Emotion Patterns in Music Playlists

Sara Giammusso[1][2] Mario Guerriero [1][2]

[1]MSc student in Data Science Department, EURECOM, Télécom ParisTech, France
[2]MSc student in Department of Control and Computer Engineering, Politecnico di Torino, Italy

Second Project meeting

Table of Contents

Table of Contents

Introduction

Previously On Sara&Mario Project...

In the previous meeting we analyzed the **state-of-the-art** of text-based Emotion Detection.

Next steps:

- Analyze **existent** emotion classifiers
- Research text corpus with **labeled** emotions
- Natural language **processors** and **embedders**
- Statistics and **details** about MoodyLyrics

Table of Contents

Emotion classifiers analysis

The emotion classifiers APIs we analyzed are:

1. IBM Watson NLU
2. IBM Watson Tone Analyzer
3. ParallelDots AI
4. Qemotion

1) IBM Watson: Natural Language Understanding (I)

Watson is a **question answering computer** system capable of answering questions posed in **natural language**, developed by IBM.[2]

Cool story

In 2011, the Watson computer system competed on Jeopardy! against legendary champions Brad Rutter and Ken Jennings winning the first place prize of $1 million [2].

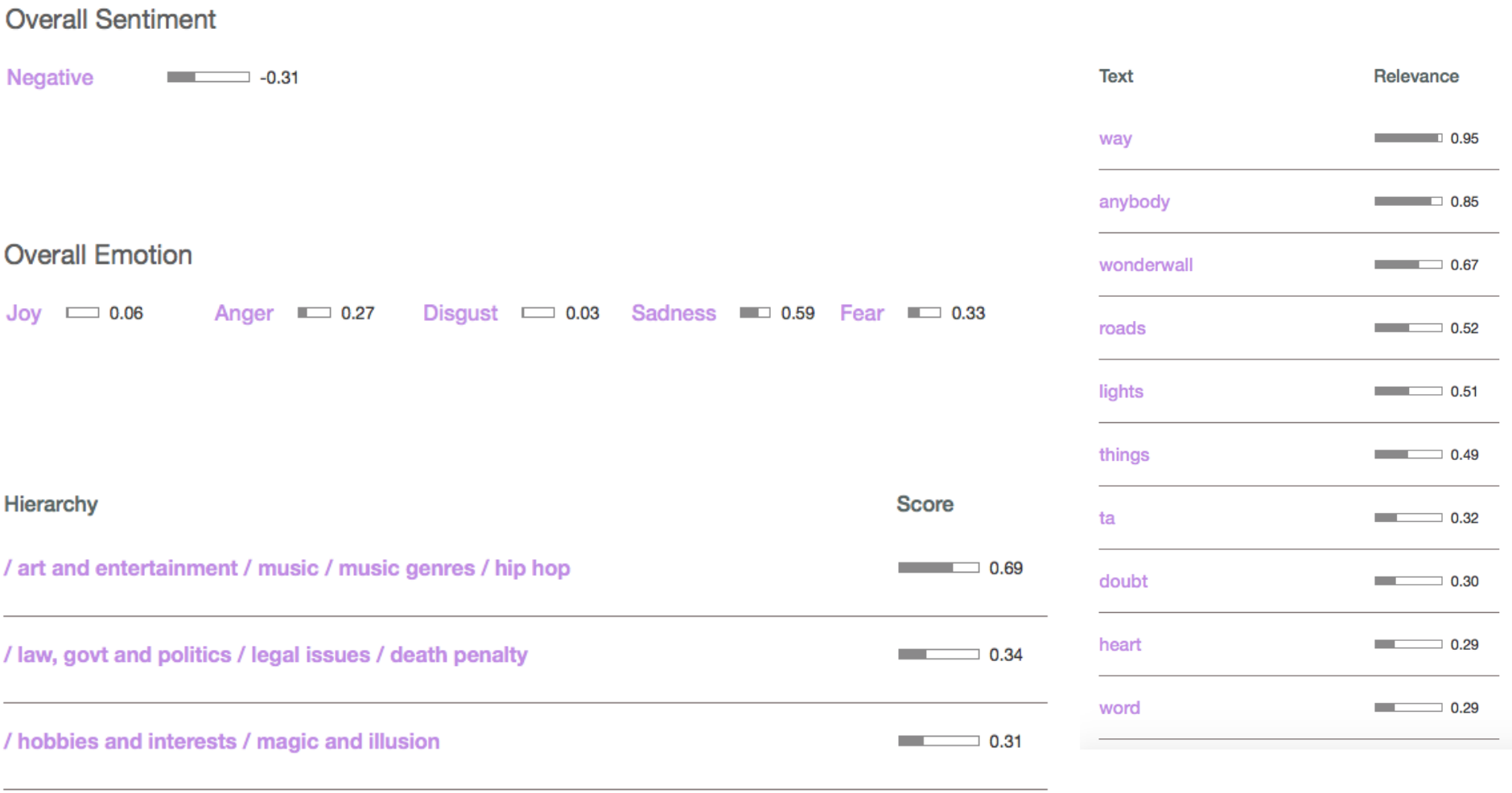
1) IBM Watson: Natural Language Understanding (II)

Natural Language Understanding is a collection of **APIs** that allows to:[1]

- Recognize the **overall sentiment**, in a scale from negative to positive [-1,1];
- Detect the **emotion percentage** between: joy, anger, disgust, sadness, fear;
- Determine **keywords** ranked by relevance;
- Extract **entities**: people, companies, organizations, cities and other information;
- Classify content into a **hierarchical categories**;
- Identify **general concepts** that may not be directly referenced in the text;
- Distinguish the **semantic roles** parsing sentences into subject, action and object.

1) IBM Watson NLU: Demo (I)

Results obtained analyzing **Oasis** - **Wonderwall** lyrics (I).



1) IBM Watson NLU: Demo (II)

Results obtained analyzing **Oasis - Wonderwall** lyrics (II).

Concept	Score
2008 singles	0.95
2009 singles	0.91
2005 singles	0.75
Billboard Alternative Songs number-one singles	0.74
Number-one singles in New Zealand	0.74
Journey	0.65
Wonderwall	0.64
English-language films	0.63

Name	Type	Score
Backbeat	Company	0.86

That they 're gonna throw it back to you

Object *Subject*

2) IBM Watson: Tone Analyzer

It uses linguistic analysis to detect joy, fear, sadness, anger, analytical, confident and tentative tones found in text. [3]

Possible sources

Tweets, Online Review, Email message, your own text.

It uses both:

- **the document level**: to get a sense of the overall tone
- and the **sentence level**: to identify specific areas of your content where tones are the strongest.

The results obtained with **Oasis - Wonderwall** are identical to the ones obtained from **IBM Watson: NLU**

3) ParallelDots APIs: Demo

Their **Emotion Analysis classifier** is trained on their proprietary dataset and tells whether the underlying emotion behind a message is: **Happy, Sad, Angry, Fearful, Excited, Funny or Sarcastic**.[4]
The result obtained analyzing **Oasis - Wonderwall** lyrics is showed in the following figure.

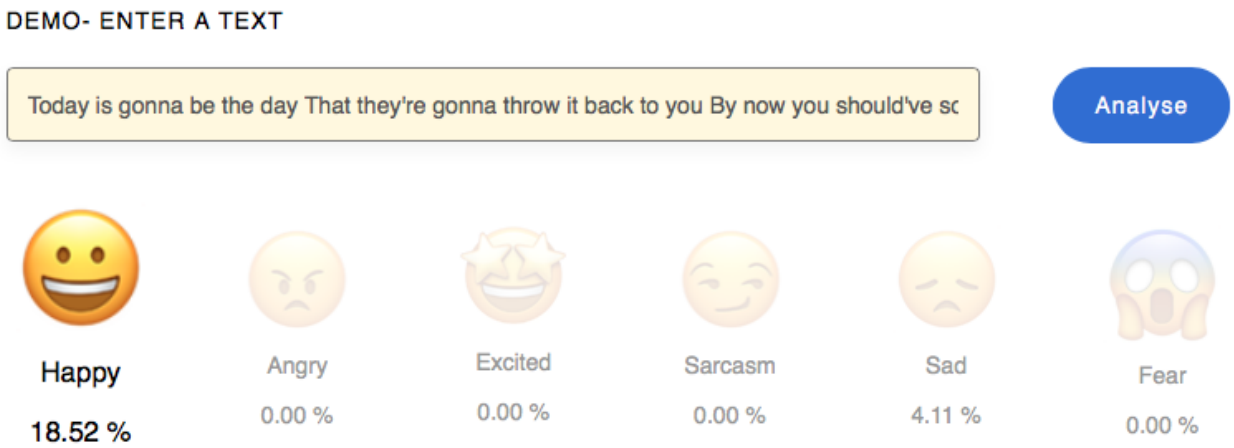


Figure 1: Output for Oasis - Wonderwall

4) Qemotion

Qemotion detects the main emotion of the speech and will define the corresponding emotion in terms of **temperature** (literally temperature) [5].

- From 31°C to 40°C $\rightarrow$ Happiness
- From 21°C to 30°C $\rightarrow$ Surprise
- From 11°C to 20°C $\rightarrow$ Calm
- From 6°C to 10°C $\rightarrow$ Fear
- From -5°C to 5°C $\rightarrow$ Sadness and Disappointment
- From -14°C to -6°C $\rightarrow$ Anger
- From -20°C to -15°C $\rightarrow$ Disgust

Note about plans

Note

None of these APIs is free, however they offer a limited number of free API calls

Table of Contents

NLP libraries

The main NLP libraries are:

1. NLTK
2. TextBlob
3. Stanford's CoreNLP
4. SpaCy

1) NLTK: Natural Language Toolkit

Recommend only as an **education and research** tool.
Pros:

- its **modularized** structure makes it excellent for learning and exploring NLP concepts;
- over 50 corpora and lexicons, 9 stemmers, and dozens of algorithms to choose from (also a con);

Cons:

- **Heavy** and it has a **steep** learning curve;
- **slow** and not production-ready.

See also **NTLK Book** [6].

2) TextBlob

Built on top of NLTK.
Pros:

- More intuitive;
- Gentle learning curve.

See also **"NLP Basics with TextBlob"** [7].

3) Stanford's CoreNLP

Java library with Python wrappers.
Pros:

- fast;
- support for several major languages.

4) SpaCy

It's a new NLP library designed to be **fast**, streamlined, and production-ready.
Pros:

- **minimal**: it doesn't flood you with a lot of options
- its philosophy is to only present **one algorithm** (the best one) for each purpose.

Cons:

- it's **new**, so its support community is not as large as some other libraries
- it currently only supports English.

See also **"Intro to NLP with SpaCy"** [8].

Word Embeddings

Word embeddings are a set of feature learning techniques which map **words** or phrases from the vocabulary to vectors of **real numbers**. These techniques map **sparse** word vectors into **continuous** space based on the surrounding context.

Example

If "**"salt"** and **"seasoning"** appear within the same context, the model will indicate that "salt" is conceptually closer to "seasoning," than, say, "chair."

There are 2 main Embedding libraries:

1. Word2Vec
2. FastText

Word2Vec VS FastText (I)

Main difference

Word2vec treats each word in corpus like an **atomic entity** and generates a vector for each word.

FastText treats each as **composed** of character ngrams, so the vector for a word is made of the sum fo this character n grams.

Example

The word vector "apple" is a sum of the vectors of the n-grams "ap", "app", "appl", "apple", "ppl", "pple", "pple", "ple", "le" (assuming hyperparameters for smallest ngram[minn] is 3 and largest ngram[maxn] is 6).

Word2Vec VS FastText (II)

This difference manifests as follows:

1. **Rare words**: even if words are rare their character n grams are still shared with other words - hence the embeddings with FastText can still be good.
2. **Out of vocabulary words**: FastText can construct the vector for a word from its character n grams even if word doesn't appear in training corpus.
3. **Hyperparameters choice**: FastText requires to choose the the minimum and maximum n-gram sizes, and this directly impacts:
   - computation time
   - memory requirements

Table of Contents

lyrics_downloader.py (1)

We wrote a Python script for downloading lyrics. We used:

- MoodyLyrics to get songs information (artist, title and emotion)
- LyricWikia to download the lyrics

lyrics_downloader.py (2)

Our script produces in output:

- A folder containing lyrics in files named: *EMOTION_ARTIST_TITLE-OF-SONG*
- A log file in which we keep track of the errors we found

Table of Contents

References

[1] IBM Watson: Natural Language Understanding APIs
https://www.ibm.com/watson/services/natural-language-understanding/

[2] IBM Watson - Wikipedia
https://en.wikipedia.org/wiki/Watson_(computer)

[3] IBM Watson: Tone Analyzer
https://en.wikipedia.org/wiki/Watson_(computer)

[4] ParallelDots APIs https://tone-analyzer-demo.ng.bluemix.net/

[5] Qemotion http://www.qemotion.com/demo.php

[6] NTLK Book http://www.nltk.org/book/

[7] Natural Language Basics with TextBlob
http://rwet.decontextualize.com/book/textblob/

[8] Intro to NLP with SpaCy
https://nicschrading.com/project/Intro-to-NLP-with-spaCy/