The Song Search

Introduction

- The aim of our project is to build an Information Retrieval system for music.
- This is similar to shazam.
- We curated a sample dataset and queries from GTZAN & other audio clips from online.
- The core of this IR system comes from finding efficient representations for songs and performing a retrieval task with the said representations.
- We were intrigued by the Tensorflow Magenta teams music transcription models.
- These models transcribe songs into MIDI
- We experimented with a couple of these models.

What's MIDI?

- MIDI is a communication standard that allows digital music gear to speak the same language.
- MIDI is short for Musical Instrument Digital Interface. It's a protocol that allows computers, musical instruments and other hardware to communicate.

Methodologies

- Experimented with onset and frames model for pianos
- Experimented with onset and frames model for drums
- Finally decided to use the MT3 model finally for multi instrument audio files.

Onset & Frames

The dual objectives are learned on each stack:

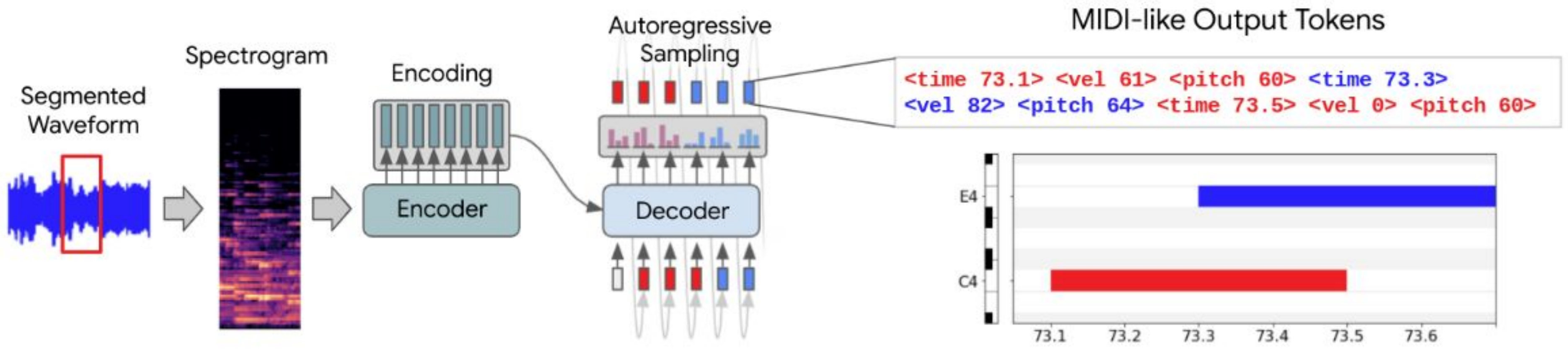
- Frame Prediction: Trained to detect every frame where a note is active.
- Onset Prediction: Trained to detect only onset frames (the first few frames of every note)

- Each instrument needs a new arch.
- Tedious to build custom arch for each instrument.

MT3

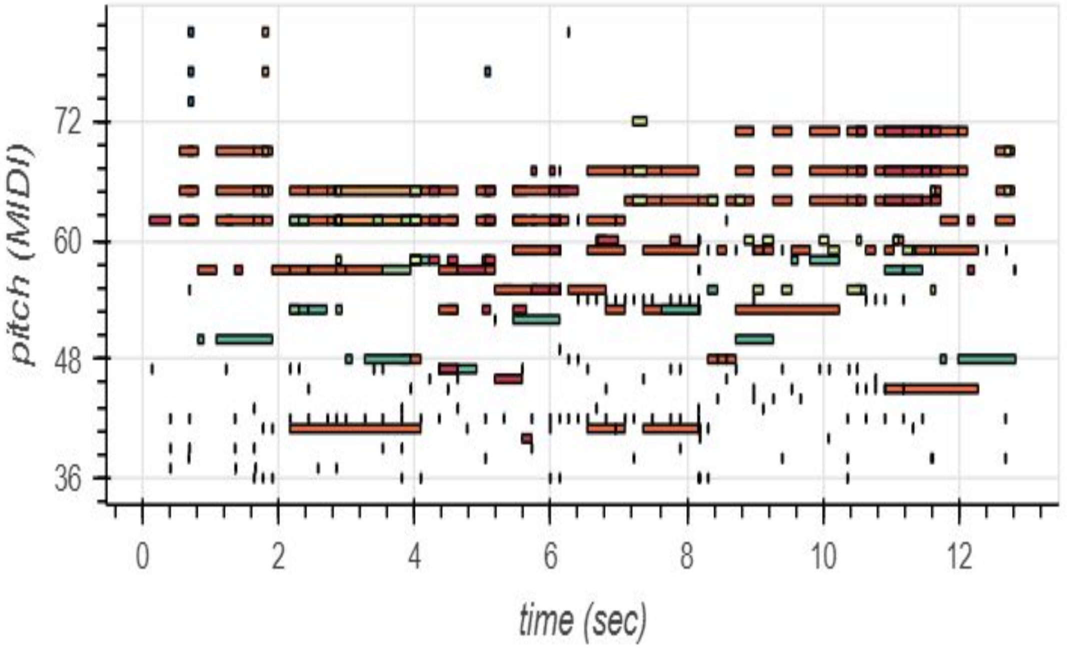
- MT3 stands for Multi-Task Multi-track Music Transcription
- It uses off-the-shelf transformers, as they work well if not better than custom neural networks as we had seen for Piano/Drums. They are modeled to take spectrograms as input and output a sequence of MIDI-like note events.
- They modeled it as a sequence-to-sequence task, using the Transformer architecture from T5.
- Further, to retrain this architecture for newer instruments, we would only need to change the vocabulary of the output.

MT3


Segmented Waveform
Spectrogram
Encoding
Encoder
Autoregressive Sampling
Decoder
MIDI-like Output Tokens
E4
C4
73.1
73.2
73.3
73.4
73.5
73.6

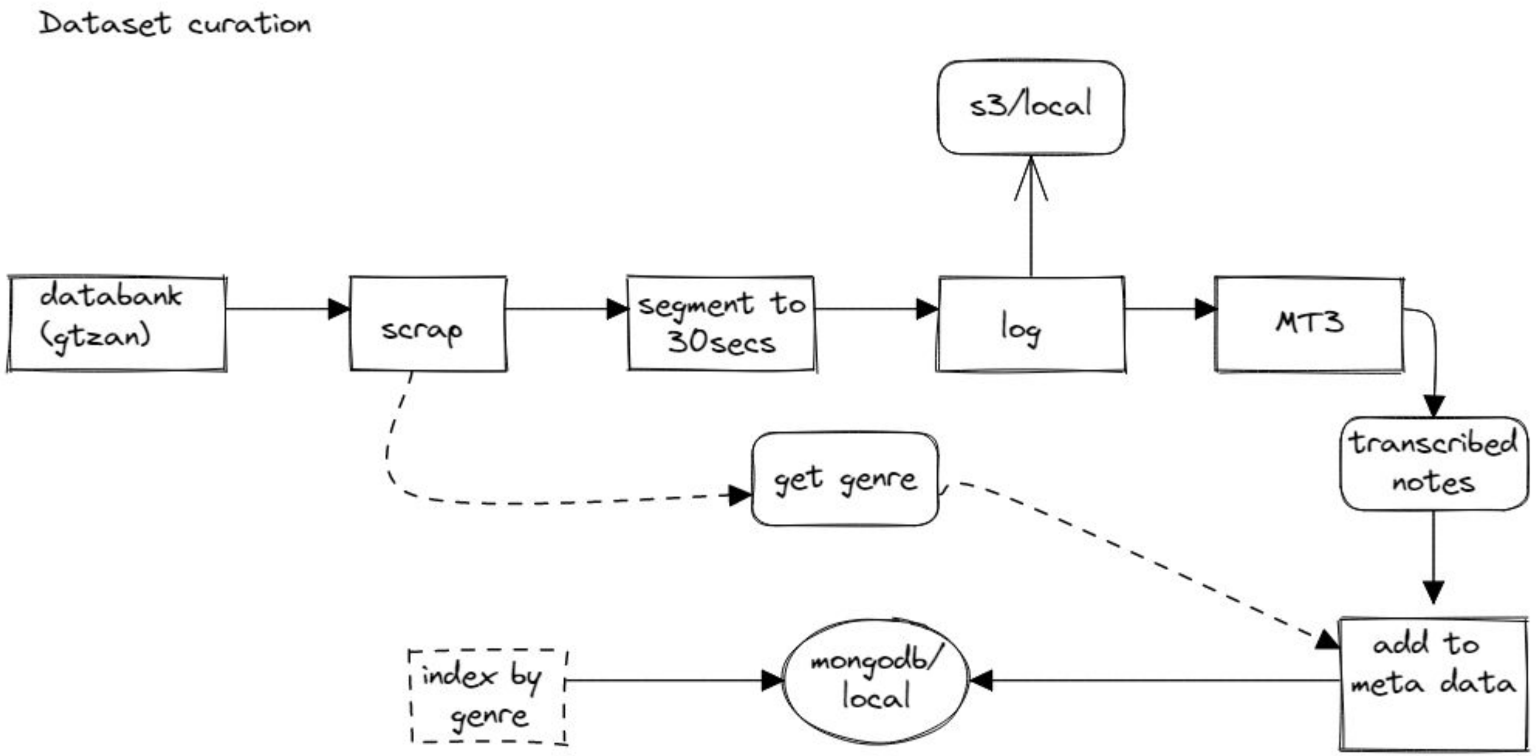
<time 73.1> <vel 61> <pitch 60> <time 73.3>
<vel 82> <pitch 64> <time 73.5> <vel 0> <pitch 60>

MT3 output

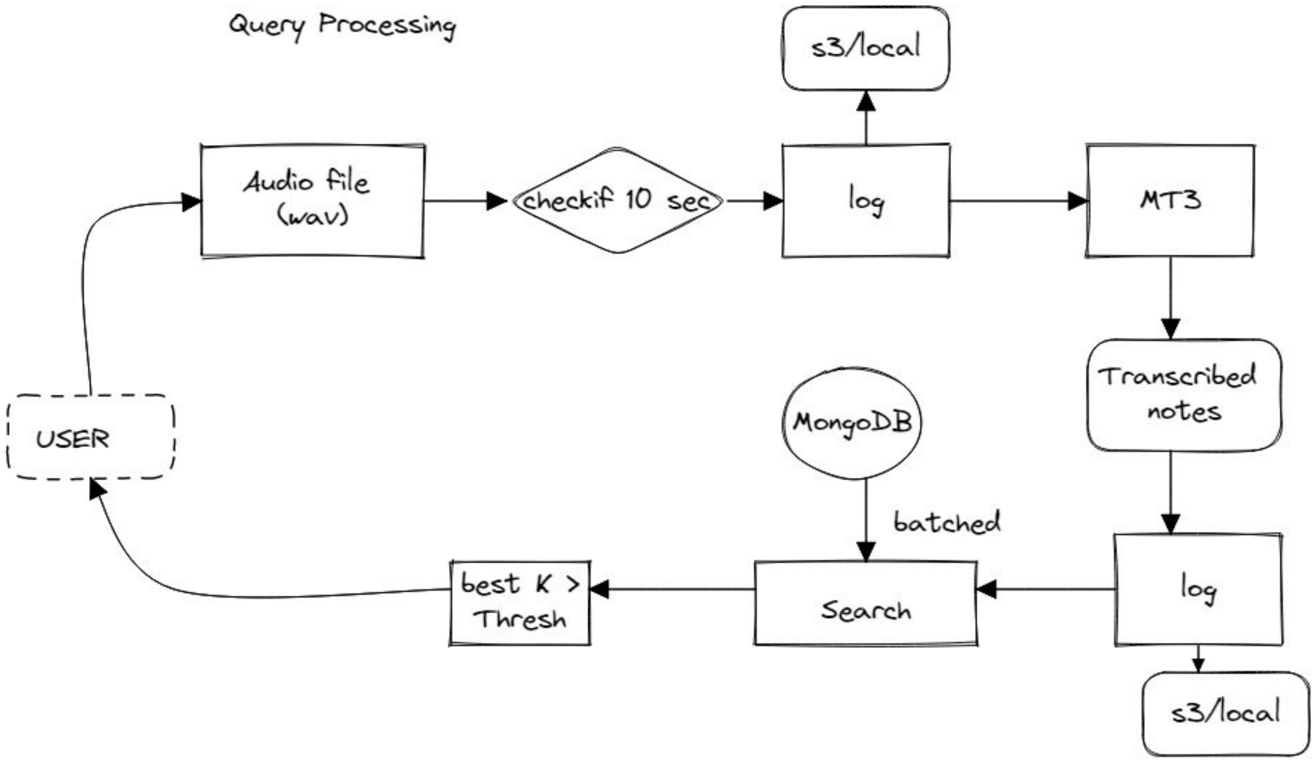


```
notes {
  pitch: 40
  velocity: 127
  start_time: 3.33
  end_time: 3.45
  instrument: 2
  program: 42
}
notes {
  pitch: 39
  velocity: 127
  start_time: 3.45
  end_time: 3.56
  instrument: 2
  program: 42
}
```

Arch diagram



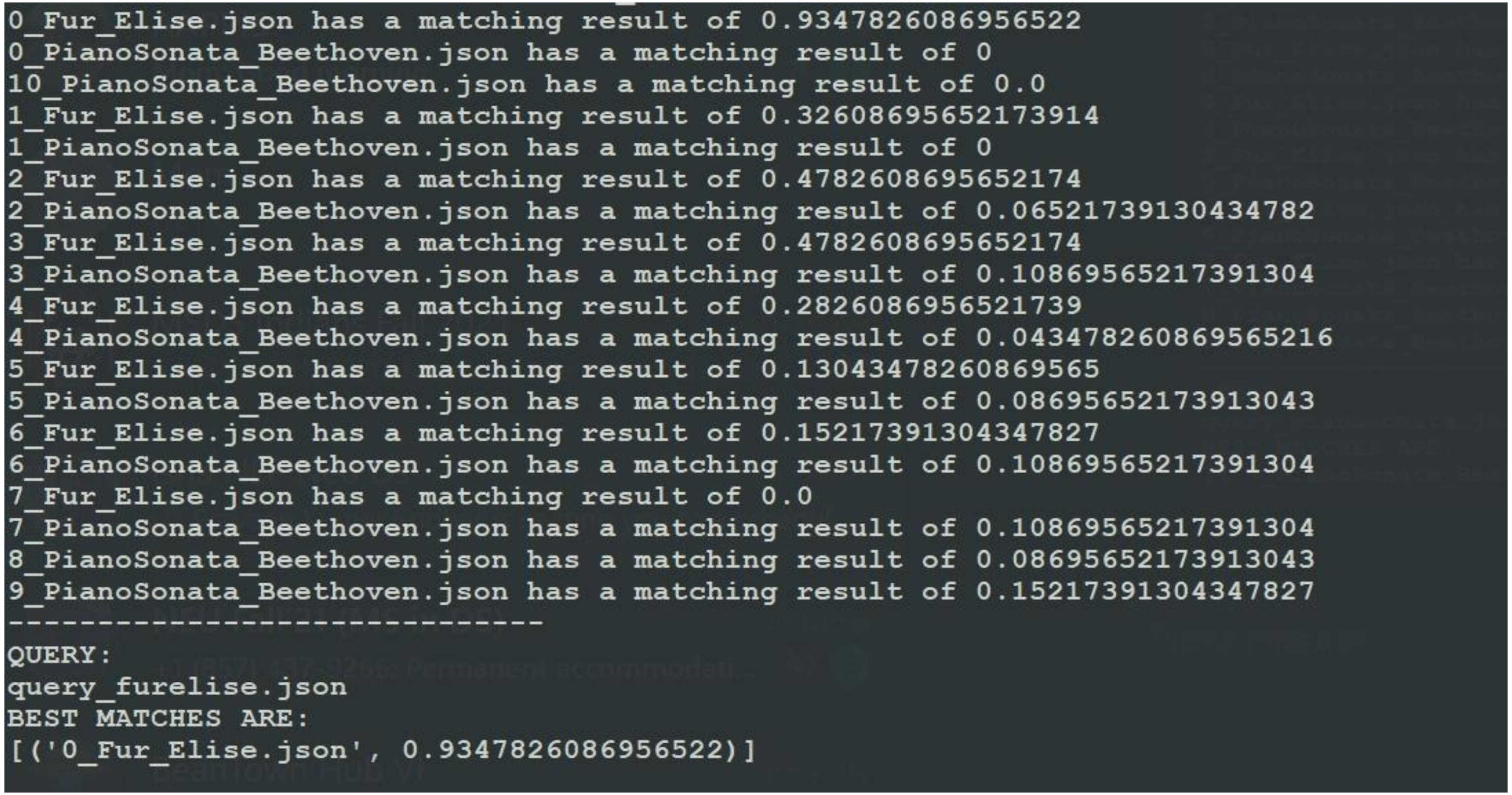
Arch Diagram (cont.)



Results

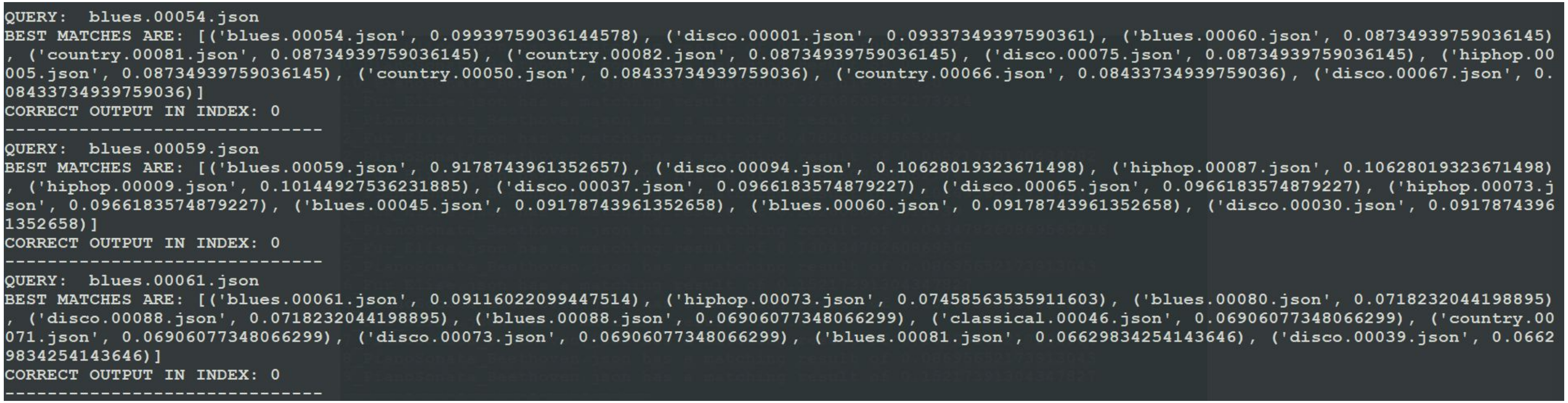
- MT3 has an average frame F1 score of 0.85 across different datasets
- MT3 has an average onset F1 score of 0.8 across different datasets
- The IR system with the MT3 model has an overall accuracy of 74% in the top 5 candidate set and a MAP of 0.68.

Results (cont.)



```
0_Fur_Elise.json has a matching result of 0.9347826086956522
0_PianoSonata_Beethoven.json has a matching result of 0
10_PianoSonata_Beethoven.json has a matching result of 0.0
1_Fur_Elise.json has a matching result of 0.32608695652173914
1_PianoSonata_Beethoven.json has a matching result of 0
2_Fur_Elise.json has a matching result of 0.4782608695652174
2_PianoSonata_Beethoven.json has a matching result of 0.06521739130434782
3_Fur_Elise.json has a matching result of 0.4782608695652174
3_PianoSonata_Beethoven.json has a matching result of 0.10869565217391304
4_Fur_Elise.json has a matching result of 0.2826086956521739
4_PianoSonata_Beethoven.json has a matching result of 0.043478260869565216
5_Fur_Elise.json has a matching result of 0.13043478260869565
5_PianoSonata_Beethoven.json has a matching result of 0.08695652173913043
6_Fur_Elise.json has a matching result of 0.15217391304347827
6_PianoSonata_Beethoven.json has a matching result of 0.10869565217391304
7_Fur_Elise.json has a matching result of 0.0
7_PianoSonata_Beethoven.json has a matching result of 0.10869565217391304
8_PianoSonata_Beethoven.json has a matching result of 0.08695652173913043
9_PianoSonata_Beethoven.json has a matching result of 0.15217391304347827
------------------------------
QUERY:
query_furelise.json
BEST MATCHES ARE:
[('0_Fur_Elise.json', 0.9347826086956522)]
```

Results (cont.)



```
QUERY:  blues.00054.json
BEST MATCHES ARE: [('blues.00054.json', 0.09939759036144578), ('disco.00001.json', 0.09337349397590361), ('blues.00060.json', 0.08734939759036145)
, ('country.00081.json', 0.08734939759036145), ('country.00082.json', 0.08734939759036145), ('disco.00075.json', 0.08734939759036145), ('hiphop.00
005.json', 0.08734939759036145), ('country.00050.json', 0.08433734939759036), ('country.00066.json', 0.08433734939759036), ('disco.00067.json', 0.
08433734939759036)]
CORRECT OUTPUT IN INDEX: 0
------------------------------
QUERY:  blues.00059.json
BEST MATCHES ARE: [('blues.00059.json', 0.9178743961352657), ('disco.00094.json', 0.10628019323671498), ('hiphop.00087.json', 0.10628019323671498)
, ('hiphop.00009.json', 0.10144927536231885), ('disco.00037.json', 0.0966183574879227), ('disco.00065.json', 0.0966183574879227), ('hiphop.00073.j
son', 0.0966183574879227), ('blues.00045.json', 0.09178743961352658), ('blues.00060.json', 0.09178743961352658), ('disco.00030.json', 0.0917874396
1352658)]
CORRECT OUTPUT IN INDEX: 0
------------------------------
QUERY:  blues.00061.json
BEST MATCHES ARE: [('blues.00061.json', 0.09116022099447514), ('hiphop.00073.json', 0.07458563535911603), ('blues.00080.json', 0.0718232044198895)
, ('disco.00088.json', 0.0718232044198895), ('blues.00088.json', 0.06906077348066299), ('classical.00046.json', 0.06906077348066299), ('country.00
071.json', 0.06906077348066299), ('disco.00073.json', 0.06906077348066299), ('blues.00081.json', 0.06629834254143646), ('disco.00039.json', 0.0662
9834254143646)]
CORRECT OUTPUT IN INDEX: 0
------------------------------
```

Future Work

- We realize that the project did not include songs that contained lyrics as we were unable to parse and transcribe them in a way that was particularly useful.
- As a future extension, we hope to be able to extend this project of matching songs to include songs with lyrics.
- Also we want to swap out the Robin-Karp based notes-matching algorithm for a Smith-Waterman based notes-matching algorithm.

Thank You