Alma Mater Studiorum · Università di Bologna

DIPARTIMENTO DI INTERPRETAZIONE E TRADUZIONE
Corso di Laurea magistrale in Specialized Translation (classe LM-94)

TESI DI LAUREA
in COMPUTATIONAL LINGUISTICS

Emotion Identification in Italian Opera

CANDIDATA:
Zhang Shibingfeng

Relatore:
Alberto Barrón Cedeño

Correlatore:
Adriano Ferraresi

Anno Accademico 2020/2021
Primo Appello

Acknowledgements

I would like to express my gratitude towards my parents, Shi Xia (史霞) and Zhang Xiaozhong (张晓中), who have supported me with their love over the years. We still cannot meet due to the pandemic, but I believe the day for me to come home is just around the corner. I miss you immensely.

I would also like to express my gratitude to my boyfriend Andrea de Rose, his parents Alberto de Rose, Lucia Macrelli and the neighbor's cat Luna. Your company has supported me through my study.

In addition, I would like to thank my thesis supervisor, Alberto Barrón Cedeño, who introduced me to the field of Computational Linguistics. You opened me a gate towards new adventures. I would also like to thank professor Adriano Ferraresi for his guidance and support. Thanks also to Francesco Fernicola, Federico Garcea, and Paolo Bonora. Without your contributions to the AriEmozione project this dissertation would not have been possible.

I would also like to thank the flying ducks and swimming chickens of Resistenza park that have brought so much fun into my life. Thank you to Crem Caramel for the frozen yogurt, it is such a joy to have a wonderful gelateria like you near my house. Thank you to Yu Xinxin (余欣欣) and Zhong Yubin (钟宇彬), your friendship is priceless to me.

Thank you all.

致谢

首先，我想表达我对我的母亲史霞和父亲张晓中的感激之情。谢谢你们这些年来对我的支持与包容。我们虽然由于疫情还不能见面，但是我相信我回家的那天指日可待。我无比思念你们。

我还想表达我对我男朋友Andrea de Rose，他的父母Alberto de Rose, Lucia Macrelli 及邻居家的猫Luna的感谢。你们的陪伴支持着我完成了学业。

另外，我还想感谢我的论文指导老师Alberto Barrón Cedeño，他领我进入了计算语言学的大门。感谢Adriano Ferraresi教授对我学业的指导。感谢Francesco Fernicola, Federico Garcea和Paolo Bonora，你们对AriEmozione项目做出的贡献促成了这篇论文的诞生。

感谢Resistenza公园会飞的鸭子和会游泳的鸡，你们为我带来了许多快乐。感谢Crem Caramel的冻酸奶，在家附近有一家冰淇淋店是多么幸福的事。感谢余欣欣和钟宇彬，你们的友情对我来说弥足珍贵。

谢谢大家。

Abstract

This work aims to develop classification models able to automatically perform the task of emotion identification on Italian arias. These models enable the musicologists and the public interested in opera to investigate the emotion of Italian aria in a systematical way.

An aria can be seen as an independent unit of opera that is sung by one character. Each aria contains 1 to 8 verses. Considering an aria may transmit more than one emotion, a lower level granularity is adopted: the identification of the emotion transmitted at the verse level. On the basis of a manually labelled corpus comprised of 2,500 aria verses with their corresponding emotion, the first part of this work investigates different text representations and classification approaches.

Building on the results of the exploration in the first part, the second part investigates emotion identification at the aria level. The size of supervised data is expanded by means of self-learning[1]. The verse-level annotation is converted into aria-level annotation and each aria is assigned up to two emotion labels. I experimented with pre-trained character trigram embeddings and convolutional neural network.

For the emotion identification at the verse level, the combination of character trigram based TF-IDF and neural network with 2 hidden layers outperformed other combinations, achieving an accuracy of 0.47 on the test set. As for the emotion identification at the aria level, a convolutional neural network combined with character trigram based embeddings developed based on a corpus of Italian arias achieved an accuracy of 0.68.

My contributions in this work are following:

1. The annotation of the AriEmozione 2.0 corpus;

[1]Self-learning refers to the process of training a model on a supervised data set that is manually annotated and using such model to predict unknown instance as a means of expanding the train set (Jurkiewicz et al., 2020)

2. The conversion of annotation granularity from the verse-level to aria-level;
3. The creation of pre-trained 3-gram based vectors using the CORAGO-1700 corpus;
4. The creation of models capable of identifying emotion of Italian operas, both at verse-level and aria-level;

Contents

Chapter 1

Introduction

This dissertation explores how computational linguistics can be used to identify the emotions expressed in Italian opera.

In music, the word "aria" refers to a piece of lyrics that is sung by only one singer to express an emotion. The content of the lyrics is usually highly structured and repetitive (Burden, 1998). It is very common for researchers in the field of musicology studies to study arias using the affects transmitted by its lyrics (Scherer, 1995). Thus, constructing a model that is capable of automatically identify the emotion expressed by an aria would be of great help for researchers of musicology and also the general public that is interested in opera.

This chapter is divided in 2 sections. Section 1.1 illustrates the general objective of the dissertation and clarifies that the achievement of the general objective is subjected to the fulfilment of a series of specific sub-objectives. Section 1.2 introduces briefly the structure of the dissertation and their content.

1.1 General and Specific Objectives

The general objective of this research work is to construct a model that is capable of automatically identifying the emotion transmitted by an Italian aria. This goal can be achieved by the fulfillment of the following specific objectives:

1. On the basis of available supervised training material, determine the

most efficient model and text representation in identifying the emotion transmitted by a verse of aria;

2. Increase the amount of supervised data by the means of self-learning;
3. Convert the verse-level annotation to aria-level annotation;
4. Find the most appropriate text representation for aria emotion detection;
5. Find the most appropriate classifier for aria emotion detection;

These objectives impose a number of difficulties. First, the size of available annotated data that can be used to train the machine learning model is small (this issue will be further elaborated in Chapter 4). Second, Italian arias are written in archaic Italian. The nature of the language excludes the possibility of using most pre-trained word embeddings, such as the word embeddings of Fasttext (Bojanowski et al., 2017) as text representation, because such embedding are trained on materials written in modern languages. This research involves also the change of granularity from verse-level to aria-level of the supervised data. There are very few studies that have been conducted on the emotional analysis of Italian opera and, to the best of my knowledge, none of them has handled a similar issue.

1.2 Dissertation Structure

This dissertation is composed of 6 chapters.

Chapter 2 includes fundamental knowledge that one needs to possess in order to comprehend the research methodologies of this dissertation. It first explains the learning mechanism of the models employed in the task and how they identify automatically the emotion of a text. It also presents concepts related to model training, different evaluation metrics to assess model performance such as precision and recall, and an approach to expand the size of training data named self-learning. Later, it introduces the methods adopted for the production of vectorical representation of text, namely TF-IDF (term frequency–inverse document frequency) and text embeddings.

Chapter 3 is a brief review of related works in relevant research areas such as sentiment analysis and emotion identification.

Chapter 4 and 5 are concerned with the emotion identification of Italian opera on verse-level and aria-level. Chapter 4 focuses on the emotion prediction at verse level. It also presents in detail the AriEmozione 1.0 corpus adopted for the investigation of emotion in Italian opera, their statistics and composition. Chapter 5 explores the emotion identification methodologies at aria level, presents the annotation process of the AriEmozione 2.0 corpus of aria lyrics through a self-learning approach and explicates how to use the annotated corpus to train a model that can perform text classification by machine learning methods.

At last, in chapter 6, the full dissertation is reviewed and conclusions are drawn.

Chapter 2

Background

This chapter elaborates basic concepts that are essential in order to fully understand the subject of this research. Section 2.1 introduces the learning mechanism of neural network. Section 2.2 introduces the fundamental concepts involved in model training, the evaluation metrics commonly used for model performance in multi-class classification task and explains how they are calculated, and the concept of self-learning, a learning method adopted in this study to expand the size of annotated data set. Section 2.3 introduces the two text representations employed in this work, namely TF-IDF and word embeddings.

2.1 Introduction to Supervised Models for Text Classification

This section shows the structure and the components of the artificial neural network model, how they take numerical input and calculate output, and the special features of convolutional neural network.

2.1.1 Neural Network

As defined by *CIRP Encyclopedia of Production Engineering* (Marilena, 2014), an artificial neural network (usually called NN) is a computational model that is inspired by the structure and/or functional aspects of biological neural networks. A NN is composed of a series of computation units (usually

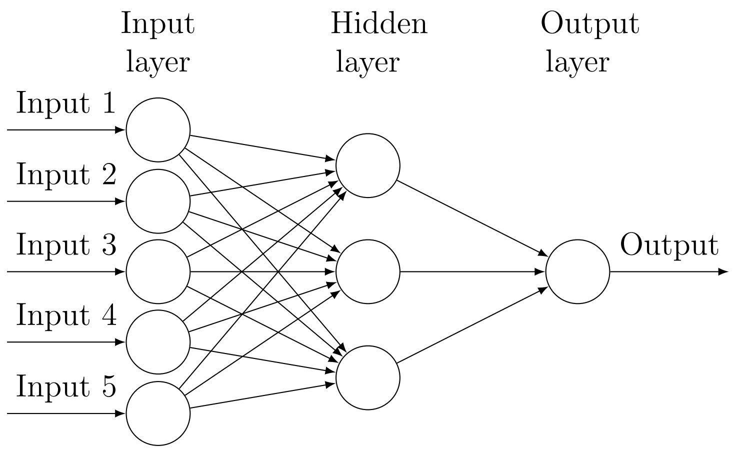


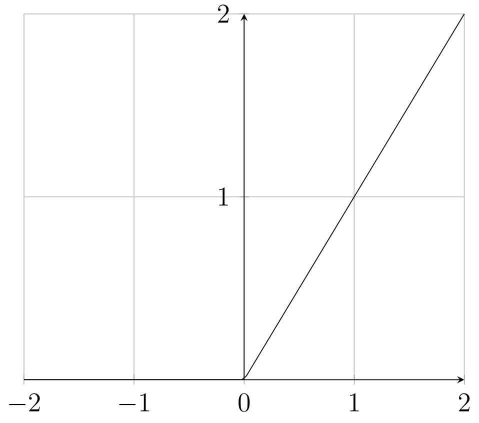
Figure 2.1: A neural network.

called neurons) organized into different layers and the connections between them. Figure 2.1 is an example of a NN. Each white ring represents a neuron and the arrows represent the connections between neurons. This NN takes 5-dimension vectors as input and produces one value as output.

At the begin of training, a random weight is assigned to each connection between two neurons. In the example of 2.1, there are five neurons in the input layer. Assume that the weight assigned to each connection is w_{ij} with $i \in \{1, 2, 3, 4, 5\}$, $j \in \{1, 2, 3\}$ (weight w_{11} belongs to the connection between the first neuron of the input layer and the first neuron of the hidden layer, weight w_{12} belongs to the connection between the first neuron of the input layer and the second neuron of the hidden layer, and so on). The input vector is $[x_1,x_2,x_3,x_4,x_5]$. The input will multiply the weights of neurons in the input layer and be summed up. The sum of the products can be represented by the following equation:

$$s_j = \sum_{i=1}^{5} w_{ij} * x_i \tag{2.1}$$

The value s_j is then feed to a function called activation function.



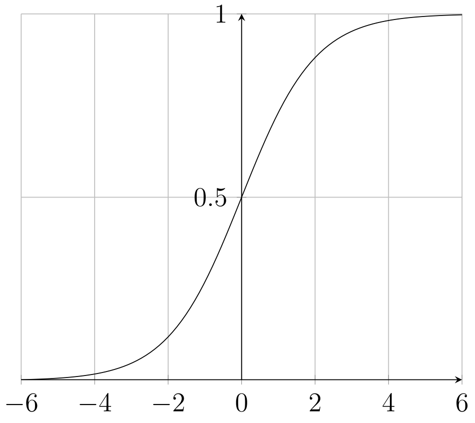


Figure 2.2: ReLU function (left) and sigmoid function (right).

2.1.1.1 Activation Function

An activation function defines the output behavior of a neuron (Lane et al., 2019). Usually neurons from the same layer share the same activation function. Here I present the three activation functions employed in this dissertation, namely rectified linear function (ReLU), Softmax function, and Sigmoid function. The ReLU activation function is defined:

$$f(x) = max(0, x) \tag{2.2}$$

where x is the input to a neuron. If x is larger than 0, then the output is x; otherwise the output is 0. Figure 2.2 shows the relation between the input (x-axis) and output (y-axis) of ReLU function.

The sigmoid function is characterized by its S-shaped form. It is defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

where x is the input to a neuron and e is a mathematical constant approximately equals to 2.72. Figure 2.2 shows the relation between the input (x-axis) and output (y-axis) of sigmoid function. As can be seen, the output of a sigmoid function is always between 0 and 1.

Unlike the ReLU function and the sigmoid function, the input to a softmax function is a vector comprised of all outputs of the neurons from the previous layer instead of the sum of these outputs. The softmax function is defined by the following formula:

$$f(a) = \frac{e^{a_i}}{\sum_k e^{a_k}} \tag{2.4}$$

where a is the input vector to a neuron. a is of length k and $i = \{i \in N \mid 0 < i \leq k\}$.

The output of the softmax function is a vector. The sum of all elements in the output vector is 1. Due to this special feature, in a classification task, the sigmoid function is usually employed as the activation function of the output layers, because the output vector elements can be interpreted as a probability distribution.

2.1.1.2 Datasets

In order to explain the learning mechanism of NN, it is necessary to clarify the concepts of supervised and unsupervised learning. In unsupervised learning, training data is not labelled, and the model will learn directly from the data (Lane et al., 2019). In supervised learning, the input data must be labelled. This section is concerned only with supervised learning.

In order to fulfil a supervised learning task, three types of datasets are needed: training set, development set (sometime also called validation set), and test set. These three datasets are used in different phases of training. Training set is used to fit the model, then the fitted model predicts on the development set to produce an unbiased evaluation of the fitted model and help tuning the model. At last, when the tuning is completed, the model predicts on the test set to evaluate the efficiency of the model (James et al., 2013). For example, in order to train a NN to distinguish if a mail is spam, the training data should be mails with its categorical label *spam* or *not spam*. The training data of mails should be split into training set, development set and test set to meet the needs of different training phases.

2.1.1.3 Loss Function

It is explained above how a NN processes the input data using weights and activation functions, passes data from one layer to another, and generates an output, yet it remains unclear why a NN is able to perform its task producing correct output. This section reveals how NN completes its task starting from a given set of random weights and a supervised data set.

In supervised learning, after the NN is given an input and produced an output, a function will quantify and measure the difference between the output and the actual label of the input. Such function is called loss function. The goal of training a neural network is to minimize the loss function based on the training data. The loss function employed in this study is called categorical cross-entropy loss function. It can be defined as:

$$f(x) = -\sum_{i=1}^{n} y_i * \log \hat{y}_i \tag{2.5}$$

where n is the length of output vector, y_i is the i_th value in the output vector and $\hat{y}_i$ is the i_th value of the expected output.

Basing on a series of calculations and the scope of minimizing the value of loss function, the weights assigned to the inter-neuron connections will be updated bit by bit, and finally the NN will converge and acquire the ability of performing the specific task. This learning mechanism involves complicate mathematical calculation and is outside the scope of this dissertation, therefore I will not go into further explanation.

2.1.2 Convolutional Neural Network

A convolutional neural network (CNN) is a class of neural network designed to process structured data[1] (Lane et al., 2019). CNN gets this name from the fact that the input data will be convolved by a filter. The major difference between CNNs and simple feed-forward NNs lays in its architecture. The input layer of CNN is called convolutional layer. This layer is composed of several filters, and each filter is defined by two elements: a weights and an activation function. Each filter is like a window that captures part of the input data. The filter will multiply the data that it currently covers with the weights, sum the result up, and pass it to the activation function. The ReLU function is commonly employed as the activation function. In that case, this process can be explained by the following formula:

$$z = max((\sum_{i=1}^{n} w_i * x_i), 0) \tag{2.6}$$

[1]Structured data is data formatted and stored in a pre-defined method.

where z is the output of the filter, w is the weight of the filter and x is the input data.

The advantage of CNN over ordinary NN is that it takes into account the correlation between words. Unlike NN that process the input data regardless of the word order, CNN's filters "scan" more than one word at each slide and thus retains also the information about word context.

2.1.3 Other Models

Other than neural networks, models such as the k-nearest neighbors algorithm, support-vector machines, logistic regression model, and Fasttext classifier are also employed in this study.

2.1.3.1 K-nearest Neighbors Algorithm

K-nearest neighbors algorithm (kNN) is a non-parametric learning method for classification. In order to determine the class of an instance l, the k nearest neighbor instances are retrieved [2]. The instance l is then classified on the basis of the plurality vote of its k nearest neighbors (Guo et al., 2003).

2.1.3.2 Support-Vector Machines

Support-vector machines are supervised models capable of performing classification task by constructs a hyperplane or set of hyperplanes in a high-dimensional space (Noble, 2006).

2.1.3.3 Logistic Regression

Logistic model, also called logit model, is a learning mechanism that applies the techniques of linear regression to classification problems (Friedman et al., 2001). It is widely used in machine learning of tasks such as text classification.

[2] k is typically a small positive integer between 1 and 10

2.1.3.4 Fasttext Classifier

Joulin et al. (2016) proposed a simple and efficient approach for text classification. In this study, multinomial logistic regression is combined with bag of words. When handling corpus of large size, the hierarchical softmax can be adopted to speed-up the computation.

2.2 NLP Experimentation and Evaluation

In the area of machine learning, "performance" refers to the degree to which a model fulfils its task, in other words, it responds to the question "How good is the model?". This section first introduces the evaluation metrics for the performance of a model, then presents a method to annotate data on the basis of these evaluation metrics.

2.2.1 Performance Evaluation

There are many metrics that can be used to interpret the performance of a classification model. The evaluation metrics employed in this dissertation are Accuracy and F1 score.

2.2.1.1 Accuracy

Accuracy is the fraction of correct predictions made by the classification model (Hossin and Sulaiman, 2015). It is defined:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{2.7}$$

For example, if a model need to identify if a mail is a junk mail or not, and 80 out of 100 mails are categorized correctly, then the accuracy of the model is 0.8.

2.2.1.2 F1 Score

The emotion identification task addressed in this dissertation is a multi-class classification task, thus this section focuses on the calculation of multi-class

		Actual		
		A	B	C
Predicted	A	7	1	2
	B	4	6	0
	C	1	2	7

Figure 2.3: Confusion matrix.

F1 score.

It is necessary to explain what Precision and Recall are in order to fully understand the meaning of F1 score. Assume Figure 2.3 is a confusion matrix of the prediction made by a model.

In Figure 2.3, each column represents the instances that belong to one class and each row represents the instances that are predicted to belong to one class. For example, the number of instances belong to A class is 7+4+1 = 12, 7 instances are predicted correctly, 4 are predicted as B and 1 is predicted as C.

The Precision of a class is the proportion of the correct prediction of one class out of all predictions of that class. For example, in the above confusion matrix, the precision of class A is $Precision_A = \frac{7}{7+1+2} = 0.7$.

The recall of a class is the proportion of the correctly predicted samples of one class out of all samples of that class. For example, the recall of class A is $Recall_A = \frac{7}{7+4+1} \approx 0.55$.

F1 score is a evaluation metric that takes into account both precision and recall. In multi-class classification tasks, the per-class F1 score can be calculated based on the precision and recall of that class:

$$f = 2 \times \frac{recall \times precision}{recall + precision} \tag{2.8}$$

The overall F1 score is calculated using per-class F1 scores. There are two different methods to calculate overall F1 score, namely macro-averaged F1 score and weighted-average F1 score. The method adopted in this dissertation is macro-averaged F1 score, which is actually the arithmetic mean of the per-class F1-scores. The calculation is defined as:

$$F_1\ score = \frac{\sum_{i=1}^{n} f_i}{n} \tag{2.9}$$

where n is the number of classes, f_i is per-class F1 score, and $F_1\ score$ is macro-averaged F1 score.

2.2.2 Self-Learning

Self-learning, sometime called self-training or incremental semi-supervised training, refers to the method of annotating raw data using a manually annotated train set with the aim of expanding the size of training data (Jurkiewicz et al., 2020). In most cases, only part of the automatically annotated data will be accepted.

At the beginning of self-learning process, the manually labelled data set, also called gold set, is split to train set and test set. The model is trained on the train set then tested on the test set, and the performance of model is recorded. Next, the model predicts on the raw data. The predictions that the model is most sure about are selected. In order to testify the reliability of the selected automatically annotated instances, the model is trained on the train set and selected instances, then tested on the test set. The test result is compared with the performance of model trained on solely train set. If the test result is better comparing to the performance before, the selected automatically annotated instances are accepted, otherwise these instances are discarded. Figure 2.4 shows the process of self-learning.

2.3 Text Representations

Text in human language is incomprehensible to computer, and that is why we need machine text representations. Text representation is a numerical representation of text that serves to make the text mathematically computable (Yan, 2009). The choice of text representation is crucial to the development of a machine learning model.

In this section, I introduce the text representations adopted in this work: TF-IDF and word embeddings.

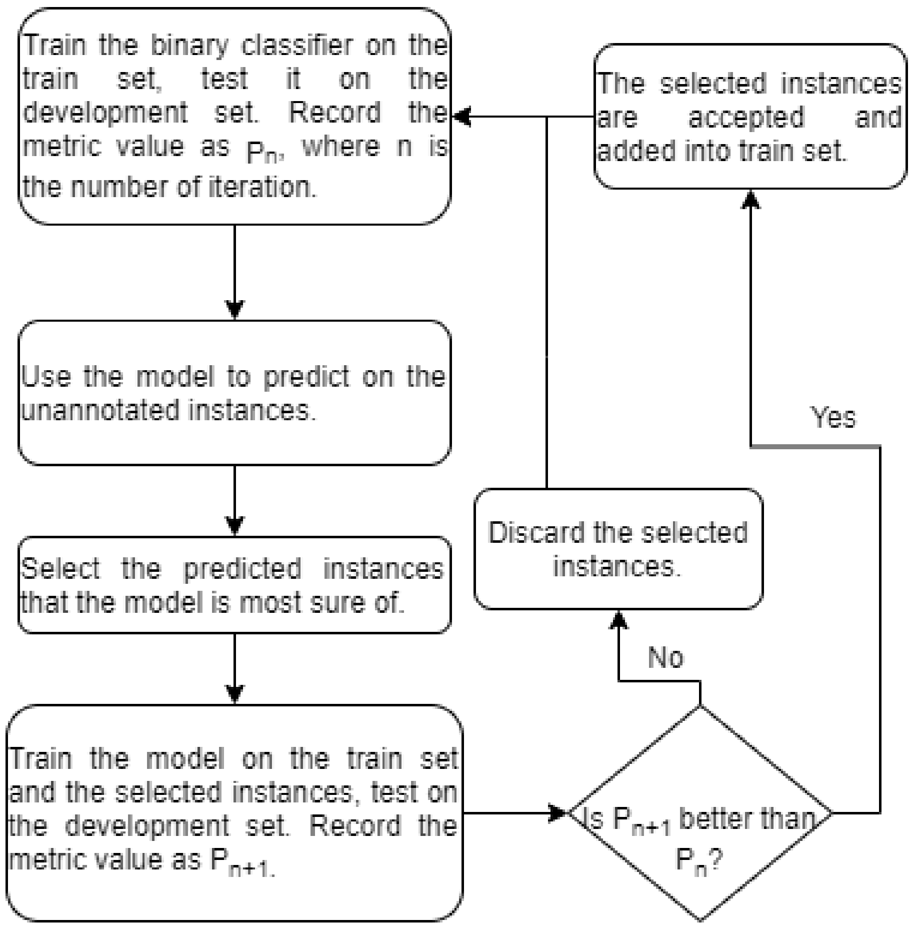

Train the binary classifier on the train set, test it on the development set. Record the metric value as P_n, where n is the number of iteration.
The selected instances are accepted and added into train set.
Use the model to predict on the unannotated instances.
Yes
Discard the selected instances.
Select the predicted instances that the model is most sure of.
No
Train the model on the train set and the selected instances, test on the development set. Record the metric value as P_{n+1}.
Is P_{n+1} better than P_n?

Figure 2.4: Self-learning process.

2.3.1 TF-IDF

Term Frequency–Inverse Document Frequency (also called TF-IDF), is a numerical measure composed of two statistics calculated separately. It reflects the importance of a text unit (e.g., token, word bigram, character tri-gram) to a corpus and can be used to address tasks like text similarity calculation, text classification and information retrieval.

Term frequency is a simple and intuitive concept[3]. It refers to the fre-

[3]It is worth noticing that the "term" of "term frequency" has a special meaning. A "term" is a text unit, it can be a word, a word 2-gram, a character 3-gram, etc.

quency of a text unit (e.g., word, word n-gram, character n-gram) in a document normalized by the number of words contained in the document. Suppose we have a document d of length n and the word i appears m times in this document, then the TF of word i in d is

$$TF_{(i,d)} = \frac{n}{m} \tag{2.10}$$

For example, in the text "Cat ate fish. Cat went outside.", the term frequency of the word "cat" is $2 \div 6 \approx 0.33$.

Term frequency shows the relation between the term and the document, however, it does not contain any information about the relation of the word in the document relative to other documents in the same corpus. Suppose we have a corpus composed of all the books ever written about Italy and we wish to investigate the differences between these books. It is likely that the word "and" appears in all these books and provides thus not so much useful information, while relatively rare words such as "Savoia" that occur only in some books may reveal the topic of the books and should thus be given more weight.

In 1972, Jones (1972) presented a statistical representation of term in a collection of documents , which is later known as Inverse Document Frequency. The calculation of this data is very simple. Suppose we have a collection C of p documents, the term i occurs in q of these documents, then the IDF of the term i can be represented as

$$IDF_{(i,C)} = log\frac{p}{q} \tag{2.11}$$

As can be seen, the more files that contain the word, the lower the IDF for the word, and vice versa.

At last, the TF-IDF of term i in document d is

$$TF - IDF_{(i,d,C)} = TF_{(i,d)} * IDF_{(i,C)} = \frac{n}{m} * log\frac{p}{q} \tag{2.12}$$

As can be seen from Equation 2.12, the more frequent term i appears in document d, the higher the TF; the more documents that contain term i, the lower is the IDF.

Sentence	an	andrew	computer	he	is	it	technician	with	works
1	0.035	0.035	0.000	0.000	0.035	0.035	0.035	0.000	0.000
2	0.035	0.000	0.000	0.095	0.035	0.035	0.035	0.000	0.000
3	0.000	0.044	0.119	0.000	0.000	0.000	0.000	0.119	0.119

Table 2.1: TF-IDF vectors.

2.3.1.1 TF-IDF in Use

Normally, classification models require the input data to have a fixed size. The same applies to TF-IDF.

In order to render the TF-IDF representation of all documents to have the same size, we need to know the size of vocabulary of the collection. Given a collection C that has the vocabulary size of k, each document contained in the collection can be represented as a k-dimensional vector. Each dimension corresponds to the TF-IDF of one word. These k-dimensional vectors are likely to be sparse, since it is probable that each document covers only a very small part of the vocabulary.

To take a simple example, assume that there is a very small corpus consists of only three sentences:

Sentence 1 "Andrew is an IT technician."

Sentence 2 "He is an IT technician."

Sentence 3 "Andrew works with computer."

In this case, the corpus vocabulary contains 9 words, respectively *an*, *andrew*, *computer*, *he*, *is*, *it*, *technician*, *with*, and *works*. Each sentence can be represented as a 9-dimensional vector. Table 2.1 shows the TF-IDF vectors for the example corpus. In practice, the vocabulary of a corpus is usually of significantly larger size and so is the TF-IDF vector's length, but the calculation method of TF-IDF vector is the same.

2.3.2 Word Embeddings

In 2013, Mikolov et al. (2013) proposed an unsupervised method to encode the meaning of text. This intuition of this method can be summarized by a well-known quote uttered by J. R. Firth:

You shall know a word by the company it keeps (Firth, 1957).

The text representation created in this method is called word embeddings. Word embeddings are vectors that represent the semantic meaning of word in the context of the corpus employed for training (Lane et al., 2019).

Word embeddings are trained using a neural network composed of one input layer, one hidden layer, and one output layer. The number of neurons in the hidden layer is the same as the number of dimensions of embeddings while the number of neurons in the input layer and the output layer is the same as the vocabulary size of the training corpus. The activation function used in the output layer is Softmax.

There are two approaches to train word embeddings: skip-gram and continuous bag-of-word. The skip-gram approach predicts the context word from the word of interest. The continuous bag-of-word approach predicts the word of interest based on the context word. This section focuses on the skip-gram method because it is the method adopted in this dissertation.

As mentioned above, the learning mechanism used to generate word embeddings is unsupervised, which means that the input data does not need to be labelled. Suppose the we need to train word vectors using the following sentence[4]:

Example Sentence "Andrew is a sweet guy."

When training word vectors, one of the hyperparameters is the window size of the context. This value represents the number of context word of the word of interest will be taken into consideration and used as output. Suppose a window size of the context is 1, then for each input word (except the first word and the last word of the sentence) there are two context words, one on the left and one on the right. Table 2.2 presents the word of interest (focus word) and its context word.

The example contains 5 words, therefore each word can be converted to a one-hot vector[5] of 5 dimensions with each dimension representing one word like shown in Table 2.3.

Based on the focus word-context word pairs, we can create a train set that contains 8 samples to generate the word embeddings. For example, when the input is $[1, 0, 0, 0, 0]$ (the one-hot vector of *andrew*), the expected output is

[4]This is an simplified example. In practice the corpus employed to train word embeddings is usually of very large size.

[5]A one-hot vector is a vector that has one dimension equals to 1 and the rest dimensions equal to 0.

Focus Word	Left Context Word	Right Context Word
andrew	/	is
is	andrew	a
a	is	sweet
sweet	a	guy
guy	sweet	/

Table 2.2: Example of focus word and context word for skip-gram word embeddings computation.

word	One hot vector
andrew	[1, 0, 0, 0, 0]
is	[0, 1, 0, 0, 0]
a	[0, 0, 1, 0, 0]
sweet	[0, 0, 0, 1, 0]
guy	[0, 0, 0, 0, 1]

Table 2.3: Words and their correspondent one-hot vectors.

$[0, 1, 0, 0, 0]$ (the one-hot vector of *is*). As mentioned above, the number of neurons in the input layer and the output layer is the same as the vocabulary size of the training corpus, which in this example is 5. Suppose there are 2 neurons in the hidden layer [6], When the training is finished, each neuron in the hidden layer holds 5 weights. Each weight can be represented by w_{ij} with $i \in \{1, 2, 3, 4, 5\}$, $j \in \{1, 2\}$. After training, the output of NN will be ignored, and the dot product of one-hot vector and the weight matrix of hidden layer will be save as word embeddings. Table 2.5 represents this process. The calculated word embeddings can be saved and used as text representations to perform different tasks.

[6]the number 2 is adopted to simplify the example, actually the hidden layer of the model employed to train word vectors usually has 100 to 300 neurons.

$$\begin{bmatrix}1 & 0 & 0 & 0 & 0\end{bmatrix} \times \begin{bmatrix}w_{11}\\w_{21}\\w_{31}\\w_{41}\\w_{51}\end{bmatrix} \begin{bmatrix}w_{12}\\w_{22}\\w_{32}\\w_{42}\\w_{52}\end{bmatrix} = \begin{bmatrix}w_{11}\\w_{12}\end{bmatrix} \tag{2.13}$$

Figure 2.5: Calculate the word vector of *andrew*.

Chapter 3

Related Work

This chapter gives a general overview of research work on sentiment analysis or emotion identification. Section 3.1 focuses on research in the field of sentiment analysis and introduces three major approaches adopted in addressing this task. Section 3.2 centers around studies on emotion identification.

3.1 Sentiment Analysis

Sentiment analysis, also called opinion mining, aims at determine the polarity of a text by investigating text features (Liu and Özsu, 2009). The decision of sentiment analysis approaches is often binary (positive, negative) or ternary (positive, negative, and neutral).

On the basis of granularity of study, the sentiment analysis tasks can be divided into three levels: document level (e.g., (Turney, 2002), (Zhang et al., 2009)), sentence level (e.g., (Jagtap and Pawar, 2013)), and aspect level (e.g., (Zhang et al., 2012)). Document level sentiment analysis tasks focus on determining the sentiment expressed by a whole document, while sentence level sentiment analysis tasks consider each sentence as a unit that carries sentiment. Aspect level sentiment analysis tasks aim to distinguish entities from a text and to distinguish the sentiment polarity of these entities (Shivaprasad and Shetty, 2017). For example, Zhang et al. (2012) propose a aspect level sentiment analysis system named "Weakness Finder". This system is capable of analyze the customer reviews on one product and find out the unsatisfied aspects in these reviews such as price and after-sell service.

The existing approaches of sentiment analysis can be categorized into three types: knowledge-based approach, statistical approach, and hybrid approach (Cambria et al., 2017).

Knowledge-based approach relies on resources such as affect words (e.g. Senticnet (Cambria et al., 2010)). Kundi et al. (2014) presents a lexicon-based framework that aims to perform sentiment analysis on English tweets that contain opinion toward smartphone products. In this study, the detection of text polarity is based on various lexicon resources for sentiment analysis. Researchers experimented with both binary classification and ternary classification. Similarly, Al-Ayyoub et al. (2015) constructed a sentiment lexicon of Arabic words leveraging existing English lexicon and conducted sentiment analysis on tweets in Arabic on the basis of such lexicon. In this study, researchers extracted Arabic stems from collected news and the publicly available data set created by Abuaiadah et al. (2014), translated the stem from Arabic to English and use an existing English lexicon to determine the sentiment value of each stem. The lexicon-based model achieved a promising performance in categorizing 300 Arabic tweets into Positive, Neutral, and Negative.

Statistical approaches such as neural network are commonly-used in sentiment analysis tasks too. Dos Santos and Gatti (2014) proposed a deep convolutional neural network for the binary sentiment analysis on movie reviews. The corpus adopted in this research was the Stanford Sentiment Treebank (Socher et al., 2013). The CNN employed in this task possesses two convolutional layers and is capable of extracting features from the character-level up to the sentence-level.

Hybrid approach is the combination of knowledge-based approach and statistical approach. Feldman et al. (2011) proposed a sentiment analysis application called "The Stock Sonar" (TSS) that is capable of execute sentiment analysis on articles relative to the stock market. This application is created on the basis of a hybrid approach that integrates sentiment lexicon, phrase-level patterns and semantic event. This hybrid approach achieved a quite promising result.

3.2 Emotion Identification

On the basis of numerous previous researches on the binary or ternary sentiment analysis, researchers are seeking to move to a higher level and solve the more complicate task of multi-class emotion identification. Such researches are conducted on various forms of text such as social media content (e.g., Tweets (Roberts et al., 2012), Facebook posts (Pool and Nissim, 2016)), lyrics (Hu et al., 2009), news (Kirange and Deshmukh, 2012), and children's fairy tales (Alm et al., 2005) using both rule-based classifiers and the supervised or unsupervised machine learning models.

Some of these works adopted more intuitive rule-based systems. For example, Asghar et al. (2017) proposed a rule-based framework for sentence-level emotion identification of user reviews using an emotion lexicon. Researchers created a mix-mode classifier that takes into account not only emotion words, but also emoticons and slang and compared the performance of mix-mode classifier with another classifier that is created using only emotion words as resources. Both classifiers are trained and tested on a corpus of news. The mix-mode classifier outperformed the other one. Similarly, Tromp and Pechenizkiy (2014) introduced a rule-based algorithm that predicts the affect of sentences by matching pre-defined linguistic patterns of the examined text. In this study, researchers created eight emotion labels for the text leveraging Plutchik's wheel of emotions (Plutchik, 1980). The data set adopted in this study is Affect Dataset (Alm, 2008). This data set is composed of sentences extracted from different books. Each sentence is labelled by only one emotion. Using the approach proposed, this classifier outperformed the SVM classifier combined with TF-IDF. Strapparava and Mihalcea (2007) explored the connection between emotions and lexical semantics on the basis of a data set comprised of news titles extracted from different websites. The data set adopted in this study use a fine-grained annotation, which means that each entry is assigned not only a series of emotion labels but also the corresponding emotion intensity values of each label. There are six emotion labels, respectively *anger, disgust, fear, joy, sadness*, and *surprise*. Each emotion label of an entry corresponds to an emotion intensity value between 0 and 100. In order to identify the emotion transmitted by a news headline, researchers opted for a rule based system capable of obtaining dependency graph of the news title using the Stanford syntactic parser and rating each word for every emotion on the basis of various lexical resources such as SentiWordNet (Esuli and Sebastiani, 2006) and WordNetAffect (Strapparava et al., 2004).

Some researchers opted for a hybrid approach, making use of both supervised and unsupervised machine learning and rule-based methods to achieve a higher accuracy. For example, Gievska et al. (2014) undertook a 7-class emotion detection research on the International Survey on Emotion Antecedents and Reactions data set[1]. In this study, researchers considered 7 emotions labels on the basis of Ekman's six emotional categories (Ekman, 1994), namely *anger*,*fear*,*sadness*,*disgust*,*joy*,*surprise*, and *neutral*. They experimented with only lexical-based method, only machine learning method, and the blended method that uses both lexicon and machine learning. The lexical-based method is developed using a variety of different language resources such as WordNetAffect (Strapparava et al., 2004), AFINN (Nielsen, 2011), H4Lvd[2] and NRC word-emotion association lexicon (Mohammad and Turney, 2013). As for the machine learning methos, researchers tested with different algorithm and chose SVM algorithm for its significant precision advantage. The hybrid method takes into account both the predictions of lexical-based method and machine learning method to generate the output. At last, the hybrid method outperformed the other two method.

[1]http://www.affective-sciences.org/system/files/webpage/ ISEAR.zip
[2]http://www.wjh.harvard.edu/ inquirer/Home.html

Chapter 4

Emotion Identification in Italian Opera at the Verse Level

This chapter presents in details the corpus employed to investigate the automatic identification of emotion at the verse level and the experiments involved in identifying the emotion expressed by aria verses. Since an aria may transmit more than only one emotion, the granularity that this work begins with is verse-level.

4.1 The AriEmozione 1.0 Corpus

AriEmozione 1.0 is the corpus used in this section to investigate the emotion identification of Italian opera at the verse level. It is also a subset of the CORAGO-1700 corpus. Constructed by the Department of Classical Philology and Italian Studies of the University of Bologna under the CORAGO project[1], the CORAGO-1700 corpus is composed of Italian arias written between 1655 and 1765.

AriEmozione 1.0 contains about 2,500 verses of 1,005 Italian arias written in archaic Italian. The corpus contains $34,608$ tokens ($4,458$ types). The

[1]CORAGO is the Repertoire and archive of Italian opera librettos. It constitutes the first implementation of the RADAMES prototype (Repertoriazione e Archiviazione di Documenti Attinenti al Melodramma E allo Spettacolo)Pompilio et al. (2005); See http://corago.unibo.it.

Number of verses	1	2	3	4	5 to 9	10 to 14
Number of aria	65	690	132	66	44	8

Table 4.1: Number of verses contained in Aria.

average length of verses is 72.5 characters. Each aria contains 1 to 14 verses and most arias contain 2 or 3 verses (see Table 4.1).

The emotion labels of AriEmozione 1.0 are created leveraging the tree of emotion classification proposed by Parrott (2001). The principal layer of emotion tree consists of six emotions: *love*, *joy*, *surprise*, *anger*, *sadness*, and *fear*. We decided to substitute *surprise* by *admiration* after a close scrutiny to the material and to preserve the rest 5 labels. As described by Fernicola et al. (2020) the classes are defined as follows:

Amore (love) incl. affection, lust, longing.

Gioia (joy) incl. cheerfulness, zest, contentment, pride, optimism, enthrallment, relief.

Ammirazione (admiration) admiration or adoration of someone's talent, skill, or other physical or mental qualities.

Rabbia (anger) incl. irritability, exasperation, rage, disgust, envy, torment.

Tristezza (sadness) incl. suffering, disappointment, shame, neglect, sympathy.

Paura (fear) incl. horror and nervousness.

AriEmozione 1.0 corpus was annotated by two Italian native speakers independently under the guidance as shown in Figure 4.1. They were required to annotate verses with the emotion it transmitted, an optional secondary emotion, and their level of confidence (*total confidence*, *partial confidence*, or *very doubtful*) for such annotation. The Cohen's kappa coefficient [2] (Fleiss et al., 1969) in this stage was 32.3, which reflects the level of complexity of this annotation task. The same annotators then discussed all dubious instances basing on their annotation in the first stage in order to reach consensus. We

[2]Cohen's kappa is a statistic that measures inter-annotator reliability for qualitative items.

First of all, thank you for helping with this work. We are a group of researchers from the D. of Classical Philology and Italian Studies and the D. of Interpreting and Translation, both at UniBO. Your work will help us to produce artificial intelligence models to analyse the lyrics in music.
At this stage we are focused on opera. You will annotate arie in Italian from diverse periods, looking for the emotions that they express. Your work consists of identifying the emotion expressed in each of the verses composing an aria. You can choose among six emotions (or none of them), which are defined next: [. . .]

Each row is divided in six columns:
id A unique id, tied to the verse. Do not modify it.
verse A verse, inside of an aria. This is the text that you are going to analyse.
emotion Here you can select the expressed emotion (or none of them)
emotion sec. This is available to choose a secondary emotion, in case it is really difficult to choose just one
confidence Not being 100% sure is ok. If that is the case, please let us know by choosing the right confidence level (default: "I am sure").
comments Feel free to tell us something about this instance, if you feel like.

Figure 4.1: Instructions given to the annotators of the emotions in the AriEmozione 1.0 corpus.

then partitioned the AriEmozione 1.0 corpus into train set, development set and test set. Table 4.2 shows the emotion distribution of each set and the whole corpus. The class "nessuna" includes verses that do not express any emotion (e.g. verses like "Alessandro" that consist of only a name).

Each entry of AriEmozione 1.0 is composed of a unique ID, verse, emotion label, and the annotator's confidence level. Table 4.3 shows some entries of AriEmozione 1.0. Table 4.4 shows the verses that belong to the same aria.

4.2 Experiments

Experiments in this stage aim to identify the emotion transmitted by Italian aria at verse level. Different classification models are tested combining with different text representation to reach a satisfactory result.

We adopted TF-IDF vectors (based on word and on character 3-grams), pre-trained 300-dimensional embeddings of Fasttext (Grave et al., 2018), and dense representations LDA (Hoffman et al., 2010) as text representation. Spacy Italian tokenizer and casefold were adopted for text pre-processing. As for classification model, we experimented with k–Nearest Neighbors al-

	amore	gioia	ammirazione	rabbia	tristezza	paura	nessuna	total
train	289	274	289	414	503	166	38	**1,973**
dev	36	31	23	84	61	12	3	**250**
test	37	39	30	64	54	15	11	**250**
overall	362	344	342	562	618	193	52	**2,473**

Table 4.2: AriEmozione 1.0 statistics.

ID	**verse**	**class**	**confidence level**
ZAP1599826_01	Che ti vedrà placata e vuol morirti al piede vittima sventurata d'un infelice amor	Amore	Total Confidence
ZAP1596132_00	Non perdo la calma fra' ceppi o gli allori; non va sino all'alma la mia servitù	Gioia	Very Doubtful
ZAP1593309_02	Palme il Gange a lui prepari e d'Augusto il nome impari dell'incognito emisfero il remoto abitator	Ammirazione	Total Confidence
ZAP1600210_01	L'ombra del figlio esangue m'ingombra di terror	Paura	Total Confidence
ZAP1595664_01	Non deggio, non voglio sentirlo accusar	Rabbia	Total Confidence
ZAP1593295_05	arti dagli occhi miei, lasciami per pietà	Tristezza	Total Confidence

Table 4.3: Instances of the AriEmozione 1.0 corpus, showing the unique ID, verse, class it belongs to, and the annotators' confidence level.

gorithm (kNN), logistic regression, support-vector machines (SVM), fully connected neural networks with 2 or 3 hidden layers, and Fasttext classifier (Joulin et al., 2016). The experimented settings of the above-mentioned models can be found in Table 4.5.

The train set and development set of AriEmozione 1.0 are unified to perform 10-fold cross-validation. Then the representation-model combinations of promising results are adopted for training on train set and development set and test on the test set. Models' performance was evaluated on the basis of macro-average F1 score and Accuracy (cf. Section 2.2.1). TF-IDF vectors, Fasttext word embeddings and LDA are tested using *k*-NN, SVM, logistic regression, and NN, while Fasttext classifier is tested with and without pre-

ID	verse	class	confidence level
ZAP1598336_00	L'augelletto in lacci stretto perché mai cantar s'ascolta?	Ammirazione	Partial Confidence
ZAP1598336_01	Perché spera un'altra volta di tornare in libertà	Ammirazione	Partial Confidence
ZAP1598336_02	Nel conflitto sanguinoso quel guerrier perché non geme?	Ammirazione	Partial Confidence
ZAP1598336_03	Perché gode colla speme quel riposo che non ha	Tristezza	Partial Confidence

Table 4.4: All verses contained in aria ZAP1598336.

Model	Settings
k-NN	L2-Norm exploring with $k \in [1, \dots 9]$.
SVM	RBF; both explored with $c \in [1, 10, 100, 1000]$ and $\gamma \in [1e-3, 1e-4]$.
Log Reg	Multinomial Logistic Regression with Newton-CG solver.
NN	2 (3) hidden layers with neuron numbers $\in [32, 64, 96, 128, 256]$ ($\in [8, 16, 32, 64, 96]$); 20% dropout for each hidden layer; ReLu activation function for input/hidden layers; softmax activation function for output layer; categorical cross-entropy loss function; Adam optimizer; epochs $\in [1, \dots 15]$
FastText	300 dimensions embeddings with or without pre-training; learning rate $\in [0.3, 0.6, 1]$; epochs $\in [1, 3, 5, 10, \dots, 100]$

Table 4.5: Summary of explored model configurations.

trained embeddings for word and character 3-gram.

4.3 Results

The test results of NNs and Fasttext classifier are shown in Table 4.6[3].

As can be seen from Table 4.6, in the cross-validation, TF-IDF of character 3-grams consistently outperforms the other text representations across all the models and reached a F_1 score of 0.49 in cross-validation using 3-layers NN. In contrast, the performance of the Fasttext pre-trained embeddings is less promising, which is most likely related to the linguistic nature of the corpus: verses in AriEmozione 1.0 are written in 18th century Italian, while the

[3]The full experiment results can be viewed at `https://docs.google.com/spreadsheets/d/1Ztjry2mJs6ufCZM1O5CQRyZ8pA5YDnToNOhONGX1nWO/edit?ts=5f5a0fd0#gid=1437275830`

Model	Representation	$F1_{cross-val}$	$Acc_{cross-val}$	$F1_{test}$	Acc_{test}
2-layers NN	char 3-grams	0.42	43.61	0.47	46.86
2-layers NN	words	0.42	42.91	0.43	43.10
2-layers NN	LDA char	0.27	29.56	0.27	31.80
3-layers NN	char 3-grams	0.49	41.86	0.40	41.84
3-layers NN	words	0.47	42.60	0.40	41.84
3-layers NN	LDA char	0.26	31.41	0.30	31.80
FastText	char 3-grams	0.43	45.00	0.41	42.37
FastText	pre-trained chars	0.43	47.00	0.41	41.00
FastText	words	0.42	42.56	0.39	44.07
FastText	pre-trained words	0.38	41.00	0.40	42.00

Table 4.6: Results of neural networks and FastText classifiers on different representations for the ten-fold cross-validation and on the test set of the AriEmozione 1.0 corpus.

training material of the pre-trained embeddings is the text from Wikipedia[4] written in modern Italian. The performance of LDA is always the poorest through all the models.

As for the results on the test set, the best performance is achieved by the combination of character 3-grams TF-IDF and 2-layers NN, obtaining a F1 score of 0.47 and an Accuracy of 46.86.

Table 4.7 shows the confusion matrix of the best test result. As can be seen, the model showed better performance when predicting the class "Rabbia": about 64% instances belong to "Rabbia" are classified correctly. However, model's performance in predicting verses of “ammirazione” or “gioia” was relatively poor: only 37% of “ammirazione” verses and 31% of "gioia" verses were classified correctly. This can be attributed to the fact that the model tends to confuse example of “ammirazione” and “gioia”. Besides, class “rabbia” and class “tristezza” show the same problem: about 17% of “rabbia is classified as “tristezza” while 19%of “tristezza” is classified as “rabbia”. This is most likely due to the fact that both “ammirazione” and “gioia” are positive emotions and both “rabbia” and “tristezza” are negative emotions. The same pattern is observed in the confusion matrices of other NN models as well.

There are many factors that contribute to the complexity and difficulty of

[4] `https://it.wikipedia.org/wiki/Pagina_principale`

	ammirazione	amore	gioia	paura	rabbia	tristezza
ammirazione	**0.37**	0.03	0.18	0.07	0.11	0.06
amore	0.03	**0.43**	0.13	0.00	0.09	0.17
gioia	0.27	0.16	**0.31**	0.20	0.09	0.07
paura	0.10	0.03	0.00	**0.40**	0.02	0.07
rabbia	0.20	0.14	0.03	0.13	**0.64**	0.17
tristezza	0.17	0.14	0.13	0.07	0.19	**0.48**

Table 4.7: Confusion matrix for the 2-layers neural network with TF-IDF character 3-grams.

this task. First, the average length of verses is very short, therefore is unable to carry much information. All verses are written in 18th Italian and, due to the nature of the language, the performance of high-dimensional pre-trained vectors created on the basis of texts in modern Italian is unsatisfactory. The size of training data is also very limited. Considering the difficulties of this study, it is already good to achieve a F1 score of 0.47. Nevertheless, it is also undeniable that a classifier with F1 score of 0.47 and Accuracy of 46.86 has no practical value in real life, since on average one of two predictions is wrong.

In order to address these problems, the size of annotated training data needs to be expanded. If the size of training data is significantly larger, the training data itself can also be used to generate pre-trained unsupervised word vectors, thus solve the problem of the unsuitable Fasttext pre-trained word vectors. I would also opt for aria-level annotation rather than verse-level to enhance the length of each input data. These problems and the solutions are presented in chapter 5.

Chapter 5

Emotion Identification in Italian Opera at the Aria Level

In Chapter 4, I explored the emotion identification of Italian operas at the verse level using different text representations and models. The best performance is achieved by the combination between TF-IDF of character 3-grams and 2-layer NN. On the basis of the outcome in Chapter 4, this chapter describes the methods adopted to annotate the AriEmozione 2.0 corpus and to explore the automatic identification of Italian opera at aria-level.

This chapter is divided into three sections. Section 5.1 centers around the annotation of AriEmozione 2.0 by means of self-learning. After the annotation stage, experiments were conducted on both AriEmozione 1.0 and AriEmozione 2.0 and the test results are compared and analyzed. Section 5.2 focuses on the conversion of annotation from verse-level to aria-level, laying the groundwork for the exploration of emotion transmitted by aria. At last, Section 5.3 presents experiment results of aria-level emotion identification and the discussion on the experiment results.

5.1 The AriEmozione 2.0 Corpus

In this section, the AriEmozione 2.0 corpus is introduced in details and the annotation process of the AriEmozione 2.0 corpus is presented. As mentioned in Section 4.1, CORAGO-1700 is a corpus composed of Italian arias and the AriEmozione 1.0 corpus is a sub-corpus of CORAGO-1700. Similarly, the AriEmozione 2.0 corpus is another subset of CORAGO-1700.

5.1.1 Annotation of AriEmozione 2.0

Just as AriEmozione 1.0, the AriEmozione 2.0 corpus is a subset of CORAGO-1700. AriEmozione 2.0 is composed of about 70,000 verses. It is significantly larger than AriEmozione 1.0. Similar to AriEmozione 1.0, each verse in the AriEmozione 2.0 corpus has a unique ID, but does not possess a label that is added manually by human annotator. Instead, the annotation of instances in AriEmozione 2.0 is done automatically by classification models. The details of annotation method are presented in this section.

This task of corpus annotation has many difficulties. First of all, due to the limited size of training material and the complexity of the task, the accuracy of the available models is very low, in the best case scenario the accuracy is only around 0.47. Besides, the precision of model's prediction for each class varies.

The model that has achieved the best performance on predicting emotion of Italian opera verses is a 2-layers NN and the representation employed in the experiment is the TF-IDF of character 3-grams (see Section 4.3). Hence, I use a NN of the same configurations in the experiment and character 3-gram TF-IDF as text representation to annotate the AriEmozione 2.0 corpus. The approach of self-learning is adopted to annotate automatically AriEmozione 2.0.

From the above analysis, it is clear that it is not feasible to label AriEmozione 2.0 directly using the existing model and TF-IDF of character 3-grams. I decided to adopt a common approach named One-versus-all (OVA) to address this problem. The OVA approach decomposes the multi-class classification problem of N classes into N binary classification problems in order to reduce the complexity of the classification task. Each binary classifier is responsible to identify the instances of one specific class from the rest $N - 1$ classes. When predicting an instance, the model that produces the highest output will be considered as the "winner" and the instance will be labelled by the class that this model corresponds to (Aly, 2005).

The annotation process of AriEmozione 2.0 can be divided into four steps:

1. The multi-class classification task is transformed into six binary classification tasks. Each binary classification aims to single out instances of one specific emotion from all instances. Details of this step is introduced in Section 5.1.1.1.

2. Several instances are selected from AriEmozione 2.0 and added into

	tristezza	altro
tristezza	0.32	0.68
altro	0.13	0.87

Table 5.1: Normalized confusion matrix of binary classification "Tristezza verses altro".

train set to improve the efficiency of binary classifiers. This process is named "Instances Pre-selection" and is introduced in Section 5.1.1.1.

3. Six binary classifiers collaborate to annotate AriEmozione 2.0. Section 5.1.1.2 focuses on this step

4. Several experiments are conducted on the AriEmozione corpus 2.0 to asses the quality of annotation. Experiments results are presented in Section 5.1.1.3

5.1.1.1 Instances Pre-selection

As already mentioned, I opted for One-versus-all (OVA) approach to convert multi-class classification problem to several binary classification problems. The train set of AriEmozione 1.0 is merged with development set of AriEmozione 1.0, on the basis of which six groups of training data are created. Each binary training data corresponds to one emotion. For example, in the binary training data for the classification of "Tristezza", the label of instances belong to "Tristezza" remains unchanged, while the label of instances belong to other categories becomes "Altro" ("other" in Italian). The classifier used in this phase is a NN with the same configurations of the best performance NN in chapter 4, the only difference is that the NN used for binary classification has 2 neurons instead of 6 in the output layer.

The OVA approach raised yet another problem. The supervised material, after the conversion from multi-class to binary-class, is extremely imbalanced. For example, in the binary train set for class "Paura", the ratio between instances of "Paura" and "Altro" is nearly 1:10. These highly imbalanced data sets lead to the even poorer performance of neural network than before. As shown in Table 4.7, in the best case scenario of multi-class classification task, about 48% of "Tristezza" instances were correctly predicted, however, this percentage plummeted to 32% in binary classification of "Tristezza".

Considering that these binary classifiers serve to detect instances belong to the emotion class, not the “altro” class, what matters most in the model’s performance is the recall of the emotion class. In order to enhance this metric value, several instances are selected from the AriEmozione 2.0 corpus and labelled by means of self-learning. This process starts from a binary classifier and its corresponding binary training data. First, the training data is split into train set and development set on a 8:2 ratio. The model is trained by the binary train set, tested on the binary development set and the recall of the emotion class is recorded, then the model predicts on the AriEmozione 2.0 corpus. The instances that the model shows most confidence in are selected and labelled. Another model with the same configurations of the previous model is trained on both selected instances and the binary train set, then tested on the development set. If the recall of emotion class of the new model is higher than the recall of original model, these selected instances are added into train set, otherwise they are discarded. One iteration is completed here and it starts the next iteration from the beginning. As one can imagine, the recall of the emotion class becomes higher and higher as the iterations. The text representation adopted in this process is always the TF-IDF of character 3-grams. Figure 2.4 shows the instances pre-selection process.

At the end of instances pre-selection iterations, all selected instances and their labels are saved. With the aim of testifying their reliability, I experimented with these instances selected from AriEmozione 2.0. I trained the binary models with three differently groups of data, respectively the *train set of AriEmozione 1.0*, *train set of AriEmozione 1.0 + new instances labelled as the emotion class*, and *train set of AriEmozione 1.0 + all new instances.* These models are then tested on the development set of AriEmozione 1.0. The set of *train set of AriEmozione 1.0 + new instances labelled as the emotion class* outperformed other sets thorough all six emotion. The text representation adopted for the tests is still the TF-IDF of character 3-grams.

As can be seen from Table 5.2, the combination *train & new instances* tends to achieve higher accuracy and F1 score comparing to the combination *train & new instances of emotion.* However, the most important metric in the pre-selection phase is the recall of the emotion class. It is evident that *train & new instances of emotion* outperforms other train set options in terms of the recall of the emotion class. Therefore, *train & new instances of emotion* is used as the train set to prepare the models that serve to label automatically the AriEmozione 2.0 corpus.

Emotion class	Train set	Accuracy	F1	Recall
ammirazione	train	0.851	0.787	0.019
	train & new instances	0.881	0.872	0.455
	train & new instances of emotion	0.882	0.878	0.530
amore	train	0.861	0.8	0.024
	train & new instances	0.886	0.889	0.671
	train & new instances of emotion	0.867	0.867	0.696
gioia	train	0.853	0.808	0.111
	train & new instances	0.866	0.856	0.407
	train & new instances of emotion	0.847	0.848	0.504
paura	train	0.921	0.899	0.111
	train & new instances	0.968	0.97	0.917
	train & new instances of emotion	0.952	0.956	0.924
rabbia	train	0.789	0.749	0.241
	train & new instances	0.812	0.81	0.586
	train & new instances of emotion	0.802	0.802	0.589
tristezza	train	0.746	0.724	0.296
	train & new instances	0.753	0.754	0.47
	train & new instances of emotion	0.747	0.754	0.529

Table 5.2: The evaluation of pre-selected instances from AriEmozione 2.0. The values of the column "recall" are the recalls of the emotion class.

5.1.1.2 Six NNs in Synergy

As explained in Section 2.1.1.1, the output of softmax function, which is commonly used as the activation function in the output layer, can be interpreted as a probability distribution. It is this feature of the softmax function that is exploited in this study to annotate the AriEmozione 2.0 corpus.

On the basis of pre-selected instances in the previous section, six binary neural network collaborate to annotate the AriEmozione 2.0. All these neural networks have softmax as the activation function in the output layer. Each neural network is responsible for the identification of one emotion. The training data adopted in this process is *train set of AriEmozione 1.0 + development set of AriEmozione 1.0 + pre-selected instances of the emotion class*. There are six sets of training data, each corresponds to one emotion.

First, neural networks are trained on *train set + development set + pre-selected instances of the emotion class*. Then the neural networks are used to predict the instances of the AriEmozione 2.0 corpus. The output of neu-

Emotion label	Frequency	Emotion label	Frequency
amore	10697	rabbia	13874
gioia	10716	tristezza	20084
ammirazione	11201	paura	4600
		total	71172

Table 5.3: AriEmozione 2.0 statistics.

ral network for each instance is a 2-dimensional vector with each dimension represents the probability that the instances belong to that class. For example, if the prediction of one instance is $(0.8, 0.2)$, it suggests that the neural network believes the probability that the instance belongs to the "altro" class is 0.8 and the probability that the instance belongs to the emotion class is 0.2. After every neural network has predicted on the AriEmozione 2.0 corpus, each instance in AriEmozione 2.0 has six predictions, where each prediction is a 2-dimensional vector that has one dimension represents "altro" and the other dimension represents the emotion class. In order to determine the emotion label of an instance, its six correspondent prediction are compared. For instance, if the predictions of neural networks of an instance v are the followings: $(0.88, 0.12)_{ammirazione}$, $(0.45, 0.55)_{amore}$, $(0.36, 0.64)_{gioia}$, $(0.11, 0.89)_{paura}$, $(0.49, 0.51)_{rabbia}$, $(0.23, 0.77)_{tristezza}$. As explained above, the second dimension of the output vector represents the possibility that the instance belongs to the correspondent emotion. In this given example, the prediction of the binary neural network that distinguish "paura" from "altro" shows the hightest confidence level in the emotion class, thus this instance is labelled as "paura". In this way, all instances in the AriEmozione 2.0 corpus are annotated. Table 5.4 presents some verses contained in the AriEmozione 2.0 corpus. Table 5.3 shows the class distribution of AriEmozione 2.0 corpus.

5.1.1.3 Experiment Results

In order to asses the reliability of the annotation of the AriEmozione 2.0 corpus, a series of experiments are taken and the experiments results are compared. Three different groups of training data are adopted for experimentation. They are respectively *AriEmozione 2.0*, *train set of AriEmozione 1.0 + development set of AriEmozione 1.0*, and *AriEmozione 2.0 + train set of AriEmozione 1.0 + developments set of AriEmozione 1.0*. The model used is a 2-layers neural network with the same configurations of the best-

ID	**verse**	**class**
ZAP1597771_00	Se resto sul lido, se sciolgo le vele infido, crudele mi sento chiamar	Tristezza
ZAP1597852_01	Ancor che misero sia questo core, pur soffre placido l'altrui rigore, l'amato carcere lasciar non sa	Tristezza
ZAP1598033_00	Punirò quel cor fallace e saprai per tuo tormento che si brama il tradimento, ma dispiace il traditor	Rabbia

Table 5.4: Instances from the AriEmozione 2.0 corpus. Each instance has a unique ID, the verse and its correspondent emotion.

Train set	Accuracy	F1 score
train set & development set	0.413	0.394
AriEmozione 2.0	0.417	0.411
AriEmozione 2.0 & train set & developments set	0.419	0.413

Table 5.5: Experiment results: accuracy and F1 score.

performance model in Table 4.6 and the text representation is TF-IDF based on character 3-grams. In order to enhance the reliability of the test results, each test is repeated 3 times and the arithmetic mean of metrics is recorded. The model is trained on different training data, and tested on the test set of the AriEmozione 1.0 corpus. Table 5.5 shows the accuracy and F1 score of experiments. Table 5.6 shows the diagonal values of the confusion matrix of experiments.

As can be seen, the presence of the AriEmozione 2.0 corpus does not enhance significantly the overall performance of the model. Nevertheless, it renders the per-class performance of the model more stable. As can be seen from Table 5.6, when the model is trained on train set & development set, the recalls of both class "amore" and "paura" are close to zero, while when the model is trained on *AriEmozione 2.0* & *train set* & *developments set*, these two value rise respectively to 0.243 and 0.4.

5.2 From Verse-level to Aria-level

As mentioned in Section 4.3, model's unsatisfactory performance in predicting emotion transmitted by Italian verses can be attributed to several reasons.

Train set	ammirazione	amore	gioia	paura	rabbia	tristezza
train set & development set	0.333	0.006	0.3	0.067	0.532	0.6
AriEmozione 2.0	0.556	0.234	0.276	0.4	0.441	0.55
AriEmozione 2.0 & train set & developments set	0.556	0.243	0.279	0.4	0.439	0.549

Table 5.6: Experiments results: diagonal values of confusion matrix.

First, the size of training data is limited. The presence of the AriEmozione 2.0 corpus addresses this problem, increasing significantly the size of the training data. Nevertheless, two issues remain: the short length of each verse and the incompatibility between the Italian Operas written in 18th century Italian and pre-trained embeddings created on the basis of training materials in modern Italian. This section focuses on the two latter problems. First, I increase the length of each input data by converting the annotation of the corpora from verse-level to aria-level. Then, on the basis of available arias, I used Fasttext tool (Bojanowski et al., 2017) to build an embedding based on character 3-grams.

As described in the previous section, several instances are selected from the AriEmozione 2.0 corpus via the means of self-learning to improve the performance of binary classifiers. Then these binary classifiers collaborate together to annotate automatically the AriEmozione 2.0 corpus. The annotation quality of the AriEmozione 2.0 corpus is tested. In order to increase the length of each input data and explore the emotion of Italian operas at aria level, the verse-level annotation of the AriEmozione 1.0 corpus and the AriEmozione 2.0 corpus needs to be converted to aria-level.

Considering that the length of an aria is relatively short and it is unlikely that one aria can transmit many emotions, an aria can be assigned at maximum two emotions. The emotion of an aria is determined by the most frequent emotion label among its verses. Suppose there is one aria composed of 4 verses, where 3 of them are labelled as “gioia” and 1 as “ammirazione”, the aria should be labelled as “gioia”. In case there are two emotions that are equally frequent in an aria, then the aria should be assigned two emotion labels. For example, an aria composed of 2 “gioia” verses and 2 “ammirazione” verses should be labelled as “gioia & ammirazione”. If there are more than 2 emotions that are equally frequent in an aria, the labels are considered as invalid and the aria will be discarded. In this process, 6 arias from AriEmozione 1.0 and 1623 arias from AriEmozione 2.0 are discarded

AriEmozione version	1.0	2.0
Emotion label	*Gold*	*Silver*
ammirazione	184	9,927
amore	180	9,357
gioia	174	9,487
paura	103	3,952
rabbia	274	11,637
tristezza	302	16,665

Table 5.7: Statistics of single emotion labels at aria level. The column "Gold" corresponds to the instances contained in AriEmozione 1.0 (manually annotated) and the column "Silver" corresponds to the instances contained in AriEmozione 2.0 (automatically annotated).

for having more than 2 labels. To avoid confusion, in the following, the arias contained in the AriEmozione 1.0 corpus are referred to as "gold instances" and the arias contained in the AriEmozione 2.0 corpus are referred to as "silver instances". Table 5.7 shows the statistics of single emotion labels at aria level. Table 5.8 shows the statistics of emotion labels of aria.

As mentioned in the beginning of this section, the other problem that leads to the poor performance of the classification models is that the Italian arias in the experiments are written in 18th Italian, while the pre-trained embeddings adopted as text presentation are trained on texts written in mordern Italian. In order to address this issue, I build 300-dimensional word embeddings using both gold aria instances and silver aria instances as unsupervised training material. The epoch of the training model is 5 and the learning rate is 0.05, embeddings are generated based on the character 3-grams. These embeddings are used as text representation in Section 5.3 to investigate emotion identification at the aria level.

5.3 Results

This section focuses on emotion identification at the aria level. The text representation adopted is the pre-trained embeddings created on the basis of arias in Section 5.2. As for the classification models, I opt for convolutional neural networks (see Section 2.1.2) with one convolutional layer, two hidden layers and one output layer. The activation function adopted for convolu-

AriEmozione version	1.0	2.0
Emotion label	*Gold*	*Silver*
ammirazione	109	2172
amore	122	2084
gioia	107	2099
paura	57	757
rabbia	195	3224
tristezza	185	5399
ammirazione&amore	9	1317
ammirazione&gioia	31	1769
ammirazione&paura	9	546
ammirazione&rabbia	12	1878
ammirazione&tristezza	14	2245
amore&gioia	13	1407
amore&paura	5	404
amore&rabbia	7	1456
amore&tristezza	24	2689
gioia&paura	5	642
gioia&rabbia	8	1381
gioia&tristezza	10	2189
paura&rabbia	5	579
paura&tristezza	22	1024
rabbia&tristezza	47	3119
Overall	**996**	**38380**

Table 5.8: Statistics of the annotation at the aria level. The column "Gold" corresponds to the instances contained in AriEmozione 1.0 (manually annotated) and the column "Silver" corresponds to the instances contained in AriEmozione 2.0 (automatically annotated).

learning rate	epoch	accuracy
0.0001	10	0.616
0.0001	15	0.652
0.0001	20	0.638
0.001	10	0.678
0.001	15	0.658
0.001	20	0.654

Table 5.9: Experiment results of the emotion identification at the aria level. The text representation adopted is 300-dimensional pre-trained embeddings based on character 3-grams. The models are CNNs.

tional layer is ReLu function and the stride is set to 3. Both hidden layers have 2500 neurons, dropout of 0.1 and sigmoid function as the activation function. The output layer uses sigmoid function as the activation function and generates a 6-dimensional output vector. The loss function of the CNNs is binary crossentropy loss function and the optimizer is Adam optimization algorithm.

The models are trained on the silver instances and then tested on the gold instances. Different configurations are explored. Table 5.9 shows the experiment results. As can be seen, the best performance is achieved when learning rate is set to 0.001 and the epoch is set to 10. Table 5.10 to Table 5.15 shows the confusion matrices of such model. There are six confusion matrices because the emotion identification at the aria level is a multi-label task. Each confusion matrix shows model's performance on one label. The class "other" includes the rest five labels.

As can be seen from the confusion matrices, the performances of models on all classes are promising. The class "ammirazione" achieved the highest recall of 84.8% and the class "amore" has the lowest recall of 69.4%. These results show that the CNN model is competent in distinguish each emotion from the aria. However, the overall accuracy of the model is 0.678. This phenomena can be attributed to the unbalanced distribution of instances with double labels in gold instances and silver instances. Table 5.8 shows that about 55% of silver instances has two labels, while only 17% of gold instances has two labels. It is likely that many instances in AriEmozione 1.0 with only one label are predicted as two-label instances by the CNN model. This is most likely the main factor that influenced the model's performance.

Overall, the performance of the model is good considering the difficulties of the task, however, there still remains much room for improvement. The model achieved very promising accuracy in the identification of each singular emotion label, but also showed weakness with multi-label classification task. I believe the performance of the model can be improved by changing the single-label instance - multi-label instance ratio of silver instances.

	other	ammirazione
other	0.825	0.175
ammirazione	0.152	0.848

Table 5.10: Normalized confusion matrix of the class "ammirazione" in the emotion identification at aria level.

	other	amore
other	0.947	0.053
amore	0.306	0.694

Table 5.11: Normalized confusion matrix of the class "amore" in the emotion identification at aria level.

	other	gioia
other	0.912	0.088
gioia	0.224	0.776

Table 5.12: Normalized confusion matrix of the class "gioia" in the emotion identification at aria level.

	other	paura
other	0.966	0.0336
paura	0.291	0.709

Table 5.13: Normalized confusion matrix of the class "paura" in the emotion identification at aria level.

	other	rabbia
other	0.928	0.072
rabbia	0.208	0.792

Table 5.14: Normalized confusion matrix of the class "rabbia" in the emotion identification at aria level.

	other	tristezza
other	0.807	0.193
tristezza	0.209	0.791

Table 5.15: Normalized confusion matrix of the class "tristezza" in the emotion identification at aria level.

Chapter 6

Conclusions and Future Work

Chapter 6 is divided into two sections. Section 6.1 is a sum-up of the outcomes of this dissertation. Section 6.2 introduces the papers and corpora produced during this study. Section 6.3 outlines the possible work that can be take out in the future.

6.1 Research Outcomes

This thesis explored the emotion identification of Italian operas on both verse-level and aria-level. The research outcomes allow scholars in the field of musicological studies and the lay-public to investigate systemically the emotion transmitted by Italian operas.

The research starts with the identification of the emotion at the verse level. After a series of experiments on various text representations(e.g, Latent semantic analysis, latent dirichlet allocation, Term Frequency - Inverse Document Frequency based on character 3-grams, Term Frequency - Inverse Document Frequency based on words) and different models (e.g., k-nearest neighbors algorithm, support-vector machines, neural network), the combination of Term Frequency - Inverse Document Frequency based on character 3-grams and a 2-layer feed-forward neural network achieved the best performance. However, even the best performance model is not good in predicting emotion on the verse level and has thus no practical value, considering that one out of two predictions is wrong.

The failure in the attempts to identify emotion at the verse level is attributed to several reasons. First, the length of each verse is relatively short, therefore it does not contain much valuable information. Second, the training material at hand was short (2,500 annotated verses). Third, since the arias are written in 18th Italian, several publicly available embeddings are not effective representation alternatives.

In order to address these issues, the second stage of research moved to a higher granularity and concentrated on predicting emotion transmitted by an entire aria. The AriEmozione 2.0 corpus was constructed and annotated automatically in order to increase the size of available training data. As for the issue of embeddings, novel 300-dimensional character 3-gram embeddings were trained based on the CORAGO-1700 corpus. These embeddings were then used as text representation to explore the emotion transmitted at the aria level. As for models, CNNs of different configurations are tested and achieved promising results.

6.2 Research Products

This research has produced two publications and two corpora.

The publications are:

1. Francesco Fernicola, Shibingfeng Zhang, Federico Garcea, Paolo Bonora and Alberto Barrón Cedeño. AriEmozione: Identifying emotions in opera verses. In *Italian Conference of Computational Linguistics*, 2020[1]

2. Shibingfeng Zhang, Francesco Fernicola, Federico Garcea, Paolo Bonora and Alberto Barrón Cedeño. AriEmozione 2.0: Identifying emotions in opera arias. Selected follow-up paper for the *Italian Journal of Computational Linguistics* (to appear).

The corpora are:

1. AriEmozione 1.0: a corpus contains about 2,500 Italian opera verses written in 18th century Italian. Each verse has its unique ID and an emotion label. The annotation of this corpus is done manually by two human annotators.

[1]This publication is available at `http://ceur-ws.org/Vol-2769/paper_58.pdf`.
The AriEmozione 1.0 corpus is available at `https://zenodo.org`.
The code is available at `https://github.com/TinfFoil/AriEmozione`.

2. AriEmozione 2.0: a corpus contains about 70,000 Italian opera verses written in 18th century Italian. Each verse had its unique ID and an emotion label. The annotation of this corpus is done by text classification models.

6.3 Future Work

As for future work, the annotation of the AriEmozione 2.0 corpus could be refined considering a finer granularity for a label distribution learning task (Zhao and Ma, 2019). In stead of assigning to each aria one or two emotion labels, the emotion transmitted by each aria could be represented using not only a set of emotion labels, but also the corresponding emotion intensity values these labels.

Instead of performing an automatic annotation by self-learning, I can construct an emotion lexicon on the basis of the AriEmozione 1.0 corpus, and then use the emotion lexicon to annotate AriEmozione 2.0. An interesting research avenue is to consider multi-modal aspects of the Italian operas. Not only the written verses, but also the sheet music and even the scene representations to will be taken into consideration to refine the emotion decision.

Bibliography

Diab Abuaiadah, Jihad El Sana, and Walid Abusalah. On the impact of dataset characteristics on arabic document classification. *International Journal of Computer Applications*, 101(7), 2014.

Mahmoud Al-Ayyoub, Safa Bani Essa, and Izzat Alsmadi. Lexicon-based sentiment analysis of arabic tweets. *International Journal of Social Network Mining*, 2(2):101–114, 2015.

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. Emotions from text: machine learning for text-based emotion prediction. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 579–586, 2005.

Ebba Cecilia Ovesdotter Alm. *Affect in text and speech*. PhD thesis, University of Illinois at Urbana-Champaign, 2008.

Mohamed Aly. Survey on multiclass classification methods. *Technical Report*, 19:1–9, 2005.

Muhammad Zubair Asghar, Aurangzeb Khan, Afsana Bibi, Fazal Masud Kundi, and Hussain Ahmad. Sentence-level emotion detection framework using rule-based classification. *Cognitive Computation*, 9(6):868–894, 2017.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X.

Michael Burden. The new grove dictionary of opera, ed. stanley sadie. *Early Music*, 26(4):669–670, 1998.

Erik Cambria, Robert Speer, Catherine Havasi, and Amir Hussain. Senticnet: A publicly available semantic resource for opinion mining. In *AAAI fall symposium: commonsense knowledge*, volume 10, 2010.

Erik Cambria, Dipankar Das, Sivaji Bandyopadhyay, and Antonio Feraco. *A practical guide to sentiment analysis*. Springer, 2017.

Cicero Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

Paul Ekman. All emotions are basic. *The nature of emotion: Fundamental questions*, pages 15–19, 1994.

Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *The International Conference on Language Resources and Evaluation*, volume 6, pages 417–422, 2006.

Ronen Feldman, Benjamin Rosenfeld, Roy Bar-Haim, and Moshe Fresko. The stock sonar—sentiment analysis of stocks based on a hybrid approach. In *Twenty-third IAAI conference*, 2011.

Francesco Fernicola, Shibingfeng Zhang, Federico Garcea, Paolo Bonora, and Alberto Barrón-Cedeño. Ariemozione: Identifying emotions in opera verses. In *Italian Conference on Computational Linguistics*, 2020.

John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

Joseph L Fleiss, Jacob Cohen, and Brian S Everitt. Large sample standard errors of kappa and weighted kappa. *Psychological bulletin*, 72(5):323, 1969.

Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics, 2001.

Sonja Gievska, Kiril Koroveshovski, and Tatjana Chavdarova. A hybrid approach for emotion detection in support of affective interaction. In *2014 IEEE International Conference on Data Mining Workshop*, pages 352–359. IEEE, 2014.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. *arXiv preprint arXiv:1802.06893*, 2018.

Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 986–996. Springer, 2003.

Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

Mohammad Hossin and MN Sulaiman. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, 5(2):1, 2015.

Yajie Hu, Xiaoou Chen, and Deshun Yang. Lyric-based song emotion detection with affective lexicon and fuzzy clustering method. In *International Society for Music Information Retrieval*, pages 123–128, 2009.

VS Jagtap and Karishma Pawar. Analysis of different approaches to sentence-level sentiment classification. *International Journal of Scientific Engineering and Technology*, 2(3):164–170, 2013.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. ApplicaAI at SemEval-2020 Task 11: On RoBERTa-CRF, Span CLS and whether self-training helps them. *arXiv preprint arXiv:2005.07934*, 2020.

DK Kirange and RR Deshmukh. Emotion classification of news headlines using svm. *Asian Journal of Computer Science and Information Technology*, 5(2):104–106, 2012.

Fazal Masud Kundi, Aurangzeb Khan, Shakeel Ahmad, and Muhammad Zubair Asghar. Lexicon-based sentiment analysis in the social web. *Journal of Basic and Applied Scientific Research*, 4(6):238–48, 2014.

Hobson Lane, Cole Howard, and Hannes Hapke. *Natural Language Processing in Action*. Manning Publications, 2019.

Ling Liu and M Tamer Özsu. *Encyclopedia of database systems*, volume 6. Springer New York, NY, USA:, 2009.

D'Addona Doriana Marilena. *Neural Network*, pages 911–918. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Saif M Mohammad and Peter D Turney. Crowdsourcing a word-emotion association lexicon. *Computational intelligence*, 29(3):436–465, 2013.

Finn Årup Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*, 2011.

William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

W Gerrod Parrott. *Emotions in social psychology: Essential readings*. Psychology press, 2001.

Robert Plutchik. A general psychoevolutionary theory of emotion. In *Theories of emotion*, pages 3–33. Elsevier, 1980.

Angelo Pompilio, Lorenzo Bianconi, Fabio Regazzi, and Paolo Bonora. Radames: A new management approach to opera: repertory, archives and related documents. In *First International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'05)*, pages 4–pp. IEEE, 2005.

Chris Pool and Malvina Nissim. Distant supervision for emotion detection using Facebook reactions. *arXiv preprint arXiv:1611.02988*, 2016.

Kirk Roberts, Michael A Roach, Joseph Johnson, Josh Guthrie, and Sanda M Harabagiu. Empatweet: Annotating and detecting emotions on twitter. In *International Conference on Language Resources and Evaluation*, volume 12, pages 3806–3813, 2012.

Klaus R Scherer. Expression of emotion in voice and music. *Journal of voice*, 9(3):235–248, 1995.

TK Shivaprasad and Jyothi Shetty. Sentiment analysis of product reviews: a review. In *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 298–301. IEEE, 2017.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

Carlo Strapparava and Rada Mihalcea. Semeval-2007 Task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 70–74, 2007.

Carlo Strapparava, Alessandro Valitutti, et al. Wordnet affect: an affective extension of wordnet. In *International Conference on Language Resources and Evaluation*, volume 4, page 40, 2004.

Erik Tromp and Mykola Pechenizkiy. Rule-based emotion detection on social media: putting tweets on Plutchik's wheel. *arXiv preprint arXiv:1412.4682*, 2014.

Peter D Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. *arXiv preprint cs/0212032*, 2002.

Jun Yan. *Text Representation*, pages 3069–3072. Springer US, Boston, MA, 2009.

Changli Zhang, Daniel Zeng, Jiexun Li, Fei-Yue Wang, and Wanli Zuo. Sentiment analysis of chinese documents: From sentence to document level. *Journal of the American Society for Information Science and Technology*, 60(12):2474–2487, 2009.

Wenhao Zhang, Hua Xu, and Wei Wan. Weakness finder: Find product weakness from chinese reviews by using aspects based sentiment analysis. *Expert Systems with Applications*, 39(11):10283–10291, 2012.

Zhenjie Zhao and Xiaojuan Ma. Text emotion distribution learning from small sample: A meta-learning approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3948–3958, 2019.