Automatic Harmonization Viewed as a Machine Translation Problem

Dept. of CIS - Senior Design 2014-2015[*]

Nicole Limtiaco
limni@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Rigel Swavely
rigel@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

ABSTRACT

In the past, approaches to automatic harmonization of a melody have taken two forms: using rules provided by music theory or by predicting the chord under the melody. This attempt approaches the problem from a machine translation perspective, modeling the melody of a song as the source language and each harmony as a target language. By generating the harmony lines explicitly rather that generating them as a consequence of the chord on each beat, we hope that the algorithm will be able to create more cohesive, creative works.

In order to accomplish this, we will create a translation model to represent the probability of one note harmonizing with another note. Additionally, we will create a language model to represent a note following a phrase of notes in the same line of music. A perplexity metric will be used to evaluate the performance of our algorithm. If there is time, we hope to incorporate human evaluation and data driven rhythmic variation.

1. INTRODUCTION

Many would consider music composition to be an art form that is accomplished primarily through human creativity. Writing music is a process that seems to require complex thought which fundamentally cannot be boiled down to a list of straightforward processes. Our project aims to answer the question, "Can computers imitate the complex and creative process of composing music?" We narrow the question down to the more specific and well-defined problem of generating a group of harmony lines for a given melody line.

We will define a *melody* as a sequence of input *notes*, where each note contains information about pitch and timing. A *harmony* will be a sequence of output notes which are produced with the constraint that the sequence of notes supports the input melody. We define the term *voice* to be some sequence of notes, either a melody or one of the harmonies, with a certain set of identifying characteristics. For example, the bass part in rock music is one type of voice and the soprano's part in an opera is another. The end result we wish to generate will be a group of n voices–the input melody and $n-1$ automatically-composed harmonies–that, when played together, sound coherent and pleasant.

We propose to approach the problem from the view point of machine translation, where the input language is the melody and the target language is the specific harmony part you wish to generate. The first analogy that can be drawn between a pair of natural languages and a pair of melody and harmony voices is that both have a sense of "word translations". Just as there may be several words in the target language that could be sensible translations of a given word in the input langauge, there are also several notes in the harmony "language" that can harmonize well with an input melody note. Importantly, however, it is not the case that any note can harmonize well with the melody note. Some notes will sound dissonant when played together and still other notes may not even be in the harmony language, since harmony voices can have specified note ranges.

The second analogy that allows us to view this problem as a machine translation problem is that, like natural language, only certain strings of tokens (i.e. notes) are sensible in a given harmony voice. For example, the statement "colorless green ideas sleep furiously" contains all english words but is unlikely to be understood by an english speaker because the string of words in not sensible based on the rules of our language. Similarly, a very inharmonious sequence of notes may not sound sensible in the context of its harmony voice, if the notes are even recognized as music at all. This analogy shows us that it should be possible to generate new content in the harmony voice similar to how machine translation algorithms are able to generate new natural language content.

2. RELATED WORK

Automatic harmonization is a subset of the automatic musical composition problem, which dates as far back as the field of artificial intelligence itself. Perhaps the earliest work in automatic composition is Hiller and Isaacson's *Illiac Suite* [5], which is widely accepted as being the first musical piece composed by an electronic computer. Hiller and Isaacson used a generate-and-test method that generated musical phrases psuedo-randomly and kept only those that adhered to a set of music-theory-inspired heuristics.

Staying in line with the use of musical rules, Ebcioğlu [3] provided a break-through in the specific field of automatic harmonization through his CHORAL system. CHORAL, bolstered by about 350 music-theory and style rules expressed in first order logic, performed the task of writing four-part harmonies in the style of J.S. Bach. With these logical predicates, Ebcioğlu reduced the problem of composing a harmony to a simple constraint satisfaction problem. Similar later works, notably by Tsang & Aitkin [11], also crafted the harmonization problem as a problem in constraint satisfaction, but with a significantly smaller amount

[*]Advisor: Insup Lee (lee@cis.upenn.edu).

($\sim$20) of musical rules. The result of these constraint-based works were musically sensible; however, crafting the constraints such that the output is musical requires deep, human knowledge about music in general and about the style of music to be produced in particular.

More recent works began to put data-driven methods into use in order to infer the patterns that govern real compositions. A simple case-based model implemented by Sabater *et al.* [8] was built to generate accompanying chords for a melody line. To a choose a harmony chord for a given context of previous harmony chords and the currently sounding melody note, the system would check a case base to see if any cases had the same harmony context and melody note and use the corresponding harmony chord if such a match were found. Musical heuristics were used to generate a chord if no match was found in the case base. An automatic harmonizer utilizing neural networks was also built by Hild *et al.* [4] to produce a harmony chord for a given melody quarter beat. Input features to the network included harmony context, current melody pitch, and whether or not the beat was stressed.

As these examples show, the previous harmony context and the melody pitch are important signals in deciding what the current harmony phrase should be. Many works have been conducted that model these signals using n-gram Markov Models. A Markov model assigns probabilities to some event C_t conditioned on a limited history of previous events [C_{t-1}... C_{t-n}], implictly making the assumption that the event C_t depends only on a small amount of information about its context. For example, a system called MySong produced by Simon *et al.* [10] generates chord accompaniment given a vocalized melody by using a 2-gram Markov model for the harmony context and a 2-gram Markov model for the melody context. A similar system implented by Scholz *et al.* [9], which also generates chord accompananiments, experimented with 3-gram to 5-gram Markov models and incorporated smoothing techniques commonly seen in NLP to account for cases in which the model has not seen a context that is present in the test data. Most recently, Raczyński *et al.* [7] uses discriminative Markov models that model harmony context, melody context, and additionaly the harmony relationship to the tonality.

The recent Senior Design Project by Cerny *et al.* [1] also used melody pitch and previous harmony context as their main signals for determining the next harmony chord to generate. However, they used these signals as inputs to an SVM classifier, as opposed to training a Markov model.

3. PROJECT PROPOSAL

Like the more recent attempts at automatic harmonization described previously, our attempt at this problem will also take a data driven approach. However, instead of trying to predict the chords that fit a certain melody and generating the harmony based on these chords, we look at the individual lines themselves and how they have been translated into harmonies in the dataset. For each note m in the melody line, we will predict one harmony note h per voice based on two characteristics: the probability of m being harmonized by h and the probability of h following the previous notes in the harmony, both according to the training data.

3.1 Anticipated Approach

In order to predict the best harmony note h to harmonize with a melody note m and follow other notes in the harmony, we make an analogy to machine translation techniques. In machine translation, when attempting to translate from some source language S to a foreign language F, two models are developed - a language model and a translation model. A language model determines for each phrase $f_1..f_n$ of foreign words how likely it is for some word f to follow this phrase. Generally, this is accomplished by using a Markov model of length n trained on some foreign language data set. We can apply this technique directly to our harmonizer, determining how likely it is that some note h will follow a sequence of notes $h_1..h_n$ according to the harmonies that our algorithm has trained on. More formally, our language model is defined as the following:

$$p_{LM}(H) = \Pi_{i=1}^{l} P[h_i|h_{i-1}..h_{i-n}]$$

Additionally, we may apply certain penalties to this model as in the case in machine translation, potentially based on distance or dissonant intervals between notes. Representing this penalty as the function d, our new translation model becomes:

$$p_{LM}(H) = \Pi_{i=1}^{l} P[h_i|h_{i-1}..h_{i-n}] * d(h_i, h_{i-1})$$

The translation model refers to how likely it is that some foreign word f is a translation of a source word s. This model is created by training on a data set of alignments from source words to foreign words and cannot be a simple translation dictionary, as one source word may translate to several different foreign words depending on context. We again apply this concept directly to harmonization, determining the best harmony note h to match a melody note m. Like in natural language translation, there are many possible candidate notes h to match one particular note m, and the best one can be chosen based on context. However, harmonization is a simpler problem in that every harmony note is already aligned to its corresponding melody note based on when the notes sound, whereas in natural language, the words must be aligned to each other in a separate processing step. To formalize the translation model, we have:

$$p_{TM}(M|H) = \Pi_{i=1}^{l} P[m_i|h_i]$$

To put these two models together, given some melody line M we would like to find a harmonly line H such that:

$$H = argmax_H p(H|M) = p_{TM}(M|H)p_{LM}(H).$$

To simplify computation, we will maximize the log of the function above so that:

$$H = argmax_H \ log(p_{TM}(M|H)) + log(p_{LM}(H))$$

$$= argmax_H \sum_{i=1}^{l} log(P[m_i|h_i]) + log(P[h_i|h_{i-1}..h_{i-n}]) + log(d(h_i, h_{i-1}))$$

In order to maximize the function above, we plan to implement a beam search by building up the potential translations $H^j = \{h_0, ..., h_j\}$ for $M^j = \{m_0, ..., m_j\}$ where $0 \leq j \leq l$ and only maintaining the top k solutions for each increment of j.

3.2 Technical Challenges

The initial challenge is to compile a database of music scores and to find a way to interact with the scores programmatically. To this end, we propose using a library developed at MIT called music21 [2] which provides a simple interface for querying the library's database of musical scores.

Another problem we may enounter is the combination of multiple harmony lines. This is because it is possible for two harmonies to sound consonant with the same melody, but sound dissonant with each other. For example, given a melody note $m = E$, two viable candidates to harmonize might be $h_1 = C$ and $h_2 = B$. While these both harmonize well with m in some context, h_1 and h_2 in this case form a very dissonant interval which is less likely to be found in a well formed harmony. In order to mitigate this issue, we may alter the translation model to condition on more than one note. For example, when deciding the assignment to h_2, this method would take both $m = E$ and $h_1 = C$ into account in order to determine a value for h_2 which harmonizes with both. However, if one wished to create an n part harmony, this would require $n - 1$ translation models conditioned on $1 \ldots n - 1$ other lines. This poses a problem both in time and space efficiency as well as sparseness of data - as models are conditioned on more and more lines, the number of unique contexts trained on increases greatly, and therefore each context is encountered in the training data less often. This means that differentiating between likely and unlikely contexts will become difficult without enough data to train on, and may limit the number of parts that can be produced well. Additionally, this means that adding the voices in different orders may affect the output of the algorithm. This can be mitigated by trying every possible ordering of harmony generation and choosing the one that optimizes our performance metric.

3.3 Evaluation Criteria

In order to evaluate our model, we will attempt to minimize its perplexity, a common metric used in natural language processing. This metric captures the likelihood that our algorithm would predict an existing song in our test set. The perplexity PP [6] of a probability distribution p is defined over all songs s in our test set as:

$$log_2 PP = -\sum_s log_2 p(m_s|h_s)$$

This measure gives us a real number metric to evaluate different iterations of our algorithm. If time permits, we may have human evaluators compare the results of different iterations of our algorithm or compare our algorithm to existing automatic harmonization systems.

4. RESEARCH TIMELINE

We have completed some preliminary steps for our project and plan to complete our work along this proposed timeline.

- ALREADY COMPLETED: Preliminary reading. Experimented with musci21 library and corpus by implementing a deterministic harmony of thirds generator. Initial steps taken to train language model and translation models.
- PRIOR-TO THANKSGIVING : Finish training language model and translation model. Write beam search to find solution for one harmony voice.
- PRIOR-TO CHRISTMAS : Tune beam size and n-gram size. Implement code to generate harmony voices in different orders and determine which order is best.
- COMPLETION TASKS : Verify implementation is bug-free. Complete write-up.
- IF THERE'S TIME : Investigate incorporating rhythm and timing decisions into the generated harmonies. Get humans to evaluate output.

5. REFERENCES

[1] David Cerny, Jiten Suthar, and Israel Geselowitz. U-amp: User input based algorithmic music platform. 2014.

[2] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data.

[3] Kemal Ebcioğlu. An expert system for harmonizing chorales in the style of j.s. bach. *The Journal of Logic Programming*, 8(1 - 2):145 – 185, 1990. Special Issue: Logic Programming Applications.

[4] Hermann Hild, Johannes Feulner, and Wolfram Menzel. Harmonet: A neural net for harmonizing chorales in the style of j. s. bach. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 267–274. Morgan-Kaufmann, 1992.

[5] L.A.J. Hiller and L.M. Isaacson. *Experimental Music: Composition with an Electronic Computer.* McGraw-Hill, 1959.

[6] Philipp Koehn. *Statistical Machine Translation.* Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[7] Stanisław A. Raczyński, Satoru Fukayama, and Emmanuel Vincent. Melody harmonization with interpolated probabilistic models. *Journal of New Music Research*, 42(3):223–235, 2013.

[8] Jordi Sabater, J. L. Arcos, and R. López De Mántaras. Using rules to support case-based reasoning for harmonizing melodies. In *Multimodal Reasoning: Papers from the 1998 AAAI Spring Symposium (Technical Report, WS-98-04). Menlo Park, CA*, pages 147–151. AAAI Press, 1998.

[9] R. Scholz, E. Vincent, and F. Bimbot. Robust modeling of musical chord sequences using probabilistic n-grams. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 53–56, April 2009.

[10] Ian Simon, Dan Morris, and Sumit Basu. Mysong: Automatic accompaniment generation for vocal melodies. In *CHI 2008 Conference on Human Factors in Computing Systems.* Association for Computing Machinery, Inc., April 2008.

[11] C.P. Tsang and M. Aitken. Harmonizing music as a discipline of constraint logic programming. 1991.