSD. Other folds are distributed similarly. *Count* refers to the cumulative number of tags assigned at the given *tag_idx*. Similarly, *Proportion* refers to the cumulative percentage.



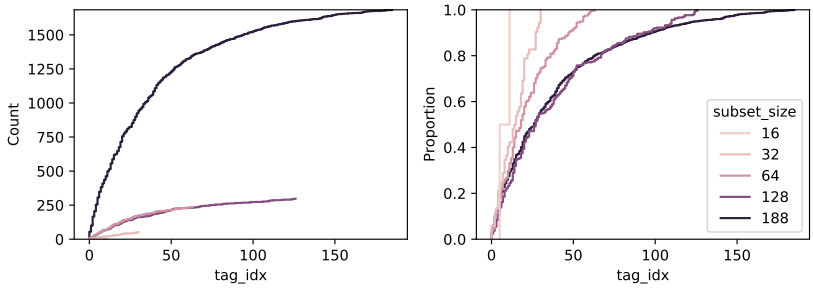
(a) 0 seeds



(b) 1 seed



(c) 2 seeds



(d) 5 seeds

Figure F.2: Distribution of tag assignments for the first fold of MTAT. Other folds are distributed similarly. *Count* refers to the cumulative number of tags assigned at the given *tag_idx*. Similarly, *Proportion* refers to the cumulative percentage.

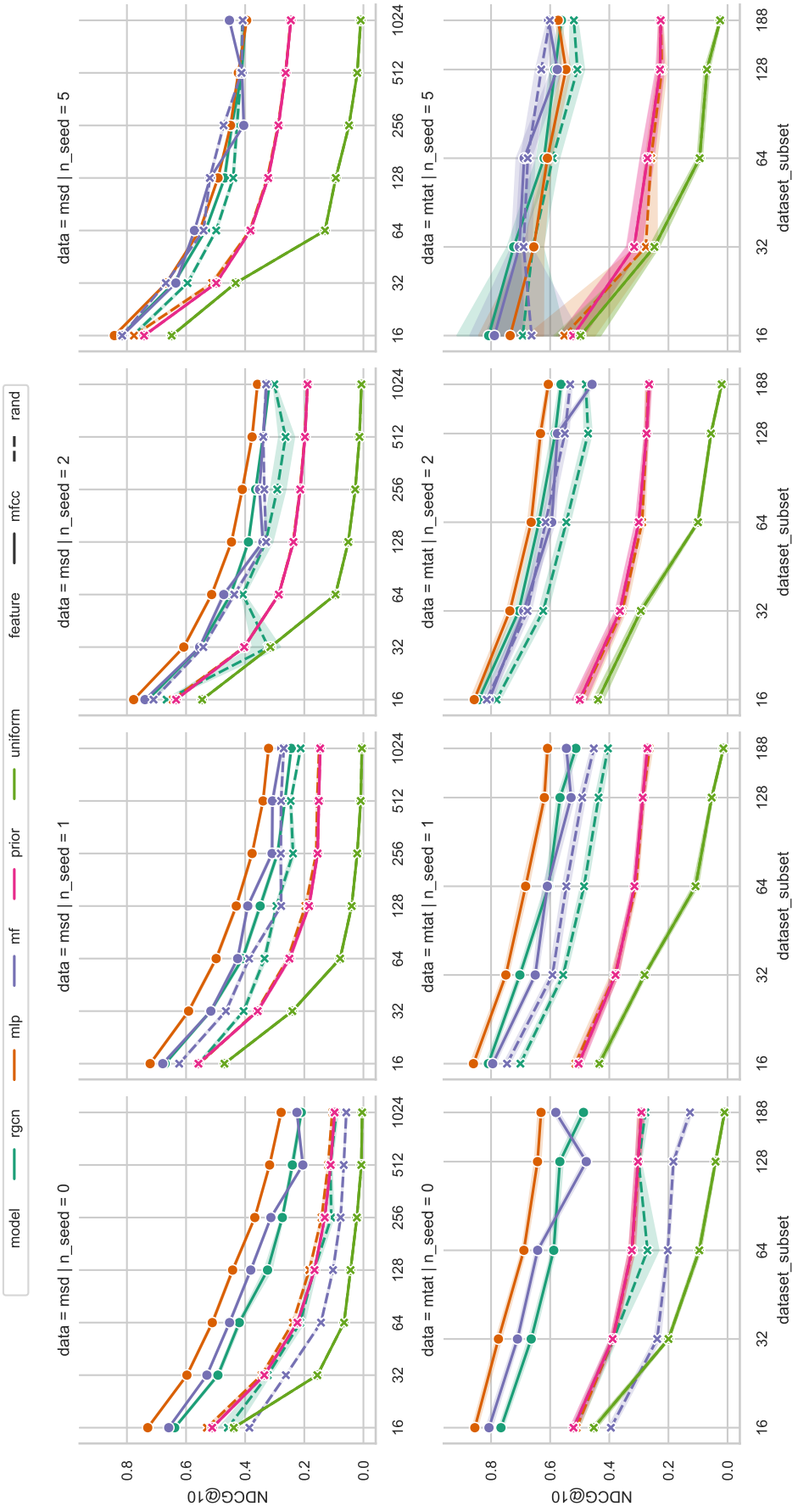


Figure F.3: NDCG score, including all tags in the subset vocabulary.

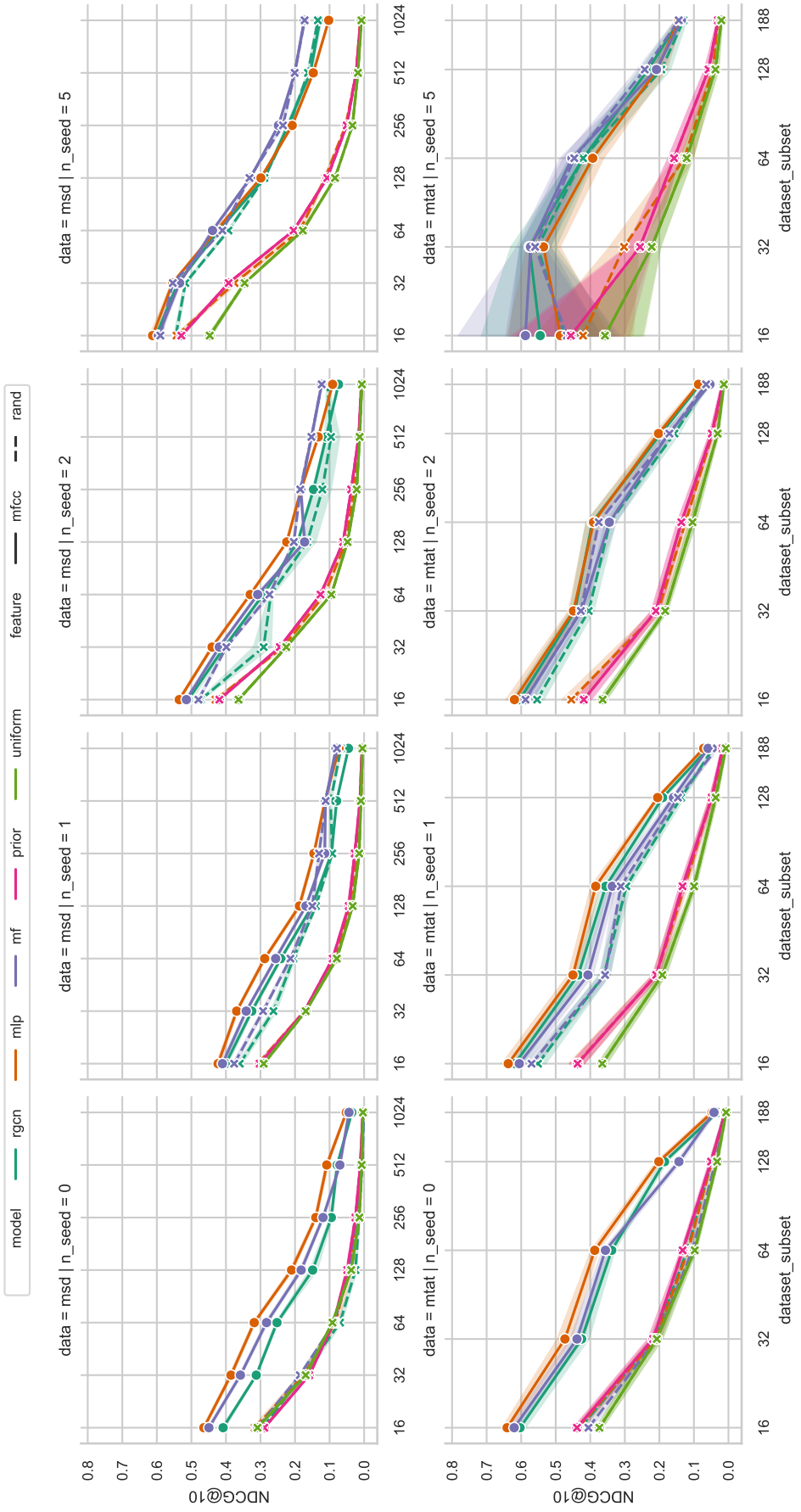


Figure F.4: NDCG score, top 50% of the most frequent tags are omitted, leaving only tags from the long-tail.

Acronyms

AARMS Attribute-Aware Recommender ModelS. 34

ALS Alternating Least Squares. 32

BPR Bayesian Personalized Ranking. 32, 33

CAL500 Computer Audition Lab 500. 12

CBA Codeword Bernouilli Average. 12

CNN Convolutional Neural Network. 13, 14, 16

CRNN Convolutional Recurrent Neural Network. 14

CTR Collaborative Topic Regression. 17, 18

DBN Deep Belief Networks. 13, 16

DFT Discrete Fourier Transform. 13

DGL Deep Graph Library. 34, 49

DNN Deep Neural Networks. 16

FFNN Feed Forward Neural Network. 16

FPR False Positive Rate. 33

GCN Graph Convolutional Network. 24, 25, 34

GMM Gaussian Mixture Model. 11, 12

KNN K-Nearest Neighbour. 11

LDA Latent Dirichlet Allocation. 12, 13, 18

MF Matrix Factorization. 16–18, 24–26, 34–37, 39, 40, 44, 48

MFCC Mel-Frequency Cepstral Coefficients. 11–13, 24, 27, 37, 38, 40, 42–45, 48, 49, 59–74

MGP Music Genome Project. 8, 17, 22

MIR Music Information Retrieval. 1, 3, 7–9, 11, 16, 17, 21, 22, 24, 49

MLP Multi-Layer Perceptron. 13, 16, 24, 27, 32, 34–37, 40, 42–44, 47–49

MSD Million Song Dataset. 2, 3, 9, 10, 13, 14, 21, 22, 24, 27–31, 39, 40, 43, 47, 49, 59–61, 65–68, 73, 75, 82

MTAT MagnaTagATune. 2, 13, 21, 27, 29–31, 37, 39, 40, 44, 47, 49, 59, 62–65, 69–75, 83

NDCG Normalized Discounted Cumulative Gain. 33, 37, 44

R-GCN Relational Graph Convolutional Network. 25, 26, 32, 37, 39, 40, 44, 47, 48

RNN Recurrent Neural Network. 14

ROC-AUC Receiver Operating Characteristic Area Under Curve. 33, 37, 38, 44

STFT Short-Time Fourier Transform. 11

SVM Support Vector Machine. 12–14

TPR True Positive Rate. 33

WMF Weighted Matrix Factorization. 14, 17

Bibliography

[1] About the music genome project®. URL https://www.pandora.com/about/mgp. Accessed: 2021-01-21.

[2] Folksonomy coinage and definition, February 2007. URL http://www.vanderwal.net/folksonomy.html. Accessed: 2020-10-10.

[3] Pandora's long strange trip, 2007. URL https://www.inc.com/magazine/20071001/pandoras-long-strange-trip.html. Accessed: 2021-01-21.

[4] The song decoders, October 2009. URL https://www.nytimes.com/2009/10/18/magazine/18Pandora-t.html. Accessed: 2021-01-21.

[5] Million song dataset, October 2011. URL http://millionsongdataset.com/lastfm/. Accessed: 2020-10-10.

[6] The magnatagatune dataset, June 2013. URL http://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset. Accessed: 2020-10-10.

[7] *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, 2017. IEEE. ISBN 978-1-5090-4117-6. URL https://ieeexplore.ieee.org/xpl/conhome/7943262/proceeding.

[8] Juan Pablo Bello, Elaine Chew, and Douglas Turnbull, editors. *ISMIR 2008, 9th International Conference on Music Information Retrieval, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*, 2008. ISBN 978-0-615-24849-3.

[9] Thierry Bertin-Mahieux, Douglas Eck, François Maillet, and Paul Lamere. Autotagger: A model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*, 37(2):115–135, 2008. doi: 10.1080/09298210802479250. URL https://doi.org/10.1080/09298210802479250.

[10] Thierry Bertin-Mahieux, Douglas Eck, and Michael Mandel. Automatic tagging of audio: The state-of-the-art. *Machine Audition: Principles, Algorithms and Systems*, pages 334–352, 01 2010. doi: 10.4018/978-1-61520-919-4.ch014.

[11] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In Klapuri and Leider [50], pages 591–596. ISBN 978-0-615-54865-4. URL http://ismir2011.ismir.net/papers/OS6-1.pdf.

[12] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003. URL http://jmlr.org/papers/v3/blei03a.html.

[13] Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter, and Xavier Serra. The mtg-jamendo dataset for automatic music tagging. In *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML 2019)*, Long Beach, CA, United States, 2019. URL http://hdl.handle.net/10230/42015.

[14] Joseph K. Bradley and Robert E. Schapire. Filterboost: Regression and classification on large datasets. In Platt et al. [64], pages 185–192. URL https://proceedings.neurips.cc/paper/2007/hash/072b030ba126b2f4b2374f342be9ed44-Abstract.html.

[15] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. Recsys challenge 2018: automatic music playlist continuation. In Sole Pera, Michael D. Ekstrand, Xavier Amatriain, and John O'Donovan, editors, *Proceedings of the 12th ACM Conference on Recommender Systems,*

RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018, pages 527–528. ACM, 2018. ISBN 978-1-4503-5901-6. doi: 10.1145/3240323.3240342. URL https://doi.org/10.1145/3240323.3240342.

[16] Keunwoo Choi, György Fazekas, and Mark B. Sandler. Automatic tagging using deep convolutional neural networks. In Michael I. Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, editors, *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, pages 805–811, 2016. ISBN 978-0-692-75506-8. URL https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/07/009_Paper.pdf.

[17] Keunwoo Choi, György Fazekas, Mark B. Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017* DBL [7], pages 2392–2396. ISBN 978-1-5090-4117-6. doi: 10.1109/ICASSP.2017.7952585. URL https://doi.org/10.1109/ICASSP.2017.7952585.

[18] Keunwoo Choi, György Fazekas, Mark B. Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks. In Cunningham et al. [20], pages 141–149. ISBN 978-981-11-5179-8. URL https://ismir2017.smcnus.org/wp-content/uploads/2017/10/12_Paper.pdf.

[19] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark B. Sandler. The effects of noisy labels on deep convolutional neural networks for music tagging. *IEEE Trans. Emerg. Top. Comput. Intell.*, 2(2):139–149, 2018. doi: 10.1109/TETCI.2017.2771298. URL https://doi.org/10.1109/TETCI.2017.2771298.

[20] Sally Jo Cunningham, Zhiyao Duan, Xiao Hu, and Douglas Turnbull, editors. *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, 2017. ISBN 978-981-11-5179-8.

[21] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980. doi: 10.1109/TASSP.1980.1163420.

[22] Alceu de Souza Britto Jr., Fabien Gouyon, and Simon Dixon, editors. *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013, Curitiba, Brazil, November 4-8, 2013*, 2013. ISBN 978-0-615-90065-0.

[23] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. FMA: A dataset for music analysis. In Cunningham et al. [20], pages 316–323. ISBN 978-981-11-5179-8. URL https://ismir2017.smcnus.org/wp-content/uploads/2017/10/75_Paper.pdf.

[24] Sander Dieleman and Benjamin Schrauwen. Multiscale approaches to music audio feature learning. In de Souza Britto Jr. et al. [22], pages 3–8. ISBN 978-0-615-90065-0. URL http://www.ppgia.pucpr.br/ismir2013/wp-content/uploads/2013/09/69_Paper.pdf.

[25] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 6964–6968. IEEE, 2014. doi: 10.1109/ICASSP.2014.6854950. URL https://doi.org/10.1109/ICASSP.2014.6854950.

[26] Sander Dieleman, Philemon Brakel, and Benjamin Schrauwen. Audio-based music classification with a pretrained convolutional network. In Klapuri and Leider [50], pages 669–674. ISBN 978-0-615-54865-4. URL http://ismir2011.ismir.net/papers/PS6-3.pdf.

[27] Simon Dixon, David Bainbridge, and Rainer Typke, editors. *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007, Vienna, Austria, September 23-27, 2007*, 2007. Austrian Computer Society. ISBN 978-3-85403-218-2.

[28] Douglas Eck, Thierry Bertin-Mahieux, and Paul Lamere. Autotagging music using supervised machine learning. In Dixon et al. [27], pages 367–368. ISBN 978-3-85403-218-2. URL http://ismir2007.ismir.net/proceedings/ISMIR2007_p367_eck.pdf.

[29] Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green. Automatic generation of social tags for music recommendation. In Platt et al. [64], pages 385–392. URL http://papers.nips.cc/paper/3370-automatic-generation-of-social-tags-for-music-recommendation.

[30] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In Lorenza Saitta, editor, *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pages 148–156. Morgan Kaufmann, 1996. ISBN 1-55860-419-7.

[31] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017* DBL [7], pages 776–780. ISBN 978-1-5090-4117-6. doi: 10.1109/ICASSP.2017.7952261. URL https://doi.org/10.1109/ICASSP.2017.7952261.

[32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[33] Fabien Gouyon, Anssi Klapuri, Simon Dixon, M. Alonso, George Tzanetakis, C. Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Trans. Speech Audio Process.*, 14(5):1832–1844, 2006. doi: 10.1109/TSA.2005.858509. URL https://doi.org/10.1109/TSA.2005.858509.

[34] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. ACM, 2016. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939754. URL https://doi.org/10.1145/2939672.2939754.

[35] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In J. Stephen Downie and Remco C. Veltkamp, editors, *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*, pages 339–344. International Society for Music Information Retrieval, 2010. ISBN 978-90-393-53813. URL http://ismir2010.ismir.net/proceedings/ismir2010-58.pdf.

[36] Philippe Hamel, Simon Lemieux, Yoshua Bengio, and Douglas Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In Klapuri and Leider [50], pages 729–734. ISBN 978-0-615-54865-4. URL http://ismir2011.ismir.net/papers/PS6-13.pdf.

[37] Philippe Hamel, Matthew E. P. Davies, Kazuyoshi Yoshii, and Masataka Goto. Transfer learning in mir: Sharing learned latent representations for music audio classification and similarity. In de Souza Britto Jr. et al. [22], pages 9–14. ISBN 978-0-615-90065-0. URL http://www.ppgia.pucpr.br/ismir2013/wp-content/uploads/2013/09/76_Paper.pdf.

[38] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74, 2017. URL http://sites.computer.org/debull/A17sept/p52.pdf.

[39] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

[40] Mikael Henaff, Kevin Jarrett, Koray Kavukcuoglu, and Yann LeCun. Unsupervised learning of sparse features for scalable audio classification. In Klapuri and Leider [50], pages 681–686. ISBN 978-0-615-54865-4. URL http://ismir2011.ismir.net/papers/PS6-5.pdf.

[41] Keiji Hirata, George Tzanetakis, and Kazuyoshi Yoshii, editors. *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, 2009. International Society for Music Information Retrieval. ISBN 978-0-9813537-0-8.

[42] Matthew D. Hoffman, David M. Blei, and Perry R. Cook. Easy as CBA: A simple probabilistic model for tagging music. In Hirata et al. [41], pages 369–374. ISBN 978-0-9813537-0-8. URL http://ismir2009.ismir.net/proceedings/OS5-2.pdf.

[43] Helge Homburg, Ingo Mierswa, Bülent Möller, Katharina Morik, and Michael Wurst. A benchmark dataset for audio classification and clustering. In *ISMIR 2005, 6th International Conference on Music Information Retrieval, London, UK, 11-15 September 2005, Proceedings*, pages 528–531, 2005. URL http://ismir2005.ismir.net/proceedings/2117.pdf.

[44] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 263–272. IEEE Computer Society, 2008. ISBN 978-0-7695-3502-9. doi: 10.1109/ICDM.2008.22. URL https://doi.org/10.1109/ICDM.2008.22.

[45] Eric J. Humphrey, Juan Pablo Bello, and Yann LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In Fabien Gouyon, Perfecto Herrera, Luis Gustavo Martins, and Meinard Müller, editors, *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, pages 403–408. FEUP Edições, 2012. ISBN 978-972-752-144-9. URL http://ismir2012.ismir.net/event/papers/403-ismir-2012.pdf.

[46] Melvyn J. Hunt, Matthew Lennig, and Paul Mermelstein. Experiments in syllable-based recognition of continuous speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '80, Denver, Colorado, USA, April 9-11, 1980*, pages 880–883. IEEE, 1980. doi: 10.1109/ICASSP.1980.1170934. URL https://doi.org/10.1109/ICASSP.1980.1170934.

[47] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002. doi: 10.1145/582415.582418. URL http://doi.acm.org/10.1145/582415.582418.

[48] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *CoRR*, abs/1611.07308, 2016. URL http://arxiv.org/abs/1611.07308.

[49] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

[50] Anssi Klapuri and Colby Leider, editors. *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, 2011. University of Miami. ISBN 978-0-615-54865-4. URL http://ismir2011.ismir.net/.

[51] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263. URL https://doi.org/10.1109/MC.2009.263.

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

[53] Paul Lamere and Elias Pampalk. Social tags and music information retrieval. In Bello et al. [8], page 24. ISBN 978-0-615-24849-3.

[54] Edith Law and Luis von Ahn. Input-agreement: a new mechanism for collecting data using human computation games. In Dan R. Olsen Jr., Richard B. Arthur, Ken Hinckley, Meredith Ringel Morris, Scott E. Hudson, and Saul Greenberg, editors, *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009*, pages 1197–1206. ACM, 2009. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1518881. URL https://doi.org/10.1145/1518701.1518881.

[55] Edith Law, Kris West, Michael I. Mandel, Mert Bay, and J. Stephen Downie. Evaluation of algorithms using games: The case of music tagging. In Hirata et al. [41], pages 387–392. ISBN 978-0-9813537-0-8. URL http://ismir2009.ismir.net/proceedings/OS5-5.pdf.

[56] Edith Law, Burr Settles, and Tom M. Mitchell. Learning to tag from open vocabulary labels. In José L. Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part II*, volume 6322 of *Lecture Notes in Computer Science*, pages 211–226. Springer, 2010. ISBN 978-3-642-15882-7. doi: 10.1007/978-3-642-15883-4_14. URL https://doi.org/10.1007/978-3-642-15883-4_14.

[57] Tom L. H. Li, Antoni B. Chan, and Andy H. W. Chun. Automatic musical pattern feature extraction using convolutional neural network. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2010, IMECS 2010*, pages 546–550, 2010. ISBN 9789881701282. International MultiConference of Engineers and Computer Scientists 2010, IMECS 2010 ; Conference date: 17-03-2010 Through 19-03-2010.

[58] Dawen Liang, Minshu Zhan, and Daniel P. W. Ellis. Content-aware collaborative music recommendation using pre-trained neural networks. In Meinard Müller and Frans Wiering, editors, *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 295–301, 2015. ISBN 978-84-606-8853-2. URL http://ismir2015.uma.es/articles/290_Paper.pdf.

[59] Michael I. Mandel and Daniel P. W. Ellis. A web-based game for collecting music metadata. In Dixon et al. [27], pages 365–366. ISBN 978-3-85403-218-2. URL http://ismir2007.ismir.net/proceedings/ISMIR2007_p365_mandel.pdf.

[60] Brian McFee, Matt McVicar, Stefan Balke, Vincent Lostanlen, Carl Thomé, Colin Raffel, Dana Lee, Kyungyun Lee, Oriol Nieto, Frank Zalkow, Dan Ellis, Eric Battenberg, Ryuichi Yamamoto, Josh Moore, Ziyao Wei, Rachel Bittner, Keunwoo Choi, nullmightybofo, Pius Friesch, Fabian-Robert Stöter, Thassilo, Matt Vollrath, Siddhartha Kumar Golu, nehz, Simon Waloschek, Seth, Rimvydas Naktinis, Douglas Repetto, Curtis "Fjord" Hawthorne, and CJ Carr. librosa/librosa: 0.6.3, February 2019. URL https://doi.org/10.5281/zenodo.2564164.

[61] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[62] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[63] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani,

editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 701–710. ACM, 2014. ISBN 978-1-4503-2956-9. doi: 10.1145/2623330.2623732. URL https://doi.org/10.1145/2623330.2623732.

[64] John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors. *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, 2008. Curran Associates, Inc. URL http://papers.nips.cc/book/advances-in-neural-information-processing-systems-20-2007.

[65] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *14th International Workshop on Content-Based Multimedia Indexing, CBMI 2016, Bucharest, Romania, June 15-17, 2016*, pages 1–6. IEEE, 2016. ISBN 978-1-4673-8695-1. doi: 10.1109/CBMI.2016.7500246. URL https://doi.org/10.1109/CBMI.2016.7500246.

[66] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik M. Schmidt, Andreas F. Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In Emilia Gómez, Xiao Hu, Eric Humphrey, and Emmanouil Benetos, editors, *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 637–644, 2018. ISBN 978-2-9540351-2-3. URL http://ismir2018.ircam.fr/doc/pdfs/191_Paper.pdf.

[67] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In Jeff A. Bilmes and Andrew Y. Ng, editors, *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 452–461. AUAI Press, 2009. URL https://www.auai.org/uai2009/papers/UAI2009_0139_48141db02b9f0b02bc7158819ebfa2c7.pdf.

[68] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In Platt et al. [64], pages 1257–1264. URL https://proceedings.neurips.cc/paper/2007/hash/d7322ed717dedf1eb4e6e52a37ea7bcd-Abstract.html.

[69] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. *CoRR*, abs/1703.06103, 2017. URL http://arxiv.org/abs/1703.06103.

[70] Klaus Seyerlehner, Markus Schedl, Reinhard Sonnleitner, David Hauger, and Bogdan Ionescu. From improved auto-taggers to improved music similarity measures. In Andreas Nürnberger, Sebastian Stober, Birger Larsen, and Marcin Detyniecki, editors, *Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation, 10th International Workshop, AMR 2012, Copenhagen, Denmark, October 24-25, 2012, Revised Selected Papers*, volume 8382 of *Lecture Notes in Computer Science*, pages 193–202. Springer, 2012. ISBN 978-3-319-12092-8. doi: 10.1007/978-3-319-12093-5_11. URL https://doi.org/10.1007/978-3-319-12093-5_11.

[71] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, 2014.

[72] Derek Tingle, Youngmoo E. Kim, and Douglas Turnbull. Exploring automatic music annotation with "acoustically-objective" tags. In James Ze Wang, Nozha Boujemaa, Nuria Oliver Ramirez, and Apostol Natsev, editors, *Proceedings of the 11th ACM SIGMM International Conference on Multimedia Information Retrieval, MIR 2010, Philadelphia, Pennsylvania, USA, March 29-31, 2010*, pages 55–62. ACM, 2010. ISBN 978-1-60558-815-5. doi: 10.1145/1743384.1743400. URL https://doi.org/10.1145/1743384.1743400.

[73] Douglas Turnbull, Ruoran Liu, Luke Barrington, and Gert R. G. Lanckriet. A game-based approach for collecting semantic annotations of music. In Dixon et al. [27], pages 535–538. ISBN 978-3-85403-218-2. URL http://ismir2007.ismir.net/proceedings/ISMIR2007_p535_turnbull.pdf.

[74] Douglas Turnbull, Luke Barrington, and Gert R. G. Lanckriet. Five approaches to collecting tags for music. In Bello et al. [8], pages 225–230. ISBN 978-0-615-24849-3. URL http://ismir2008.ismir.net/papers/ISMIR2008_128.pdf.

[75] Douglas Turnbull, Luke Barrington, David A. Torres, and Gert R. G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Trans. Speech Audio Process.*, 16(2):467–476, 2008. doi: 10.1109/TASL.2007.913750. URL https://doi.org/10.1109/TASL.2007.913750.

[76] George Tzanetakis. Automatic musical genre classification of audio signals. In *ISMIR 2001, 2nd International Symposium on Music Information Retrieval, Indiana University, Bloomington, Indiana, USA, October 15-17, 2001, Proceedings*, 2001. URL http://ismir2001.ismir.net/pdf/tzanetakis.pdf.

[77] George Tzanetakis and Perry R. Cook. Musical genre classification of audio signals. *IEEE Trans. Speech Audio Process.*, 10(5):293–302, 2002. doi: 10.1109/TSA.2002.800560. URL https://doi.org/10.1109/TSA.2002.800560.

[78] Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion. *CoRR*, abs/1706.02263, 2017. URL http://arxiv.org/abs/1706.02263.

[79] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2643–2651, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/b3ba8f1bee1238a2f37603d90b58898d-Abstract.html.

[80] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Transfer learning by supervised pre-training for audio-based music classification. In Hsin-Min Wang, Yi-Hsuan Yang, and Jin Ha Lee, editors, *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 29–34, 2014. URL http://www.terasoft.com.tw/conf/ismir2014/proceedings/T007_118_Paper.pdf.

[81] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In Chid Apté, Joydeep Ghosh, and Padhraic Smyth, editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 448–456. ACM, 2011. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020480. URL https://doi.org/10.1145/2020408.2020480.

[82] Fei Wang, Xin Wang, Bo Shao, Tao Li, and Mitsunori Ogihara. Tag integrated multi-label music style classification with hypergraph. In Hirata et al. [41], pages 363–368. ISBN 978-0-9813537-0-8. URL http://ismir2009.ismir.net/proceedings/OS5-1.pdf.

[83] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J. Smola, and Zheng Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs. *CoRR*, abs/1909.01315, 2019. URL http://arxiv.org/abs/1909.01315.

[84] Jason Weston, Samy Bengio, and Philippe Hamel. Large-scale music annotation and retrieval: Learning to rank in joint semantic spaces. *CoRR*, abs/1105.5196, 2011. URL http://arxiv.org/abs/1105.5196.

[85] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6575.

[86] Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. An analysis of approaches taken in the ACM recsys challenge 2018 for automatic music playlist continuation. *ACM Trans. Intell. Syst. Technol.*, 10(5):57:1–57:21, 2019. doi: 10.1145/3344257. URL https://doi.org/10.1145/3344257.