Université de Fribourg

Bachelor Thesis

Irish Melody Maker

Noé Zufferey

supervised by
Alberto Tonon & Prof. Philippe Cudré-Mauroux
Exascale Infolab

May 3, 2017

Contents

1 Preface

1.1 French

À notre ère où la Machine semble vouloir remplacer l'Homme dans de plus en plus de domaines, il est intéressant d'essayer de découvrir et de comprendre les méthodes nous menant à de tels résultats. Le domaine artistique a souvent été considéré propre à l'Homme et inimitable sans, entre autre , la part émotionnelle de l'être humain. Cependant, à moins de se fier à une création parfaitement aléatoire, toute oeuvre est créée à partir de règles, qui pourraient s'apparenter à des algorithmes. Et si, la Machine était capable de comprendre ces règles et de les utiliser? C'est la question à laquelle nous avons essayé de répondre, du moins en partie, avec le projet *Irish Melody Maker*.

1.2 English

During this era, the Machine seems to replace the Human Being more and more. It is interesting to try to discover and understand the methods that lead us to such a result. The Arts, has always been considered as a Human thing since human emotions make it inimitable. However, except for cases in which pieces of Art are created totally randomly (and this constitutes their artistic value), every piece of art is created from rules, that, in some ways, look very much like algorithms. What if the Machine could understand these rules and use them? That the question to which we tried to answer, at least in part, through the project *Irish Melody Maker*.

2 Introduction

The Irish melody maker project was supervised by Alberto Tonon, PhD student at the Exascale Infolab of the University of Fribourg in Switzerland and Prof. Philippe Cudré-Mauroux, leader of the Exascale Infolab.

The main goal of the project is to create a program that generates Irish style melodies. The program has to learn how a human being composes a melody. The database of the website TheSession.org [1] is used as a reference for composing rules. *Markov Chains* [2] are used in order to create a knowledge base with which the program could imitate human composition.

3 Irish Music

Irish music is mostly dance music. It includes many styles but there are two main style categories: tunes whose measures are in $\frac{x}{8}$, like jigs ($\frac{6}{8}$) or slip jigs ($\frac{9}{8}$), and the tunes whose measures are in $\frac{4}{4}$, like reels or hornpipes. There are sometimes $\frac{3}{4}$ measures, like waltz.

A tune is most of the times composed of different parts. Each part can be repeated or not. This parts repartition is called the form. For example a tune

that begins with a melody, continues with another melody and ends with the first melody has the form *aba*.

A melody is composed in a specific mode. The mode defines the organization of each pitch, called degree, of a scale. This repartition is also defined by the interval size between two notes, called the interval sequence. A scale consists of 7 degrees. In Irish music, there are four main modes:

- Ionian
  Also referred as major mode, it is the most used.
  The interval sequence is T-T-s-T-T-T-s ("T" is for tone and "s" is for semitone)
  For example, C major scale: c-d-e-f-g-a-b-c
- Mixolydian
  It is the same as Ionian except that the seventh degree is a semitone lower.
  It also corresponds to a Ionian mode that begin on its fifth degree.
  The interval sequence is T-T-s-T-T-s-T
  For example, G mixolydian scale: g-a-b-c-d-e-f-g
- Eolian
  Also referred as minor, it corresponds to a Ionian mode that begin on its sixth degree.
  The interval sequence is T-s-T-T-s-T-T
  For example, A minor scale: a-b-c-d-e-f-g-a
- Dorian
  It is the same as Eolian except that the sixth degree is a semitone higher.
  It corresponds to a Ionian mode that begin on its second degree.
  The interval sequence is T-s-T-T-T-s-T
  For example, D dorian scale: d-e-f-g-a-b-c-d

3.1 The Data Set

Irish Melody Maker's algorithms use statistics which are created from a data set of Irish tunes. Extracted from TheSession.org [1], a platform that allows users to share Irish tunes, the data set consists of more than 20'000 tunes and 11 different styles.

In addition to this, we computed a few extra statistics.

- For every style, each part of melody is repeated in average one time, except for Three-two style for which it is two times.
- The most used key signature is A major, for all tunes and by tune type.
- The average of number of measure per tune is 22 (repeated measures are counted). We can see this average by type of tune in *Table 3*.

Irish music was chosen because, like almost every traditional music, of its repetitive melodic structure, we believe that it is simpler to automatise than other more complex music genres (e.g. Jazz or Classical music).

Table 1: *Number of tunes by style*

style	number of tunes
Reel ($\frac{4}{4}$)	9050
Jig ($\frac{6}{8}$)	6119
Hornpipe ($\frac{4}{4}$)	1880
Polka ($\frac{4}{4}$)	1850
Waltz ($\frac{3}{4}$)	1654
Barndance ($\frac{4}{4}$)	1215
Slip jig ($\frac{9}{8}$)	929
Strathspey ($\frac{4}{4}$)	766
Slide ($\frac{12}{8}$)	568
Mazurka ($\frac{3}{4}$)	369
Three-two ($\frac{3}{2}$)	243

Table 2: *Number of tunes by style and mode*

style	major	dorian	mixolydian	minor
Reel ($\frac{4}{4}$)	5573	1481	734	1262
Jig ($\frac{6}{8}$)	4022	723	505	869
Hornpipe ($\frac{4}{4}$)	1533	145	56	146
Polka ($\frac{4}{4}$)	1425	188	65	172
Waltz ($\frac{3}{4}$)	1123	119	70	342
Barndance ($\frac{4}{4}$)	991	67	29	128
Slip jig ($\frac{9}{8}$)	539	126	74	190
Strathspey ($\frac{4}{4}$)	445	139	72	110
Slide ($\frac{12}{8}$)	421	70	46	31
Mazurka ($\frac{3}{4}$)	284	20	5	60
Three-two ($\frac{3}{2}$)	125	24	15	79

4 The Algorithms

The main part of Irish Melody Maker is a method that, given a style and a mode, generate a piece of melody. Irish Melody Maker can use two different algorithms to generate its melodies. Both algorithm build a melody in two steps: first they define the rhythm of the tune, and then they generate its melody (i.e. decide the pitch of each note). The sheet of melody is represented by a list of notes. Each note is a tuple representing a rhythmic value and a pitch. During the first step, the note's pitch is undefined. The main difference between the two algorithms is how they generate the rhythm of the tune.

Table 3: *Repartition of styles and modes in the data set*

style	average of number of measure
waltz ($\frac{3}{4}$)	39
jig ($\frac{6}{8}$)	31
mazurka ($\frac{3}{4}$)	31
barndance ($\frac{4}{4}$)	30
polka ($\frac{4}{4}$)	30
hornpipe ($\frac{4}{4}$)	29
reel ($\frac{4}{4}$)	27
slip jig ($\frac{9}{8}$)	22
strathspey ($\frac{4}{4}$)	22
three-two ($\frac{2}{4}$)	22
slide ($\frac{12}{8}$)	17

4.1 Creation of the Rhythmic Part

As previously mentioned, the first step of the composition process consists in creating the rhythm of the tune. The two algorithms we designed do this by taking as input the number of measures to be created and by creating a rhythmic pattern as output. This will be repeated 2 or 4 times according to Irish music usual melodic construction.

4.1.1 Rhythmic Construction - Algorithm 1

ALGORITHM 1 is based on the frequency each rhythmic value appears in every tune of our data set. Formally, we analysed all the tunes contained in THESESSION.ORG and computed the probability of seeing a certain rhythmic value (ex. crotchet, quaver, semiquaver,...) given a type of tune (ex. jig, reel,...), that is, $\Pr(R = r \mid T = t)$, where R, T are random variables representing rhythmic values and types of tunes. Note that a triplet is considered as a unique rhythmic value.

For example, in *Figure 1*, to decide to add a crotchet (in red) at the end of the sequence, the algorithm just randomly draw a rhythmic value according to the probability of occurrence of each rhythmic value, regardless the previous rhythmic values (in black).

4.1.2 Rhythmic Construction - Algorithm 2

The second algorithm, namely ALGORITHM 2, is more sophisticated as it takes into consideration entire rhythmic structures and it exploits the context in which a certain rhythmic pattern appears. In this context, rhythmic patterns are:

- Each lonely rhythmic value (crotchet, quaver, semiquaver,...).
- Triplet.

Figure 1: *Example of generation for a reel, the last rhythmic value (red) is selected according to* $\Pr(R = r \mid T = reel)$.

- Sequence of 2 quavers, if it starts on the beat.
- Sequence of 3 quavers, if it starts on the first or the fourth beat in a $\frac{x}{8}$ measure.
- Sequence of 4 quavers, if it starts on the beat in $\frac{x}{4}$ measure.

Moreover, we model the context in which a rhythmic pattern appears by keeping track of the patterns that precede it in our data set. Due computational constraint, we adopted the Markov assumption. Thus, we assume that the $n+1$ -th rhythmic pattern only depends on a subsequence of previous patterns. As we will see in the implementation of algorithm *Section 5.2*, during our experiments we consider subsequences of length 2. Formally, we compute $\Pr(S_{n+1} = x \mid S_n = y, S_{n-1} = z)$, where the S_i are random variables representing the rhythmic value of the note at the i-th position of the sheet and x, y, z are rhythmic patterns.
As we can see in *Figure 2*, to choose the last rhythm value (in red), the algorithm will use its knowledge of the blue and the green patterns.

Figure 2: *Example of generation for a reel, the last rhythmic value (red) is selected in knowledge of the two previous sequences (blue and green).*

4.2 Melody Generation

After obtaining a rhythmic sheet, the algorithm has to assign a pitch to each rhythmic value previously computed. As the rhythmic sheet consists of a rhythmic pattern repeated a number of times, the melody construction follows the same strategy. All parts of the sheet that are based on the same rhythmic pattern will be based on the same melodic pattern.
Each note pitch is chosen by using a 3-degree *Markov Chain* [2], it means that the pitch of each note depends on the three previous note pitches. The three

first note pitches of the melody are chosen the same way, except that each non-existent previous note pitch is called ϵ to show that there is no previous note pitch. Thus, we have $\Pr(S_{n+1} = x \mid S_n = y, S_{n-1} = z, S_{n-2} = w)$, where the S_i are random variables representing the pitch of the note at the i-th position of the sheet.

ßß *Figure 3* represents the beginning of a *Markov Chain* used to generate a melody. The melody generation begins with three empty notes (ϵ), then the algorithm chooses the next note according to the statistics of three empty previous notes. Next, the chosen note and its two previous notes are used to select the next note, etc...

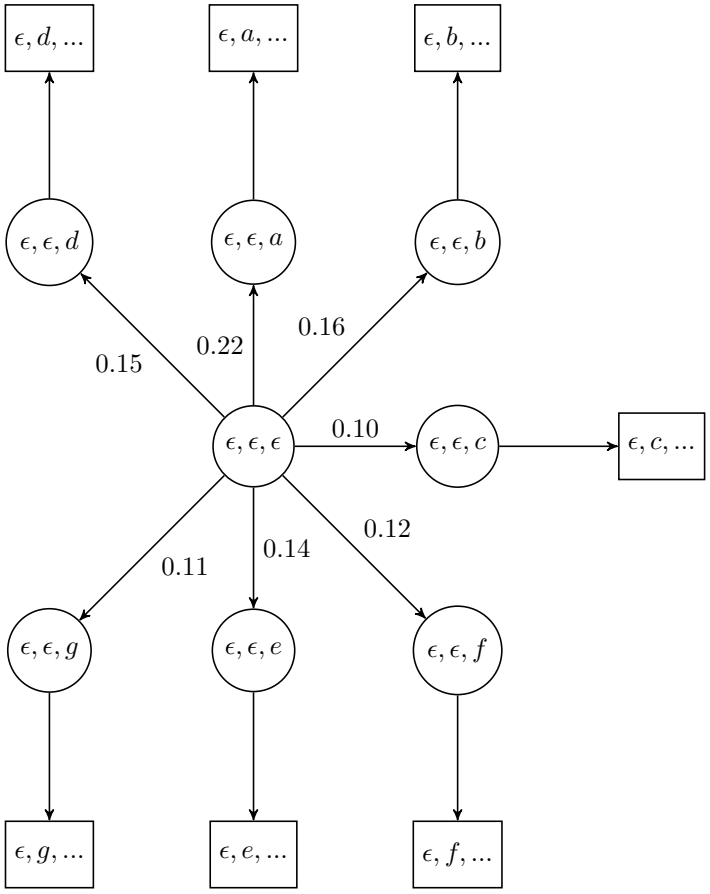

Figure 3: *The beginning of the 3-degrees Markov Chain for a minor waltz melody.*

4.2.1 Octave Shift

Due computational constraints, we had to construct the *Markov Chain* around a unique octave, that means that every note pitch is normalised to be in the same octave as the others ones. In fact, with a large distribution of range, the statistics files was long to load, not too long to a single tune generation but still really restrictive to testing. For example, with 4 octaves, we had a *Markov Chain* with more than 5 million states while decreasing this range to only one octave we just have a little bit more than 100'000 states.

Thus, the melody composed in the two last steps just fit one octave. Therefore, the algorithm has to extend the melody on several octaves. For that purpose, we created two methods, namely DOWNNOTESMELODICSHEET and UPNOTESMELODICSHEET. They both browse the entire sheet and look at the gap between the note pitches. If this spacing is too big, they consider that the second note has to be put on an other octave: an higher octave when UPNOTESMELODICSHEET is applied, and a lower one when DOWNNOTESMELODICSHEET is applied.

Both algorithms are based on general knowledge of music theory and on some music theoretical works and books [4] that analyse how melodies are composed. For example, in a *C major* mode, when a b appears in the melody, it has to "resolve" itself, that is, the b "leads" to another note that most the of time is the c just a semitone above.

4.2.2 Tune Endings

Most human beings "need" a tune that "resolves" itself. That means that they prefer when a tune is ending with a pitch and a rhythmic pattern that "shows" that it is the end. For exemple, a C is a most resolving ending pitch than a B for a tune in C major.
For that purpose, we decided to use again the data set of THESESSION.ORG. We create statistics on the two last rhythms and the three last note pitches for every tunes. Then, we use these statistics to replace the tune ending. That is to say that we randomly draw a rhythmic ending pattern according the style of the tune, we then have $\Pr(P = p \mid T = t)$, where P, T are random variables representing rhythm patterns and types of tunes. Then we replace the rhythmic ending by this rhythm pattern according to the rhythmic signature.
Next, we have to replace the pitch of the last notes, so we randomly draw a three pitches sequence according to the style of the tune but also the antepenultimate pitch of the melody. Indeed, we decided to use the ending pitch statistic like a *Markov Chain* to keep a continuity in the melody. So we have $\Pr(N_1 = x, N_2 = y \mid T = t, S_{(n-2)} = z)$ where T is a random variable representing the type, the N_i's are random variables representing the two last notes of the melody and $S_{(n-2)}$ is a random variable representing the antepenultimate note of the melody. We obtain so a proper ending for the tune.

5 Implementation of Irish Melody Maker

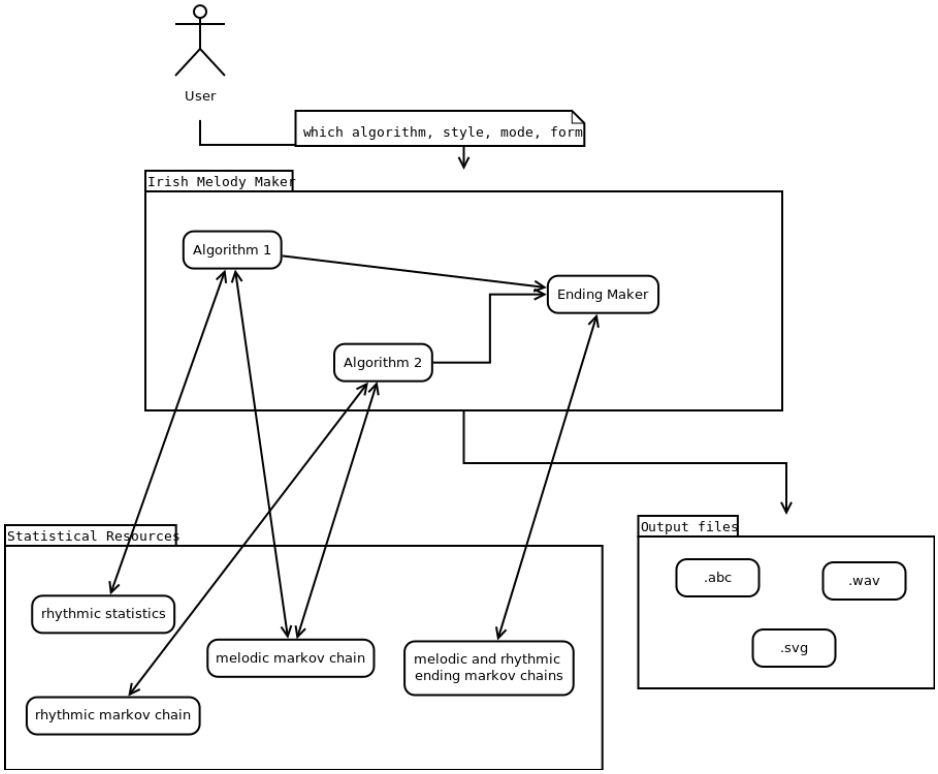

Figure 4: *Architecture diagram of* Irish Melody Maker

5.1 Languages and Libraries

Irish Melody Maker is implemented by exploiting several programming languages and serialization formats. The program is coded in python and the data that it needs is saved in Json.
Since the dataset is distributed in abc notation, the program uses a library called pysynth [3] which is able to read this format and generates audio files (*.wav*). This library is used to generate audio files from the program's output. The program use abcm2ps which is a command line program to generate svg music sheet with abc files. Furthermore, a part of this library was used to generate the program's knowledge base (statistics and *Markov Chains*) from abc notation to Json files.

5.2 Composing Statistics

As previously mentioned, before the melody creation, we had to compute some statistics which the program could used to understand how humans compose music. To be more general, every tune is transposed in *C major* and its relative modes (*A minor, D dorian or G mixolydian*). Thus, for a given mode, all tunes have the same key signature.

In order to produce its tunes, IRISH MELODY MAKER makes use of the following statistical resources:

1. MARKOVMEL3.JSON
   A JSON file representing a python dictionary. It contains all sequences of 4 notes for each style and mode as key, and their number of occurrences as value. It can easily be used to build a *3-degrees Markov Chain*.

2. RHYTHM.JSON
   A JSON file representing a PYTHON dictionary in which information how many times each rhythmic value appears for a given style is saved. It contains every rhythmic value and style as key, and their occurrences as value.

3. MARKOVRHYTHM.JSON
   A JSON file representing a PYTHON dictionary. It contains pairs of rhythmic values for each style as key and their number of occurrences as value. It can easily be used to build a *2-degrees Markov Chain*.

4. NOTEENDING.JSON
   A JSON file representing a PYTHON dictionary that contains for a given style and mode the count of how many times each combination of three notes ending a tune.

5. RHYTHMENDING.JSON
   A JSON file representing a PYTHON dictionary that contains for a given style the count of how many times each three rhythmic values combination appears.

5.3 How Irish Melody Maker works

The script GENERATE.PY is the entry point of IRISH MELODY MAKER. GENERATE.PY makes use of the COMPLETESHEET object, which is a high level object that knows form, style, rhythm, and mode of the melody that will be created and the version of the algorithm that will be used. Based on such information, COMPLETESHEET makes use of either SHEETPART_V1 or SHEETPART_V2 that respectively implement Algorithm 1 and Algorithm 2. It also makes use of the class ENDINGMAKER to create a proper ending to the tune. It creates a number of SHEETPART_V1 or SHEETPART_V2 object corresponding to the number of the tune has parts. The purpose of the SHEETPART objects is to generate a part of the melody, so, if the form is for example "aaba" there will be one

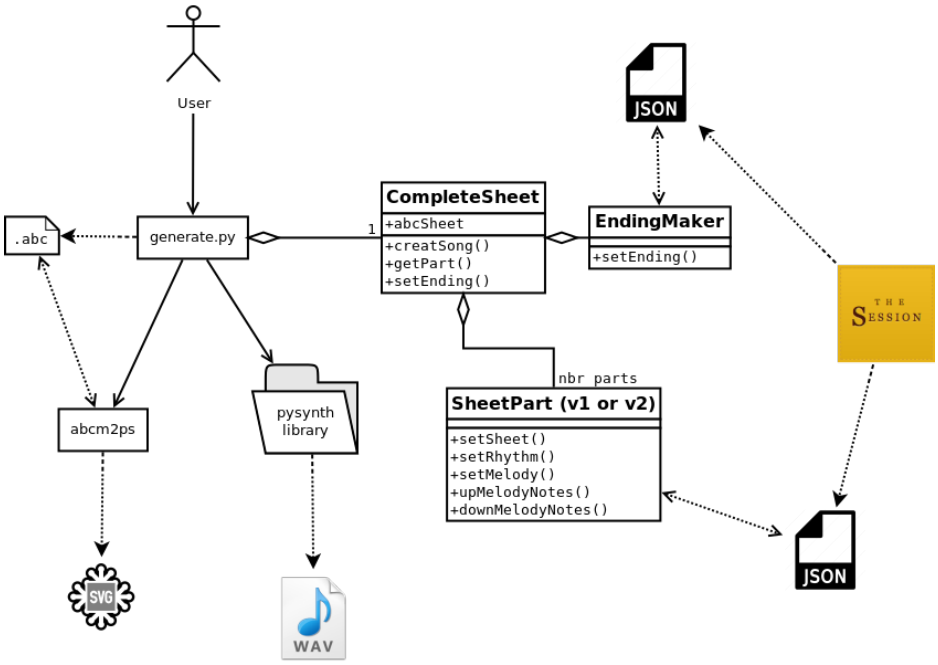

Figure 5: *Class diagram of the* Irish Melody Maker

SheetPart object to generate the part "a" and another to generate the part "b" (section 3 gives details on the subdivision of Irish tunes in parts). Every SheetPart object is constructed by loading the statistics files that it needs.

The CompleteSheet object calls the main method of each object SheetPart: setSheet(). This method works as follow: First, it calls the method setRhythmicPart() that generates the rhythm. Then it uses the setMelodicSheet() method to assign a pitch to every rhythmic figure composing the tune; for example, int *Figure 6* we can see that before the call of the setMelodicSheet() method the notes of the first measure of the tune (here a reel in c major scale) are just "c"'s.

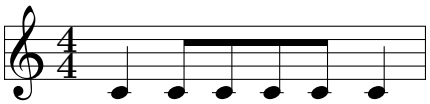

Figure 6: *First measure of a reel before the call of the* setMelodicSheet() *method*

Once the setMelodicSheet() method has been called, the melody is generated. We can see in *Figure 5.3* that pitches had been assigned to each note of

the first measure of the tune.

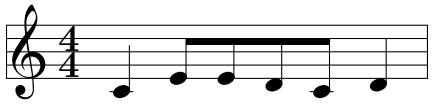

Figure 7: *First measure of a reel after the call of the* SETMELODICSHEET() *method*

This is the method of the SHEETPART objects called by SETMELODICSHEET() that return a note according to the 3 previous ones:

```
function getNote():
  if there is a note before the current position then
    lastNote = previous note
    if there is a note before lastNote
      lastNoteBefore = previous note of lastNote
      if there is a note before lastNoteBefore
        lastNoteBeforeBefore = previous note of lastNoteBefore
      end
    end
  end

  notes = array

  for each existant note
     if note is in the markov chain after lastNoteBeforeBefore +
    lastNoteBefore + lastNote
       i = 0
       while i < weight of the note in the markov chain * 10:
         add note in the array notes
         i += 1
       end
     end
  end

  return randomChoice(notes)
end
```

example1

DOWNNOTESMELODICSHEET() and UPNOTESMELODICSHEET() are the last methods to be called. As discussed in *Section 4.2.2*, they are responsible to decide when a note has to be shifted on a different octave than the previous one or note.

Once each part is generated, the COMPLETESHEET object writes the whole sheet sorting and copying every part to obtain the desired form. For example, if the tune has to have a "aaba" form, the "a" and the "b" parts have been created, the COMPLETESHEET object finally has to copy two times the part "a" and sorting every copy of each part to have a final sheet with the "aaba" form. Finally, the method SETENDING() of the ENDINGMAKEROBJECT is called. This will replace the ending of the tune by a most appropriate one using the related

statistics files.

Here is the main method of CompleteSheet, that call every other methods and objects to create the final sheet:

```
function createSong(style, mode, numerator, denominator)
  parts = array
  for each part of the needed form
    if the part hasn't been created yet
  add the returned value of getPart(style, mode, numerator,
    denominator) to the array parts
    end
  end

  setEnding(parts[lastpart])
  sheet = createAbcSheet(parts)

  return sheet
end
```

example2

5.4 Using Irish Melody Maker

To use IrishMelodyMaker, the user has to call the script generate.py with python. generate.py offers a console interface in which the user has to choose the style, mode and form of the melody and which one of the two algorithms that will be used to create it. Then, a *.wav* file is created in the folder provided for this purpose. *Figure 8* shows how the user can interact with the system.

```
$ python generate.py
choose a version of the algorithm (1 or 2) :
1
select one of following type of tune :
waltz, strathspey, hornpipe, slip jig, jig, three-two
, reel, polka, barndance, slide, mazurka
waltz
select one of following mode :
major, minor, mixolydian, dorian
aab
```

Figure 8: *An example of execution creating a simple major waltz melody.*

6 Evaluation

6.1 Crowedsourced Turing Test

The evaluation is a Turing test [5]. We randomly generated 80 melodies from each algorithm (20 jigs, slip jigs, reel and waltz). Then, we randomly selected 80 human composed melodies with the same style repartition. Each melody is displayed next to a randomly selected human composed melody of the same style.
The test works as follows: We show pairs of melodies to the testers in which one tune is composed by a human and the other one is composed by one of our algorithms. Testers can listen to the tunes and read their music sheet, based on that information they have to guess which is the tune composed by a human (an *I don't know* option is also available). Both tunes of each pair are selected randomly, so each test is different and every pair of computed/human tunes is possible. That means that one tester can try to pass the test more that once without corrupting the integrity of data. To have a better comprehension of the performance of our algorithms, we also asked testers whether they are musicians and how well they know Irish music.

6.1.1 Tools and Development

As shown by *Figure 9*, the Turing test is implemented as a Web application allowing us to find easily testers through various Web communities. The application is developed by using well-known Web tools and languages, PHP, HTML and SQL. Every test is saved as a SQL data on a MySQL database management system.

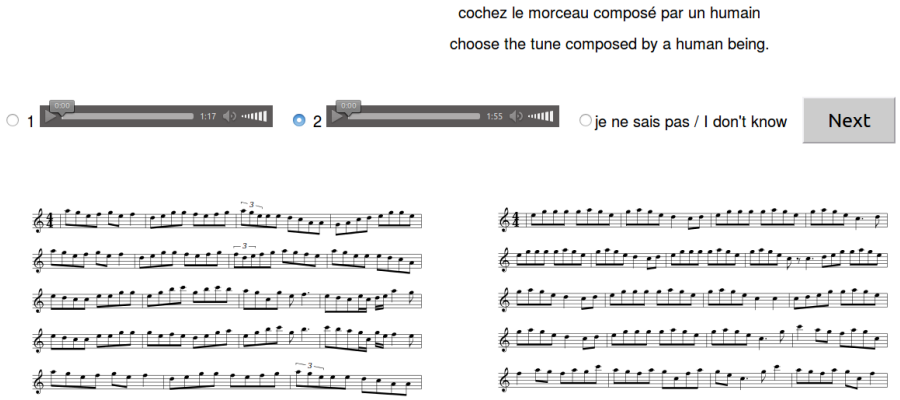

Figure 9: *An excerpt of the Turing test Web page.*

Figure 10 shows an excerpt of the data we collected, stocked as a SQL database. Each row represents the result of a test. We can see in the second

and third columns namely, “isMusician” and “isIrish” if testers are musicians and/or if they know Irish music well. Then, we have the style of the tested melody, the algorithm used to generate the synthetic one and the result of the test. An answer is considered correct when the tester guessed which tune was composed by a human being.

	id	isMusician	isIrish	style	algorithm	test	date
Edit Copy Delete	301	0	0	jig	2	1	2016-10-29 17:27:25
Edit Copy Delete	302	0	0	reel	1	0	2016-11-02 15:35:21
Edit Copy Delete	303	0	0	slip_jig	1	1	2016-11-04 09:55:02
Edit Copy Delete	304	0	0	slip_jig	2	1	2016-11-04 09:55:12
Edit Copy Delete	305	0	1	slip_jig	1	2	2016-11-04 12:49:34
Edit Copy Delete	306	1	2	reel	1	0	2016-11-04 14:17:50
Edit Copy Delete	307	1	2	slip_jig	2	0	2016-11-04 14:20:09
Edit Copy Delete	308	1	2	waltz	2	1	2016-11-04 14:22:15
Edit Copy Delete	309	0	2	waltz	1	0	2016-11-04 16:03:25

Figure 10: *An excerpt of the evaluation* SQL *data set.*

6.1.2 Results

The following results are based on 1319 tested pairs of tune. That means that each generated tunes is tested 8 times in average. A test is considered as *correct* if testers well guessed which was the human composed melody, conversely it is considered as *wrong* if they selected the algorithmically synthetic one. In every figure, the results reported are in percentage.

Which is interesting is that we can see in *Figure 11* and *Figure 12* that, excepting for reels, every styles have quite the same results. Furthermore, we can see that reel tunes computed with Algorithm 2 have a clearly better result (fewer people can find the correct answer) than the ones computed with Algorithm 1. We can also see that people testing tunes generated with Algorithm 2 had less hesitation than Algorithm 1 since they had less often selected the *I don't know* answers. Also, we can note that none of both Algorithm had pass the turing test.

However, if we look only at the tests passed by non musicians who do not know well Irish music, represented in *Figure 15*, we can see that Algorithm 2, contrary to Algorithm 1, passed the Turing test with a difference of 1.4%. We can verify the consistency of the results by looking at the results of musicians and people who does know well Irish music (*Figure 13* and *Figure 13*). We can clearly see here that there is a high difference between *wrong* and *correct*

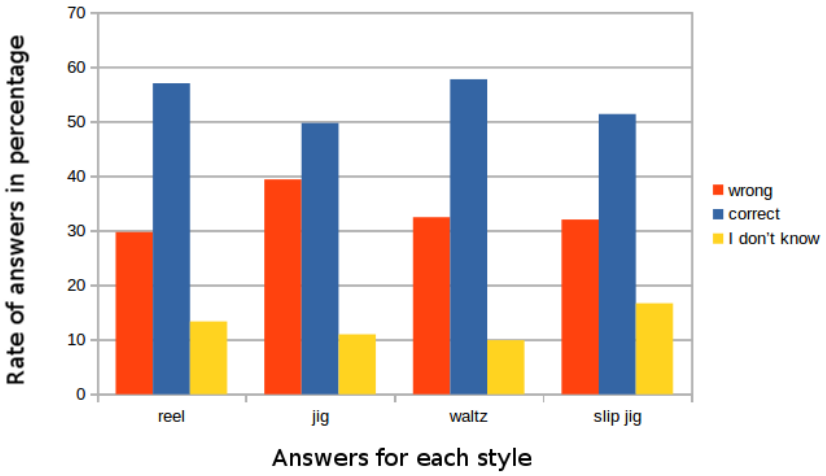

Figure 11: *results of tunes generated with Algorithm 1*

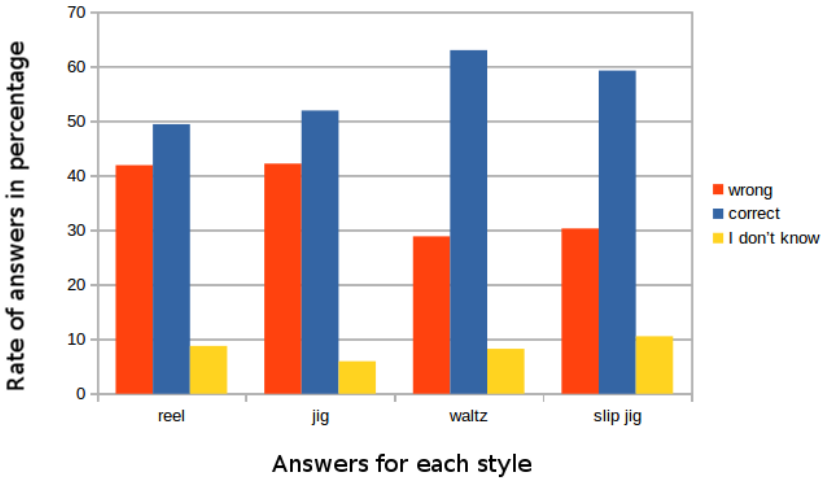

Figure 12: *results of tunes generated with Algorithm 2*

answers. Furthermore, we can see in every figure that Algorithm 2 has mostly better results than Algorithm 1.

We can also ask ourselves about the role of music sheets in the test. As a tester said to us after his test session:
"I play Irish music, I guess that's why I managed to get 9 out of 10 correct. But it seems to me that maybe the test is made a little easier by the fact that you can see the written music - the way it is written shows, quite often, if the tune is "real" or not. If you try guessing only through the audio (as I did on

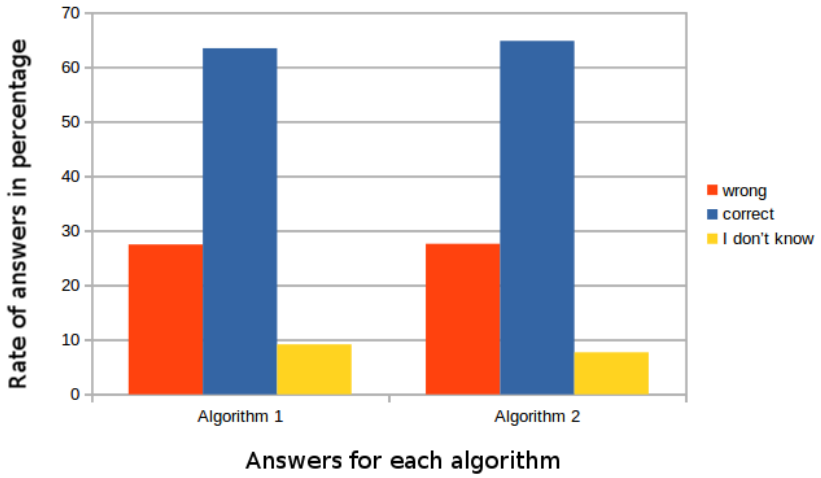

Figure 13: *results of all the tunes tested by people knowing well Irish music*

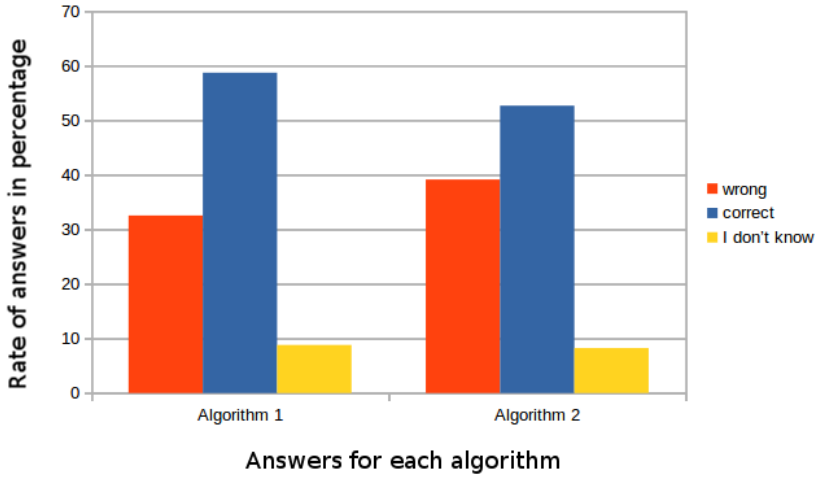

Figure 14: *results of all the tunes tested by musicians*

a few of the tunes) it's a lot harder to tell how it was produced, especially as the MIDI does not produce accents in any part of the music. That is, if you miss the first few notes of a tune, it's hard to tell where you are in the tune, because the "playing" does not mark the beats or beginning of bars." - Chris Jones

However, we can assume that the results from "normal" people are not influenced by the music sheet since non musicians can not read them.

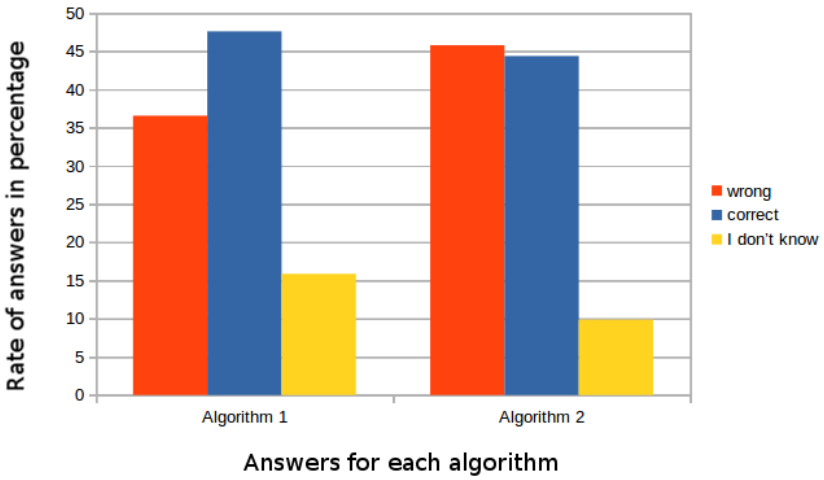

Figure 15: *results of all the tunes tested by non musician people that does not know well Irish music*

6.1.3 Complement of the Precision

To have a good idea of the differences between our two algorithms, we decided to compute the complement of the precision of the crowd, which we called $\bar{P}$. So, we computed two different complements $\bar{P}_1$ and $\bar{P}_2$. The first one takes into account the *I don't know answers*, the second one does not. The complement represent the ratio between the number of *wrong* answers (to which, depending if we compute $\bar{P}_1$ and $\bar{P}_2$ we add or not the *I don't know* answers) and the total number of answers. Let's w the number of *wrong* answers, k the number of *I don't know* answers and t the total number of tests, we have so $\bar{P}_1 = \frac{w+k}{t}$ and $\bar{P}_2 = \frac{w}{t}$. That means that if, for example, for given style and algorithm, we obtain a complement $\bar{P}_2 = 1$, for every test evaluating tunes of the given style generated by the given algorithm, the crowd gave the wrong answer. On the contrary, if we obtain a complement $\bar{P}_2 = 0$ every test was passed by the crowed. Let's remember that if the crowed pass correctly a test, that means that the algorithm fail.

Table 4: $\bar{P}_1$ *computed for each style of tunes and type of testers generated by Algorithm 1*

Tester	Reel	Jig	Waltz	Slip jig	All styles
normal	0.6	0.54	0.45	0.58	0.54
musician	0.36	0.53	0.45	0.44	0.45
Irish "expert"	0.32	0.37	0.33	0.4	0.35

Table 5: $\bar{P}_1$ *computed for each style of tunes and type of testers generated by Algorithm 2*

Tester	Reel	Jig	Waltz	Slip jig	All styles
normal	0.7	0.6	0.53	0.51	0.58
musician	0.41	0.44	0.31	0.41	0.4
Irish "expert"	0.4	0.39	0.25	0.29	0.34

Table 6: $\bar{P}_2$ *computed for each style of tunes and type of testers generated by Algorithm 1*

Tester	Reel	Jig	Waltz	Slip jig	All styles
normal	0.44	0.47	0.35	0.47	0.42
musician	0.31	0.46	0.41	0.38	0.39
Irish "expert"	0.29	0.29	0.29	0.28	0.29

Table 7: $\bar{P}_2$ *computed for each style of tunes and type of testers generated by Algorithm 2*

Tester	Reel	Jig	Waltz	Slip jig	All styles
normal	0.65	0.56	0.44	0.44	0.52
musician	0.38	0.43	0.29	0.35	0.36
Irish "expert"	0.35	0.33	0.21	0.22	0.28

6.1.4 Interpretation

As we can see in *Section 6.1.2*, Algorithm 2 is quite better than Algorithm 1. It even passed the turing test for "normal" people (non musicians who does not know well Irish music). It's interesting to note that Algorithm 2 is the one that is implementing *Markov Chains* to generate its rhythmic sheet while Algorithm 1 just uses simple statistics. We can conclude that this addition of complexity is effective to generated more "realistic" melodies. Furthermore, it's interesting to note that the results seem consistent since musicians and people who know well Irish music have quite better (they have more *correct* answers) than people who are not musicians and do not know well Irish music. That seems logical but it clearly means that the program has to be improved to be able to give a good imitation of human composition.

We can also see in *Tables 4, 5, 6, 7* that most of the time, the complement of Reels is greater than the complement of Jigs that is greater than the comple-

ment of Waltz. As we can see in *Table 1*, Reel is the style that is represented the most in the data set, that also contains a lot more of Jig tunes than Waltz tunes. Furthermore whether we compute the complement $\bar{P}_1$ or $\bar{P}_2$, the results are clearly higher for the tunes of Algorithm 2.

So, according to these results, we can say that Algorithm 2 is more effective than Algorithm 1 and that it is effective enough to fool "normal" people but not enough with muicians and people who know well Irish music (Let's remember that, not like Algorithm 1, Algorithm 2 use *Markov Chains* to generate the Rhythmic Sheet). We can also note that styles with the better results, Jigs and Reels, are both the most represented styles in the data set of TheSession.org. In conclusion, it seems quite logical that the program needs a minimum number of analysed data in order to get better results.

6.2 Expert Evaluation

We conducted a second evaluation of the algorithms by sending some of the tunes they generated to experts. For this purpose, we contacted five musicians working in different musical fields. We also compared Irish Melody Maker's algorithm to the algorithm used in the context of the endless traditional music session [9, 10], project which, as we will see in *Section 8.3* is based on the same data set.

Every expert received six tunes generated as follows:

- Tune 1: generated by algorithm 1 of Irish Melody Maker
- Tune 2: generated by the endless traditional music session
- Tune 3: generated by the endless traditional music session
- Tune 4: generated by algorithm 1 of Irish Melody Maker
- Tune 5: a music sheet from theSession.org [1]
- Tune 6: a music sheet from theSession.org [1]

Each expert was asked to rank the music sheets set from the "most human" tune to to "less human" tune.

6.2.1 Expert 1 – Christophe Dayer

Christophe Dayer is a master student in jazz music at the Bern University of the Arts.

1. Tune 3
2. Tune 5

3. Tune 6
4. Tune 2
5. Tunes 1 and 4

6.2.2 Expert 2 – Christian Zufferey

Christian Zufferey is a teacher of music theory and piano at EJMA-VS (School of Jazz and Contemporary Music of Valais). He has also obtained the best score in the first evaluation task (see *Section 6.1*), which is 11 correct answers out of 12 pairs of tunes.

1. Tunes 4, 5 and 6
2. Tune 3
3. Tune 1
4. Tune 2

He ranked Tunes 4, 5 and 6 tied at first position telling that they all were composed by a human being.

6.2.3 Expert 3 – Maryse Bétrisey

Maryse Bétrisey, who has a Master in Music Therapy from the Zürich University of the Arts, works at the neuropsychology department at Romande Rehabilitation Clinic.

1. Tune 5
2. Tune 3
3. Tune 6
4. Tune 2
5. Tune 1
6. Tune 4

She also specified that she thought Tunes 3, 5 and 6 were human, but she was not really sure about tune 6.

6.2.4 Expert 4 – Jean-Yves Zufferey

Jean-Yves Zufferey is an independent artistic manager and classical musician.

1. Tunes 4 and 6
2. Tune 5
3. Tune 1
4. Tune 3
5. Tune 2

He specified that he clearly thought Tunes 4 and 6 were composed by a human being.

6.2.5 Expert 5 – Baptiste Crettaz

Baptiste Crettaz, who has a Master in Electronic from EPFL (Polytechnic of Lausanne), works as research assistant for the acoustic section of LTS2 (signal processing laboratory) at EPFL. He is also an amateur musician of Irish music.

1. Tunes 5 and 6
2. Tune 4
3. Tune 3
4. Tune 2
5. Tune 1

He ranked Tunes 5 and 6 tied at first position telling they both were composed by a human being. And specified that he thought Tune 4 was a human composition but was not really sure.

6.2.6 Interpretation

A first important thing to note is that most of the time, every human composition is ranked as human composition. This proves that experts are able to recognise human compositions even though sometimes they have some doubt

Regarding the artificial compositions, Tunes 3 and 4 were often mistaken for human compositions, which happened more frequently for Tune 4. Conversely, Tunes 1 and 2 are ranked as artificial compositions quite at the same rate.

7 Future Work

7.1 Limitations of our Approaches

Irish Melody Maker generates melodies based on statistical data. That means that notes are chosen randomly (even if its choice is weighted according to statistical data). A big problem is the lack of a "main theme". In music theory, many tunes are composed around a simple melodic theme (easily remembered and sung) which is repeated and modified along the tune. Irish Melody Maker does not work with such a method. This means that it works well with short melodies, but it can not really create a consistent long tune.

Another less technical limitation is related to the fact that the program has no sense of music. Even if it creates a tune that respects its statistical data and rules, it does not know if the melody is interesting or not. Furthermore, none of its melodies are truly innovative. As the algorithms create their melodies based on many traditional tunes, they will always be cliché. However, this is out of the scope of this project since its main goal is to imitate human creations and not to generate new type of music.

7.2 Possible Improvements

A first improvement could simply be to include more composition rules in the algorithms. Certainly, it would work less with statistical data, like *Markov Chains*, but it could work on a simple theme that it could evolve. We believe that the algorithm will surely be better with a given style of music but it will be more difficult to extend it to other styles.

Another improvement could be to analyse how melody and rhythm influence each other. It could be interesting if the algorithms took into account this correlation and try to match them together instead of generating them separately.

Alternatively, the program could be extended by improve to a global understanding of how tunes are created, that is, how many parts has a tune, how each one is composed relative to the others, and more. This could be done statistically or by using composition rules. This approach might be able to generate tunes that are longer and more interesting.

Nevertheless, using very specific rules to generate melodies could limit the generalisability of the methods, whereas, by using statistics it is easy to extend the program to other or new styles. The interest of statistics is that it is always possible to add data or create a new data set corresponding to the other sources.

8 Related Works

8.1 Kulitta

Kulitta [6, 7] is a framework implemented in haskell and python. It can be used both as a framework to develop music composition programs and as a full music generating program. Kulitta was part of Donya Quick's Ph.D. research

at Yale University in 2014. The system works at different level of abstraction by using probabilistic temporal graph grammars, a new category of grammar that was developed for music generation by Donya Quick herself.

8.2 Emily Howell

Emily Howell [8] is one of the first effective programs for music generation. Its development started in the 90's by David Cope, who works for University of California Santa Cruz tried to create an artificial intelligence with its own musical personality. emily howell is often presented as a real composer with her own way to compose and she has also released some albums under her own name, this makes the approach philosophically interesting. The program is mostly based on latent semantic analysis.

8.3 The Endless Traditional Music Session

The Endless Traditional Music Session [9, 10] is the most interesting work related to our project, mainly because it has the same purpose of the Irish Melody Maker and it uses the same data base (the data of theSession.org [1]). The program was developed by Bob L. Sturm and João Felipe Santos. Like ours, their program was developed in Python, too. However, unlike the algorithms of Irish Melody Maker, it uses Recurrent Neural Networks [11] to generate the melodies. Furthermore, they have a funny concept: the program generates seven new tunes every five minutes. Thus, every five minutes, we can find seven new tunes on their website.

Further, thanks to the experts evaluation, both algorithms seem equally effective on melody generation, even though at the very beginning their algorithm seemed better than ours.

9 Conclusion

According the data collected during our experimental evaluations, we can conclude that melody according to generation with *Markov Chain* is a good starting point for automatic composition. Melodies are easily mistaken for human composition by people who have limited knowledge of Irish music. But a *Markov Chain* generation alone is not enough to confuse experts. For that purpose, the algorithm has to be improved by using a higher level layer that manages to create recurrent melody schemes and patterns. It needs to use not only statistics but also melodic and harmonic composing rules. But even if an algorithm can fool every human that way, it will always compose a tune that belongs to a pre-existing human music style. The next step would be to create an algorithm that can find its own style and composing rules. And then, there will be one last question to answer: who will own the copyrights?

References

[1] *thesession.org* website of traditional Irish music data base

[2] Gareth Loy, *The Mathematical Foundations of Music, volume 1*. The MIT Press, Massachusetts, 2006.

[3] Martin C. Doege, *pysynth*. https://mdoege.github.io/PySynth/, python library to read abc notation and create sound files.

[4] David Fuentes, *Figuring Out Melody*. 2010.

[5] Alan M. Turing, *Computing Machinery and Intelligence*. 1950

[6] Donya Quick, *Kulitta: a Framework for Automated Music Composition*. Yale University, 2014.

[7] Donya Quick, *donyaquick.com*.

[8] David Cope, *Experiments in Musical Intelligence*. University of California Santa Cruz

[9] Bob L. Sturm, *The Infinite Irish Trad Session*. High Noon GMT, Queen Mary University of London, 2015

[10] Bob L. Sturm & João Felipe Santos, *The Endless Traditional Music Session*. http://www.eecs.qmul.ac.uk/ sturm/research/RNNIrishTrad/

[11] Andrej Karpathy, *The Unreasonable Effectiveness of Recurrent Neural Networks*. Andrej Karpathy blog, 2015.