Exploring Corpora-Based Music Classification: Classifying Japanese Popular Music using Lyrics

Xuanrui (Ray) Qi
Department of Computer Science
Tufts University

December 18, 2018

Contents

1 Overview

Music classification is a problem important to many different areas. For music service providers, good classification can enhance user experience and

help provide recommendation services. For music researchers, good classification can help guide their research and test their theories. For music listeners, classification systems can help them discover music akin to their own musical taste.

However, music classification can be a very difficult problem. Currently, many music classification systems are essentially based on the metadata attached to music (i.e., artist, title, album name, etc.) and manually-provided tags. However, this only allows very coarse classification of musical works, e.g., according to genre.

On the other hand, genres can have a lot of internal variation, and such catch-all classifications might not be very useful to users: for instance, both Justin Bieber and the Beatles can be classified as "pop", but they share very similarities. Thus, music recommender systems based on human-provided tags (such as the one used by last.fm) have been developed as well. Compared to purely metadata-based systems, they perform much better, but rely on a large amount of user input as training data.

Can we provide classification based on the *content* of the music instead of external attributes? The answer to this question is obviously positive, but *how*? Spotify has been pioneering classification algorithms based on audio-based machine learning [11]. Here, we propose music classification based on textual corpora, namely a corpora of lyrics. Using a small corpora of Japanese popular music lyrics, we benchmark a few classification methods and compare their effectiveness in accomplishing our task.

With a good corpora-based music classification algorithm, we may use it to augment current music recommender systems, such as those use in music services such as Pandora, Spotify and last.fm. Specifically, corpora-based classification methods allows in-depth examination of thematic traits of musical works, and will allow music recommender systems to produce not only genre-based, but also theme-based recommendations of songs. For example, a user who wants a playlist for their wedding might use such a song to discover wedding-themed songs close to their own music taste. Overall, corpora-based music classification methods, if successful, might be a great advancement to the state-of-the-art of music recommender systems.

2 Related Work

There is an abundance of previous work in music recommendation systems and music classification/tagging systems, most of them utilizing machine learning methods. Text classification is also a rather mature field — however, most of the previous work on text classification assume an English-language corpora, and techniques required to deal with Japanese language[1] corpora

[1] in this report, whenever we refer to the "Japanese language", we mean modern standard Japanese, the standardized form of the Japanese language based on the Yamanote

can be much more tricky than those required for English corpora.

2.1 The "dirty work": tokenization, morphological analysis and POS tagging for Japanese

To work with Japanese language corpora, one must first tokenize the text to transform the corpus into a large array of words. However, unlike English, Japanese writing is not space-delimited. That is, tokens are not separated from each other with spaces. This makes tokenization much hard, as, in order to tokenize a sentence, one must *already* have knowledge of the sentence structure and the grammar. Consider, for example, the phrase 東京国立博物館 (*Tōkyō kokuritsu hakubutsukan*[2] "Tokyo National Museum"): the phrase can either be tokenized as 東京/国立/博物館 ("Tokyo" – "national" – "museum") or 東京国/立/博物館 ("Tokyo State" – "established" – "museum"). In this case, only one of the tokenization schemes make sense, but one cannot ascertain this unless one consult either a dictionary, or learn this from existing annotated corpora.

This brings us to the two existing approaches of Japanese tokenization: either by using dictionary-guided models, or by using machine learning. The first approach requires a Japanese dictionary, and keeps looking for the next token based on a probabilistic or deterministic model. By consulting the dictionary, one could ascertain if the next chunk is indeed a token. Most Japanese tokenizers take this approach, albeit using different underlying models: the most frequently used are MeCab [7] (based on a bigram Markov model), Kuromoji [2] and janome [15] (both based on finite state transducers).

An alternative to deterministic or simple probabilistic models is to use machine learning for this task, learning the tokenization rules from existing annotated corpora. Little work has been done in this area, as the existing tools are already sufficient. However, Morita et al. experimented with a neural tokenizer for Japanese, Juman++ [10]. In practice, we found that the performance of Juman++ is still lacking compared to MeCab and janome, but we believe that this is due to insufficient training of the neural network, rather than inherent deficiency of the approach.

2.2 Text classification

Lyric-based classification of music is essentially a text classification problem. The problem of text classification, then, is a problem central to the field of

dialect of Japanese (山ノ手言葉) spoken in modern Tokyo, and the *de facto* official language of Japan. Most songwriting in contemporary Japan is done in modern standard Japanese.

[2] in this report, whenever we Romanize Japanese text, we use the Revised Hepburn Romanization, using the same conventions as the Ministry of Foreign Affairs of Japan uses for name transcription.

natural language processing. Well-established classification algorithms, such as naïve Bayes and support vector machines (SVMs), have long been used for text classification and are known to provide solid results. These well established approaches are well documented and could perhaps be found in any natural language processing textbook, so we do not elaborate on them here. Interested readers might refer to any of the numerous introductory texts on NLP algorithms.

More recently, researchers have also attempted to use deep learning for this task. Convolutional neural networks (CNNs), for example, are known to perform well on classification tasks. Attempts to use CNNs for sentence classification have achieved good results [6]. In fact, in our work, we base our classification model on Kim's models and achieve rather good results for a small training set.

To use neural network algorithms for text classification tasks, one must convert the input array of words to an array of vectors, i.e. a matrix; in NLP, this is called a *word embedding*. The choice of the word embedding can affect the performance of neural algorithms significantly. The simplest word embedding is to use an arbitrary random vector for each word, and this is indeed sufficient in many cases. However, the state-of-the-art in this area is to use word2vec, a neurally-trained word embedding that captures semantic relationship between words [9]. For example, in a word2vec embedding, we might (approximately) have $v(\text{Paris}) - v(\text{France}) + v(\text{Italy}) = v(\text{Rome})$, correctly representing the semantic relationship between those words.

Since word2vec embeddings capture semantic information, we may also use a similar representation for entire *documents*, i.e. doc2vec [8], and then use vector similarity measures to classify documents. We do not implement this approach for our task, but this is a viable approach worth exploring as well.

2.3 Advances in music classification

Music classification has long been a major challenge in music information retrieval; however, most of the work to date have focused on genre identification using audio signal processing. Tzanetakis and Cook first attempted music classification by supervised machine learning methods [14]. Recently, with the renewed interest in neural networks, researchers have also attempted to tackle this task using convolutional neural networks (see, e.g., [12]). However, we have not yet seen any previous work on using textual (or perhaps a combination of textual and audio) corpora for music classification tasks.

3 Prototype Implementation

We have implemented two tools for music classification based on lyrical corpora — one of which uses frequency analysis, another of which uses a neural algorithm. To be more specific, the first tool uses a naïve Bayes classifier operating on a TF-IDF (term frequency-inverse document frequency) representation of songs, and the second tool uses a convolutional neural network, with word2vec embeddings of our lyrical corpus as input.

3.1 The sample task

The sample task we have identified and chosen is to classify a Japanese language song based on the season it depicts, given its lyrics. We have chosen this task specifically because Japanese poetry, including modern songwriting, have a tradition of setting their work in a specific season; however, this is usually not so textually obvious, although easily determinable for a trained human. For example, Japanese haiku poets (and songwriters) tend to associate certain phrases with different seasons. With enough statistical analysis, one can find the patten between those phrases and certain seasons. We believe that this task is perfect for statistical and/or neural classification. We implement all necessary components for the tool in the Python programming language.

3.2 Data preprocessing

Our data is raw, textual lyrics scraped from the Japanese lyrics website `uta-net`. Before we input the data to our tools, however, we need to preprocess our corpus by tokenizing the corpora, regularizing all tokens into base form, and (perhaps) getting rid of all function words. To accomplish this task, we attempted using both janome and Juman++ tokenizers. From manual inspection of a few tokenized texts, we found that janome performs better, although not significantly better, than Juman++. Moreover, janome is much faster than Juman++ due to using a simpler algorithm. Therefore, we use only the version of our corpora preprocessed by janome as input.

We have also trained a word embedding table for the Japanese language in order to provide good word embeddings for our input. Our choice of word embedding model is the standard word2vec model, trained on a recent dump of the Japanese Wikipedia.

3.3 Implementing the neural model

Our neural network architecture closely follows the model described in [6], which, in turn, is a convolutional neural network modeled on the architecture described in [3]. Specifically, we use the *non-static embedding* approach described in the paper, allowing the neural network to fine-tune the word

embeddings while training. However, we deviate slightly from the model by using word embedding vectors of dimensionality 50 rather than 300.

We reuse an implementation of the said neural network in TensorFlow [1], and train the network using a NVIDIA 840M GPU, on a training set of size 431. Each training input is the processed lyric of a song, represented as a 2D matrix (where each row represents a word). We use the standard approach of separating our dataset into three components: a training set (size 431), a "dev" set (size 48) used for evaluation and tuning during training, and a testing set (size 54) used for post-mortem evaluation of the model. The separation is done automatically and randomly each time we train the neural network.

With a rather small training set (by standards in the neural networks community), we have achieved pretty good performance results. The accuracy of our trained neural network, when tested on the randomly-picked testing set, achieved varying levels, mostly between 0.65 and 0.80. Compared to the evaluation results given in the original paper [6], we believe that this is a fairly good performance on a small data set.

3.4 Implementing the word frequency-based model

Frequency models are long known and long used for text classification and are known to be perform stably well, and many machine learning libraries have good support of them. For our task, we implement a simple TF-IDF vectorization of words, and use a multonomial naïve Bayes classifier to perform classification for the task. The entire implementation is contained within one Python script and is only about 100 lines, using the scikit-learn library for both frequency calculation and naïve Bayes classification.

Although the model is simple, the performance is surprisingly good considering the size of our training set: we use random selection, again, to select training and testing sets; the testing set is on average 10% of all data. Our accuracy fluctuates around 60% after several different training experiments; though, it is worthy to note that our naïve Bayes model trains significantly faster than a neural network: training our neural networks (1,000 epochs) takes about 45 minutes, while training a naïve Bayes classifier only takes a few seconds.

3.5 Code repository

All of our code could be found at: `https://github.com/xuanruiqi/lyrics-classification`.

4 Work Plan

4.1 Building a larger corpora

It is often said that machine learning is "all about the data". Our experiments are relatively small in scale, and our resources are limited. Specifically, all tagging is either manually completed by me, or scraped from existing websites. Consequently, the corpora we have worked with is relatively small. However, it is very hard to learn well if there is not enough training data. Therefore, the first step towards building a practical corpora-based music classifier would be to build a large enough tagged corpora, probably a collection of music lyrics with multiple manually assigned tags, such as "genre", "sentiment", and "appropriate occasions".

A challenge to this goal would be copyright issues. However, in order to circumvent this, we might opt to not include the actual text in our corpora, rather including only the tags and annotations in this corpora. This is the same approach that Kawahara, Kurohashi, and Hasida have taken while building the Kyoto University Text Corpora (hereafter "Kyoto Copora") [5]: the text of the Kyoto Corpora is taken from a proprietary dataset, the 1995 releases of the Mainichi Shimbun, and the publicly released version of the Kyoto Corpora does not contain any of the proprietary text, in order to circumvent possible copyright issues. As for the actual lyrical text, we could provide a scraper for lyrics sites that downloads the text of the lyrics, as most lyrics sites allow spiders and web crawlers to craw content from their website without issue: as long as no proprietary text is directly provided, there should be no copyright issues involved.

Another challenge to this goal would be the human effort required for the tagging. This task, however, can be crowdsourced, for example via Amazon Mechanical Turk.

4.2 Exploring new classification algorithms

One issue with music classification is that music often fall into more than one category at the same time. For example, a song might simultaneously have characteristics of rap and pop, and can be classified as both concurrently. This task, *multi-label classification*, is one that often appears in machine learning task. However, compared to single-label classification, this is a problem that has been tackled much less frequently. A recent paper by Singh, Thomas, Marshall, Shawe-Taylor, and Wallace, for example, explores multi-label classification of biomedical texts via attentive neural tree decoding [13]. In the coming years, we plan to work with researchers in machine learning and natural language processing, exploring and extending the state-of-the-art in multi-label classification.

Moreover, all of the classification models we have explored here are essentially domain-specific: that is, one trained model can only perform one

task; a model for genre classification, for instance, cannot perform sentiment analysis. This greatly increases the difficulty of creating multi-task music classifiers. However, a technique called *transfer learning* tackles this problem by transferring learned knowledge to other tasks [4]. In the coming years, we plan to investigate applications of transfer learning to music classification tasks.

4.3 Combining audio and textual corpora

Our prototype implementation uses solely lyrics to classify music, essentially reducing music classification tasks into text classification tasks. However, lyrics are different from generic text in that they are associated with audio data as well. Naturally, we want to annotate lyrical text with musical data to enhance machine learning algorithms. For example, we may want to associate each token with its pitch and/or its tempo. However, we have not seen any work exploring this direction as of yet, and we are not sure if this will bring significant performance gains. Nevertheless, in the next three years, we wish to explore this direction to see if it brings us any benefits.

4.4 Bridging the research-industry gap

Even if we develop good classification methods and algorithms for music classification, we are not sure if they are practical in industry. Industrial-scale music recommendation systems (such as those developed by Spotify and Pandora) have access to considerably more data, but are also more constrained by time and computing resources. Our models, for example, might train very slowly or require too much computing resources to be useful at scale. In the next three years, we wish to get in contact with industry and evaluate the feasibility of our solution in production environments.

5 Quality Control

Quality control is important for our task. Since our task is essentially a machine learning task, our success can be measured essentially by the accuracy of our models. In the machine learning community, a good text classification algorithm usually needs to have at least around 90% accuracy to claim success; our model's performance is currently significantly lower than 90%, at around 60% to 70%. We hope that enhanced algorithms and larger corpora could bring the accuracy of our system to around 90% or more.

However, besides accuracy, we also have other considerations. For example, we might have limitations on computing resources and/or human capital required for a successful, practical system. These benchmarks will need to be decided after consultation with practitioners and industry researchers.

6 Final Deliverable

The final deliverable of our project shall be a mature, production-deployable, corpora-based music classification system that can be integrated into other services. For example, a company like Spotify might use our system on license to provide sentiment-based music recommendations for their systems. Alternatively, we may think of recommendation as a binary classification task: music that the user has liked would be "positives", and music that the user has disliked would be "negatives"; by methods such as transfer learning, our system should able to provide good recommendations even if the input set for the task *per se* is small.

When our project has attained good accuracies, we may plan as releasing our project as a licensable product. The product would likely take form of a (likely black-box) software system that, when given input data and labels, trains a model and provides an interface for users to conduct classification tasks. Since this system will likely involve computationally-heavy machine learning tasks, it would probably only be made available to corporate users.

Moreover, when we have attained good results, we also plan to publish our approach and model in one of the machine learning conferences (such as the Conference on Neural Information Processing Systems (NIPS), or the International Conference on Machine Learning (ICML)). We intend to describe our model and approach with enough detail to replicate our results, but without disclosing any of our original code and/or data to prevent copyright and/or business issues.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: a system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016. URL: `https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf`.

[2] Atilika. Kuromoji - Japanese morphological analyzer. 2018. URL: `http://www.atilika.org/` (visited on 12/14/2018).

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, Nov. 2011. ISSN: 1532-4435. URL: `http://dl.acm.org/citation.cfm?id=1953048.2078186`.

[4] C. B. Do and A. Y. Ng. Transfer learning for text classification. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, pages 299–306, Vancouver, British Columbia, Canada. MIT Press, 2005. URL: http://dl.acm.org/citation.cfm?id=2976248.2976286.

[5] D. Kawahara, S. Kurohashi, and K. Hasida. Construction of a japanese relevance-tagged corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA), 2002. URL: http://www.lrec-conf.org/proceedings/lrec2002/pdf/302.pdf.

[6] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics, 2014. DOI: 10.3115/v1/D14-1181. URL: http://aclweb.org/anthology/D14-1181.

[7] T. Kudo. Mecab: yet another part-of-speech and morphological analyzer. 2018. URL: https://taku910.github.io/mecab/ (visited on 12/14/2018).

[8] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II-1188–II-1196, Beijing, China. JMLR.org, 2014. URL: http://dl.acm.org/citation.cfm?id=3044805.3045025.

[9] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, Lake Tahoe, Nevada. Curran Associates Inc., 2013. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.

[10] H. Morita, D. Kawahara, and S. Kurohashi. Morphological analysis for unsegmented languages using recurrent neural network language model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2292–2297, Lisbon, Portugal. Association for Computational Linguistics, 2015. DOI: 10.18653/v1/D15-1276. URL: http://aclweb.org/anthology/D15-1276.

[11] N. Patch. Meet the man classifying every genre of music on Spotify —all 1,387 of them. 2018. URL: https://www.thestar.com/entertainment/2016/01/14/meet-the-man-classifying-every-genre-of-music-on-spotify-all-1387-of-them.html/ (visited on 12/14/2018).

[12] C. Senac, T. Pellegrini, F. Mouret, and J. Pinquier. Music feature maps with convolutional neural networks for music genre classification. In *Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing*, CBMI '17, 19:1–19:5, Florence, Italy. ACM, 2017. ISBN: 978-1-4503-5333-5. DOI: 10.1145/3095713.3095733. URL: http://doi.acm.org/10.1145/3095713.3095733.

[13] G. Singh, J. Thomas, I. Marshall, J. Shawe-Taylor, and B. C. Wallace. Structured multi-label biomedical text tagging via attentive neural tree decoding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2837–2842, Brussels, Belgium. Association for Computational Linguistics, 2018. URL: http://aclweb.org/anthology/D18-1308.

[14] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, July 2002. ISSN: 1063-6676. DOI: 10.1109/TSA.2002.800560.

[15] T. Uchida. Janome v0.3 documentation. 2018. URL: https://mocobeta.github.io/janome/ (visited on 12/14/2018).