Master's Thesis in Computer Science: Recognizing artist ambiguity with machine learning techniques

Alejandro Machado González

June 19, 2012

Abstract

Spotify is a music streaming service whose ambition is to offer everyone easy access to all of the world's music. Maintaining metadata quality is a a non-trivial challenge considering that the catalogue size rules out the possibility for manual curation and that the number of content providers and different delivery formats is large.

An interesting problem that involves metadata is to have a one-to-one mapping between real-world and Spotify artists: two different artists should be two separate entities on Spotify. Currently, when several albums by artists with the same name are sent to the service, they are arranged under the same artist identifier, assuming that they belong together. This is an issue when two artists with the same name have their albums on Spotify: all the albums are grouped together, so users think they belong to the same artist and fans have a bad user experience when presented with additional clutter besides their favorite artist's releases.

This thesis explores the use of machine learning techniques to detect Spotify artists credited as having produced albums that actually belong to several real-world artists, namely *ambiguous artists*. Several features for representing the artists are presented, such as the existence of multiple matches between the Spotify artist and external music databases with curated content and the number of countries the artist has registered recordings in. Using every possible combination of the features, the examples are classified with Naive Bayes and logistic regression. Two of the resulting best performant classifiers with low false positive rates are then queried with unseen data sets of random and most popular artists to assess their predictions.

We found that the most useful features for the classification were the existence of multiple matches between the Spotify artist and the external music databases, the artist's name length and the number of languages of the artist's track names. Logistic regression proved to be superior to Naive Bayes on a test set of random artists.

When the classifier has detected the ambiguous artists, after a manual artist separation process, heuristics can be put into place so that incoming albums that could belong to more than one artist will be assigned to the most likely one. This practical solution to the artist ambiguity problem is also briefly discussed.

Contents

Chapter 1

Introduction

1.1 Spotify

Spotify is a platform for listening to music online. It provides users with the possibility of streaming a vast catalogue of more than 18 million tracks (Spotify, 2012), produced by a very large number of different artists and bands from the whole world.

Spotify was conceived as a service that gives people easy access to music and ensures that the artists benefit from it. The company has negotiated business agreements with the world's four major record labels (EMI, Sony BMG, Universal Music and Warner Music), as well as many smaller and independent labels that represent the artists. These agreements allow the online service to stream music to the users on demand.

The labels independently send music track files and tags identifying them to Spotify, in batches known as *products*. Products can be albums, singles or compilations. They are then organized by the *Spotify Content System* and served to the users. Spotify retains data on how many times every track has been streamed, and then reimburses the record labels according to the business deals that have been agreed.

Every artist in Spotify has an **artist page**, a section where the music of the artist is featured along with a biography, when it is available, and a list of related artists. Only albums and tracks that belong to the artist should be included in the artist page. Figure 1.1 shows an example of such a page.

An interesting fact is that the vast majority of the streams requested to Spotify come from a very reduced subset of tracks; that is, the most popular tracks are streamed very frequently, by many users. This implies that we can prioritize the improvement of data for the most popular artists and achieve a very significant enhancement of the user experience without excessive effort.

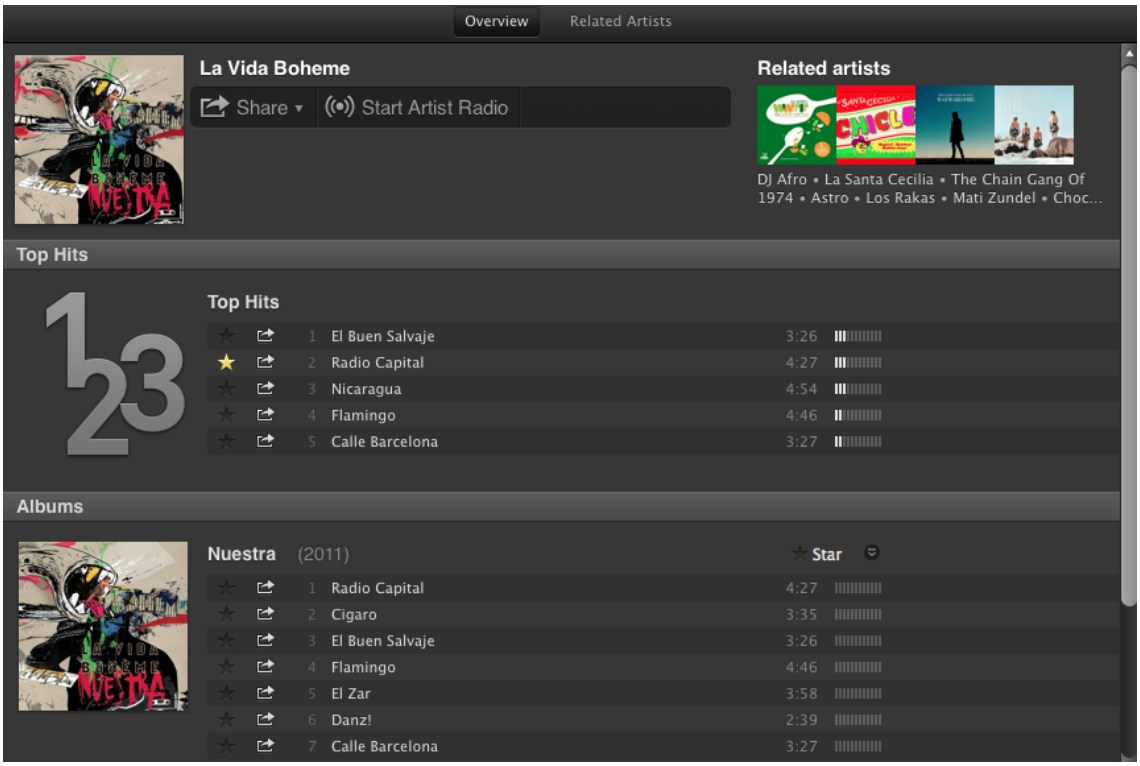

Figure 1.1: Artist page of *La Vida Bohème*.

1.2 Artist ambiguity

The tags that accompany and identify the audio files sent by the record labels are aggregated on a relational database internal to Spotify, called `metadata`. The database holds primarily information on artists, tracks, albums and relationships between them.

When a new music delivery is received by Spotify, some decisions must be made with respect to organization. The products do not include an artist identifier, since there is no universal, standard system for identifying artists in the industry. Since the deliveries can come from many different sources, it is a critical decision to determine whether the newly delivered product should belong to an already-existing artist, or to a new artist that should be created in the database. This matter is especially difficult if there is a **Spotify artist** whose name has the exact same spelling as the *artist name* tag in the incoming product: it is likely that both releases are by the same artist, but it is also possible that the new product belongs to a completely different, unrelated artist, albeit with the same name.

A **Spotify artist** will be considered *ambiguous* if there are at least two products attributed to it that actually correspond to the work of two different real-world artists. In Figure 1.2 we can see an example of an artist page that contains an album from the popular US-based singer Prince and another one from a Caribbean artist.

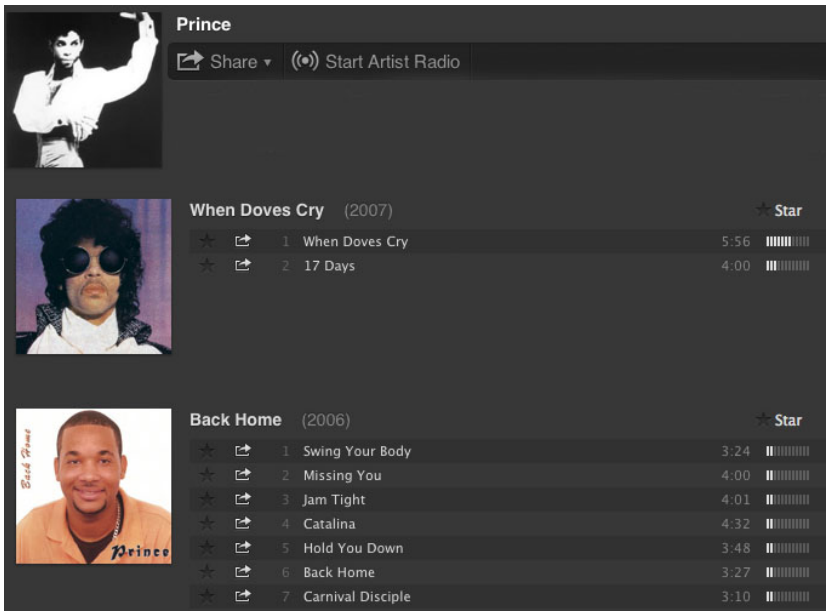

Figure 1.2: Partial view of Prince's artist page. It shows two albums produced by different artists, both having the same name.

At present, Spotify assigns the same artist identifier to all products tagged with the exact same artist name. Some artists with the same name have been manually disambiguated and split into two different entities by request of the record labels.

Ambiguous artists is not a problem unique to Spotify: other music providers, such as last.fm, face the exact same situation. In their music management system, they state that they are unable to separate the different bands that have the same name (last.fm, 2012).

Ambiguous artists are inconvenient for the user experience. When a user navigates to an artist page, he or she wants to visualize albums that have been produced by a single artist. It is confusing to encounter albums that are from the same artist according to Spotify, but have actually been produced by many different artists.

The goal of this thesis is to **develop a system that will be able to perform an automatic recognition of ambiguous artists present in**

the Spotify metadata database. Ideally, the system will produce as low a number of false positives as possible, because the ambiguous candidates will then be reviewed by a human team for their disambiguation.

1.3 Machine learning

A first approach to solving the artist ambiguity problem is to come up with a well defined algorithmic procedure based on the data that determines whether an artist is ambiguous or not. Since a way of deriving such a procedure is not obvious, we may try to solve the problem with the use of machine learning.

Learning from data has become ubiquitous in today's world, both in technology-oriented and other organizations. Whenever a system can benefit from historical data, but it is not clear what the decision rules should be, statistical learning can come in handy. Recognizing handwriting, automatic language detection and spam filtering are very common examples of problems that may be solved using these techniques.

Applying machine learning techniques to solve a problem most often means that the accuracy of the solution will not be *perfect*. To progressively overcome mistakes in the classification, it is common practice to perform several iterations, try different approaches and alter the parameters until, judging by some pre-defined metric or manually evaluating the test data, the system performs well enough

In *supervised* machine learning there is a **result** that needs to be approximated, based on a combination of factors, called *features*, that are presumably correlated to this result. There is a set of tagged examples, which must generally be produced non-algorithmically, that is called the training set. Machine learning algorithms process this data and try to create a prediction model for the outcome.

The problem that this thesis will address is a typical instance of a supervised machine learning problem, because:

- There is an outcome whose prediction is needed: whether an artist is ambiguous or not.
- There is a significant amount of data available about the entities involved, i.e. artists and albums, but no obvious way of processing this data to decide artist ambiguity.
- The system does not need perfect performance: improving the data on the most popular artists will satisfy the needs of most users.

Chapter 2

Scheme

2.1 Data

Since the system needs to identify which artists in `metadata` are ambiguous, the training and test sets will consist of artists, represented by feature vectors. In this section there will be a discussion about what kind of artist-related data might be useful for deciding artist ambiguity, while a more detailed description of the most relevant features will be explained in Chapter 3.

The `metadata` database is constantly changing, accommodating new content every day. In order for the experiments to be reproducible, it has been decided to work with a database snapshot that will remain constant over time.

Out of the total number of artists that the snapshot contains, only around 15% have two or more albums, and thus have the potential to be ambiguous.

2.1.1 Internal data

Spotify's `metadata` contains relationships between artists, albums and tracks. A simplified version of the ER diagram, with only relevant information, is presented in Figure 2.1.

As has been discussed in the introduction, artist tags do not come with a unique identifier: artist names are used as *de facto* identifiers to differentiate an artist from another.

Album data includes the name of the album and the name of the record label that owns its digital rights.

The labels provide every track they send to Spotify with a name. Many tracks also come with an International Standard Recording Code (ISRC), that in turn allow to identify the country where the recording was registered (Wikipedia, 2012). These codes are unique for every *recorded version* of a

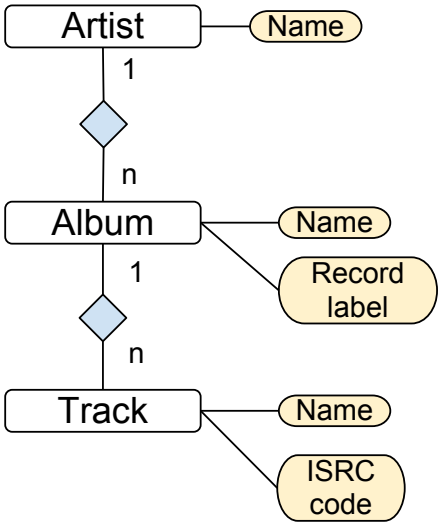

Figure 2.1: Simplified ER diagram of `metadata`.

track, and do not necessarily reflect where a piece of music may have been composed, produced or performed, nor the nationality of the artist.

2.1.2 External data sources

Description

There are several companies and websites dedicated to curating music metadata. These sites have independently built a collection of information about artists, albums and tracks that differs from the data available in the Spotify catalogue. External sources may have data that associates two different artist identifiers to two albums that Spotify has mistakenly assigned to a single artist.

This is very useful to detect ambiguous artists: if there were a *perfect* source, with correct and complete data, it would be possible to detect ambiguous artists with the following pseudocode.

```
function is_ambiguous(artist):
    matches = look up artists with the same name as artist
              in perfect source
    for candidate in matches:
        if album_matches(artist, candidate):
            mark artist and candidate as a match
```

```
    if artist matches more than one external artist:
        return True
    else:
        return False

function album_matches(artist1, artist2):
    for album1 in artist1:
        for album2 in artist2:
            if album1 == album2:
                return True
    return False
```

This could be a very simple way of recognizing ambiguous artists. However, external data sources do not always have correct data, nor are they complete. The names of the artists and albums may vary slightly between Spotify's database and external sources, so an algorithm that accommodates normalized name matching is necessary to get reasonable results.

Approximate name matching

It has been noted that many of the string mismatches are due to differences in non-alphabetical characters such as spaces, dashes, punctuation signs and leading articles (e.g. *the*, *a*). Thus, the algorithm proposed will **normalize** both names by removing these problematic elements and compare them.

To filter out edge cases where two normalized names are the same, but the actual names differ significantly, a string similarity check can be carried out between the two names. String similarity will be defined as the simple edit distance (Levenshtein, 1966), normalized with respect to the string's length.

Below is the pseudocode for a function that decides whether two strings can be considered a *match*.

```
function string_matches(a, b):
    if normalized_name(a) == normalized_name(b) and
       string_similarity(a, b) >= THRESHOLD:
            return True
    return False
```

`string_similarity` returns a value between 0 and 1 that expresses how alike two strings are. We have, after trying different values, found that a reasonable threshold is 0.85, but a more careful study of this parameter could probably yield improvements to the quality of the matcher.

We can incorporate these ideas to devise an improved version of the `is_ambiguous` function, in which we look up artists in the external sources that have *approximate* string matches to the Spotify artist, instead of exact matches.

Comparison of external databases

The three most significant curated music databases that Spotify has access to, described below, will be matched to the Spotify catalogue.

All Media Guide (AMG), formerly All Music Guide, is a US-based company founded with the purpose of compiling a thorough database of music (Guide, 2012), employing editorial staff. Their data, although far from perfect, covers an extensive segment of all recorded music. Spotify purchases access to All Media Guide's music database and includes some biographies and pictures from it in the artist pages to provide a richer experience.

MusicBrainz is an open-source initiative that seeks to maintain an updated and reliable database of music information, freely available on the web (MusicBrainz, 2012).

Discogs is another US-based company that offers an online database with artists and recordings data, and encourages users to submit and correct metadata entries (Discogs, 2012).

A comparison between the three databases was performed in order to select which of them were relevant for the detection of artist ambiguity. Table 2.1 shows the number of Spotify artists that have been matched to external sources with at least one album in common. The matching was done as described previously, taking string similarity into account.

	# of matching artists
AMG	173,394
MusicBrainz	75,591
Discogs	70,306

Table 2.1: Comparison of external sources of music information. All the databases are snapshots taken on February 6th, 2012.

AMG seems to offer the greatest potential for discovering ambiguities, since a greater number of matchings could be established. While MusicBrainz and Discogs have a smaller amount of data directly matchable to the Spotify catalogue, their crowdsourced, open access edit system make them very interesting sources to take the study further.

2.2 Classifiers

There are many different algorithms to choose from in the machine learning literature. Depending on the data available and the result desired, different classifiers may fit the bill.

The purpose of this thesis is to classify Spotify artists as ambiguous or non-ambiguous, with as little an error rate as possible. Dealing with this from a statistical point of view, the null hypothesis for every artist is that it is not ambiguous. Since the company is interested in resolving the ambiguities of the most popular artists and manual work is required to assign every album to a correct artist, there is a preference for reducing Type I errors; that is, false reports of ambiguous artists.

Errors in classification come from the bias and variance of the model, and there is a tradeoff between these two (Access, 2012). Overfitting of the training data is a common problem when using complex models (Hastie et al., 2009): the classifier becomes "used" to the training data and is not able to generalize so well. High-bias, low-variance models are often suitable when the amount of tagged data is limited, because the risk of fitting the data too hard to the features exhibited by the training examples is lower, and in this particular problem, time-consuming manual work is required to tag examples of artists according to ambiguity. Therefore, two simple model proposals for the ambiguous artist detection system will be described below.

2.2.1 Naive Bayes Classifier

This model is based on Bayes' rule, a useful theorem in the frequentist interpretation of statistics and a foundation of subjective probability that has made a comeback in the classification research field (Lewis, 1998).

The classifier works by estimating the probability that an artist belongs to a particular class (ambiguous or non-ambiguous) given its feature vector.

$$P(a_k \mid x) = \frac{P(a_k) \times P(x \mid a_k)}{P(x)} \tag{2.1}$$

a_k stands for the *ambiguity* of the artist, where a_1 is ambiguous and a_0 is non-ambiguous. Under this model, the probability that an artist belongs to the class a_k, provided that it has a feature vector x depends on:

- $P(a_k)$, the probability that an artist belongs to a given class
- $P(x)$, the probability that an artist has the feature vector $x = (x_1, ..., x_n)$.

- $P(x \mid a_k)$, the probability that a feature vector is observed given that it is known whether the artist is ambiguous or not. To estimate this, one may assume that the values of x_i are independent, thus $P(x \mid a_k) = \prod_{i=1}^{n} P(x_i \mid a_k)$, where n is the number of elements in the feature vectors. This assumption gives rise to the denomination *naive Bayes*.

2.2.2 Logistic regression

Statistical linear regression is frequently used for predicting the value of a continuous variable Y as a linear combination of one or more explanatory variables X_i. The goal of linear regression is to find a coefficient vector $(\beta_1, ..., \beta_n)$ that minimizes a predefined error metric such as least-squares. An example is shown in figure 2.2.

$$Y = \sum_{i=1}^{n} \beta_i X_i + \beta_0 \tag{2.2}$$

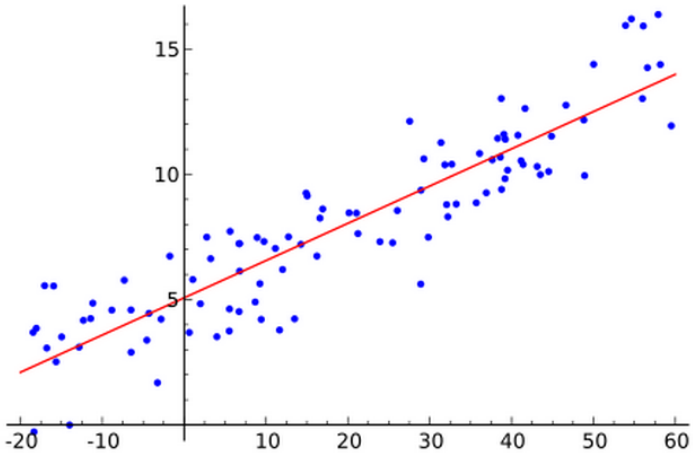

Figure 2.2: Data points (blue) being fitted by a straight line by linear regression, using least squares as the error metric to minimize.

This idea can be extended and improved for the purpose of performing binary classification, resulting in a model called *logistic regression*. The two classes are modeled with the values 0 and 1: in this case, 0 can represent a non-ambiguous artist while 1 stands for an ambiguous artist as classified by the learner. This model also assumes that the explanatory variables are independent, a goal that will be taken into account as much as possible

when the artist features are defined. The advantages of logistical over linear regression for this use case include: (Brannick, 2007)

- The result of a linear regression can give theoretically impossible results on the predicted values, greater than 1 or lower than 0. The expected value of the regression equation should be bounded between these two values.

- Logistic regression does not assume that the variance of Y is constant across values of the X_i. This assumption is not reasonable for a binary target variable, where the maximum variance is 0.25.

- Linear regression assumes that the errors of prediction are normally distributed. This is not true when the outcome variable is dichotomous: the *binomial* distribution describes the errors (Hosmer and Lemeshow, 2000).

With equation 2.3, logistic regression estimates the probability that the artist belongs to the class a_k given that its feature vector is $x = (x_1, ..., x_n)$, fitting the coefficients $(\beta_1, ..., \beta_n)$ to minimize error.

$$P(a_k \mid x) = \frac{e^{\sum_{i=1}^{n} \beta_i x_i + \beta_0}}{1 + e^{\sum_{i=1}^{n} \beta_i x_i + \beta_0}} \tag{2.3}$$

The model gets its name from the logistic function, the basis of the predictor engine. A particularly useful consequence of this is the availability of *probability estimates* based on the expected value of the predicted variable.

This fact gives a convenient advantage to logistic regression over other machine learning algorithms like the perceptron rule and linear support vector machines (Russell and Norvig, 2010, p. 745), provided that the classifier's performance is good: one can set a threshold on the probability estimate after the classification and further reduce the number of false positives if desired. Linear support vector machines act as black boxes, and it is a complex task to obtain probability estimates out of them despite some research efforts (Platt, 1999).

2.3 Tools

Two different machine learning software packages were used for the experiments.

LIBLINEAR (Fan et al., 2008) is an open-source library that features a very efficient implementation of logistic regression. It offers the possibility to get probability estimates for every classified example.

Weka (Hall et al., 2009) is a versatile and fully featured environment for performing machine learning experiments, including classification, clustering and attribute selection. It offers support for 10-fold cross-validation, which consists in splitting the training set in ten equal parts, using nine of them as the training set and the remaining as the test set. This is iterated ten times, until all the examples have been classified.

Cross-validation is able to estimate the prediction capability of a classifier much more effectively than a simple evaluation on the training set, and it will be the main tool used to evaluate the different classifiers and choose one of them. Additionally, its interface can be used with the LIBLINEAR backend.

Chapter 3

Description of the data set

This chapter describes the process followed to obtain a system capable of recognizing ambiguous artists.

3.1 Preliminary study

It is convenient to have a rough estimation of the number of ambiguous artists that could be present in the Spotify catalogue. Let us assume that a completely correct source of music metadata, with the same set of artists and albums as Spotify, is available to query. Taking into account that Spotify assigns albums tagged with the same artist name to one unique artist, it is clear that the Spotify artist whose name is N is ambiguous if and only if more than one artist in the perfect database has name N. Thus, to know the proportion of ambiguous artists in the Spotify database, it would suffice to count how many repeated names there are in the correct source and then divide it by the total number of names. This is illustrated in Figure 3.1.

It is possible to approximate the number of ambiguous artists that Spotify would contain if its catalogue was identical to an external database using this same approach. This procedure incurs errors if the external source also contains ambiguous artists or duplicate entries for a single real-world artist, but it is useful as an estimate.

3.2 Acquiring a training set

Judging by the estimate, it appears that a significant majority of the Spotify artists are non-ambiguous. As in spam detection (Erdélyi and Benczúr, 2011) and many other applications of machine learning, one of the classes is rare.

	percentage of would-be ambiguous artists
AMG	5.52%
MusicBrainz	4.42%
Discogs	7.17%

Table 3.1: Proportion of Spotify artists matching more than one name in the external databases.

Rather than attempting to maintain the estimated ambiguous-to-non-ambiguous ratio, it is a common practice to oversample the minority class to achieve a balanced data set. Doing this changes the underlying statistical problem, but if one is looking for good performance for the minority class, having a balanced data set will yield in general better results (Hlaváč, pp. 14-15). Moreover, there are sound theoretical justifications for the case of oversampling the rare class when using classification by logistic regression Scarpa and Torelli (2005). A thorough explanation of this can be found in this paper.

There is no readily available training data: it must be tagged from scratch, requiring human effort. If the data labeling was to be done completely at random, a great amount of hours of manual work would be required to achieve an example pool of a relevant size.

Artist ambiguity is a fairly common complaint in Spotify's Community forums. A web form[1] where users can report identifiers of ambiguous entities is provided and linked from the support forums. Since the input format was free text, the submissions were reviewed and it was clear that some people tried to report other kinds of problems, such as wrong track order in albums, wrong track metadata and misspellings in artists' names. Data was extracted from these user submissions, examined and tagged manually; with it, a first data set of 200 non-ambiguous and 200 ambiguous artists was compiled. Deciding the size of the data sets is not always a straightforward procedure: we consider that a set of 400 examples is a good compromise between the time spent on the manual tagging process and obtaining a number of examples that is statistically significant.

[1]Form available at `https://docs.google.com/a/spotify.com/spreadsheet/viewform?formkey=dFVLNW1MMWYtNG83RDUxaGw0ckdNdFE6MQ`

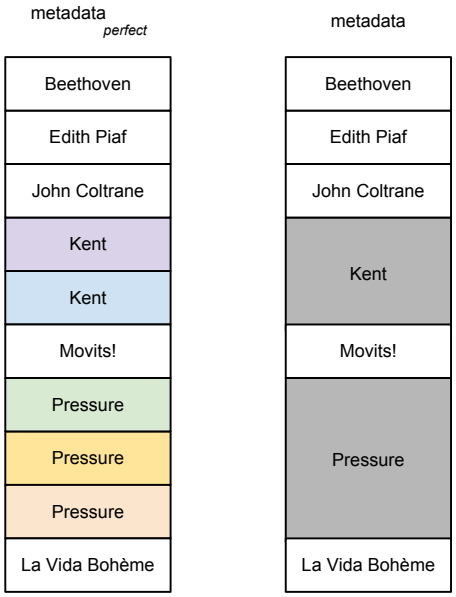

Figure 3.1: Visual representation of a *perfect* version of the Spotify metadata (left part of the figure), where every real-world artist is considered unique, next to the current state of `metadata` (right part of the figure). The proportion of ambiguous artists can be calculated by counting how many names appear more than once in the perfect source: in this example, it is 2/7.

3.3 Feature search

After obtaining a data set, we designed the vector of features representing every artist. This section will describe the different sources of information and present the most interesting features. Each feature will be referred by its unique name in the rest of this report. Every feature will include a bar plot with the distribution of ambiguous (red) and non-ambiguous (blue) artists that fall into its different possible values.

3.3.1 Artist matcher

As has been argued previously, the external sources could provide a practical ambiguity oracle with the `is_ambiguous` function. The sources are queried with the binary-response question of interest: does the artist *match* multiple artists on this source? A decision could then be taken by combining the multiple responses in some way. It is not obvious how to aggregate them, or whether or not all the sources provide relevant information (for instance, a source may have many duplicates, falsely reporting artist ambiguity). This

is where machine learning comes into play.

Three features can be extracted from and the external sources, using the artist matcher:

Feature name	Description	Type
match-amg	Does the Spotify artist match multiple entities in AMG?	binary
match-mbz	Does the Spotify artist match multiple entities in MusicBrainz?	binary
match-dgs	Does the Spotify artist match multiple entities in Discogs?	binary

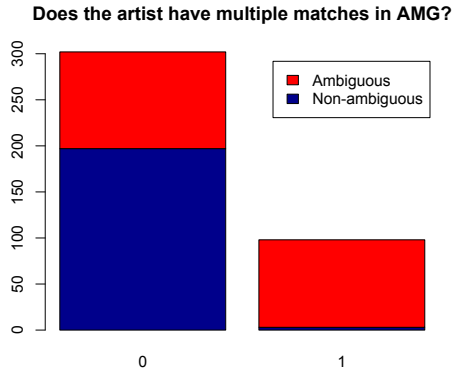

Figure 3.2: Distribution of classes in the test set for the feature **match-amg**

The **match-amg** (see Figure 3.2) feature exhibits a clear correlation with ambiguous artists: for the examples analyzed, if this particular feature takes the value 1 (i.e., if the Spotify artist matches multiple artists in AMG), the probability of the artist being ambiguous is very high. However, there is a significant amount of ambiguous artists that do not have multiple AMG matchings. If this feature alone was used to decide ambiguity, many artists with problems would remain undiscovered.

For the MusicBrainz external source (see Figure 3.3), the number of examples that exhibited multiple matches were significantly lower than for AMG. In any case, it can be inferred from the plot that any artist with multiple matches to MusicBrainz is most likely ambiguous.

The feature based on Discogs (see Figure 3.4) had a very similar behavior to the one based on MusicBrainz. About an eighth part of the examples were matched to multiple artists in the external source, and almost all of them were correctly identified as ambiguous by this feature.

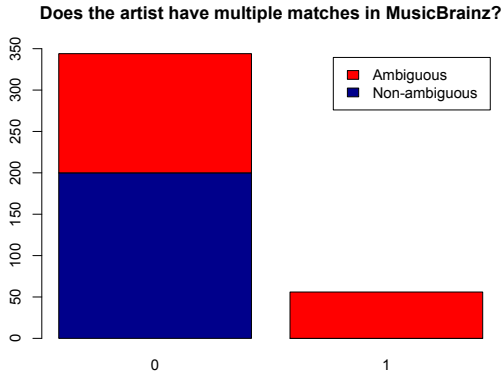

Figure 3.3: Distribution of classes in the test set for the feature **match-mbz**

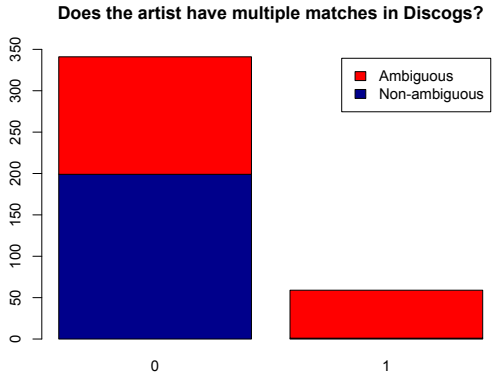

Figure 3.4: Distribution of classes in the test set for the feature **match-dgs**

3.3.2 Countries from ISRC codes

The first two characters of an ISRC code reflect the country of registration of a specific track. It is, therefore, possible to query the number of countries an artist has registered recordings in.

This gives rise to the following feature:

Feature name	Description	Type
country-count	In how many countries has the Spotify artist registered recordings?	integer

In the plot (Figure 3.5) we can see that the majority of artists that have only registered recordings in one country are non-ambiguous, and as the number of countries increases, the fraction of ambiguous artists within the class tends to increase.

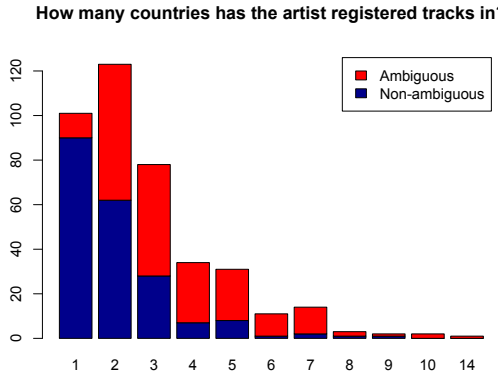

Figure 3.5: Distribution of classes in the test set for the feature **country-count**

3.3.3 Artist name length

In the process of data tagging, we noted that a correlation existed the length of an artist name and its probability to be ambiguous (see Figure 3.6). This makes intuitive sense: the shorter an artist name is, the probability that another artist has chosen the same name is higher.

Feature name	Description	Type
name-length	How many characters does the Spotify artist name have?	integer

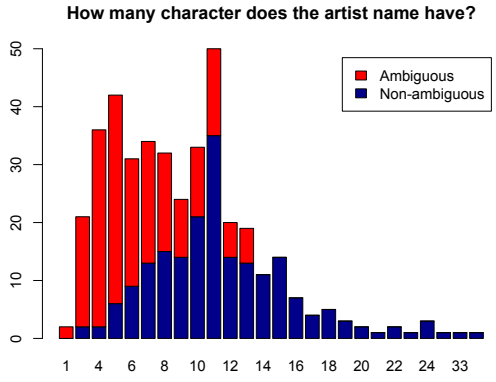

Figure 3.6: Distribution of classes in the test set for the feature **name-length**

The intuition for choosing this feature is justified with this plot: it is evident that artists with short names of this example set tend to be ambiguous.

3.3.4 Record labels

Another idea to recognize whether an artist is ambiguous is to consider the number of record labels under which they have released albums (see Figure 3.7). This feature would be useful if the majority of real-world artists usually published products under only one label: then Spotify artists that use many labels would have a greater probability of being ambiguous.

Feature name	Description	Type
label-count	How many labels does the Spotify artist name have?	integer

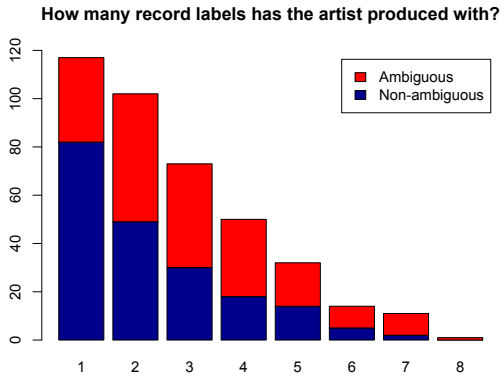

Figure 3.7: Distribution of classes in the test set for the feature **label-count**

Using the graph, it is difficult to establish a pattern of how the ambiguous fraction of the artist changes according to how many labels the artists have signed under. It may be possible that the feature adds certain information for the classification, but the relationship is not clear with this plot.

3.3.5 Language

It is a reasonable assumption that most music artists use only one language for the names of their tracks. A Spotify artist that has produced albums with tracks in different languages could have a greater chance of being ambiguous. Only a subset of the ambiguous artists contain at least two albums in different languages: the features in this section are aimed at discovering artists from this subset.

There is no readily available data in Spotify for language of tracks and albums, however, advancements in natural language processing allow to detect the language of a document with a reasonable precision (more than 80%). In

particular, Gottron and Lipka (2010) explores the identification of very short texts, such as metadata items (e.g. track names). We have contacted the first author of this publication, and he has been kind enough to provide an implementation of his system that could be used to determine the language of metadata elements.

The goal is to determine the most likely language for every album, and build a relevant feature for every artist with this information. It is possible to determine the language of the track names that compose an album in two different ways:

- Guess the language of every individual track, then perform a majority voting to establish the language of the album
- Every album is represented by its title tracks as sentences separated by full stops: the language guessed for this representation will establish the language of the album

Language recognition techniques, especially the ones based on n-gram models, are often more accurate if the documents to analyze are longer, so the latter choice may be more suitable.

It is important to note that the classifier provided by Gottron and Lipka has been trained with corpora of ten popular European languages, but it does not cover all the languages that may appear in the Spotify catalogue. This may be a source of error.

The relevant feature for the artist is if it has produced two or more albums in different languages, judging by the track names of these albums. See Figure 3.8.

Feature name	Description	Type
multilingual	Does the artist Spotify have at least two albums in different languages, when the language of an album is decided with a tracks-as-sentences representation?	binary

The plot in Figure 3.8 shows that the fraction of ambiguous entities is higher for artists that have at least two albums in different languages, according to the language identifier. However, only about half of the ambiguous artists are "recognized" by this feature, showing that there are many artists with the same name that name their tracks using the same language. On its own, this feature would miss a substantial amount of ambiguous artists.

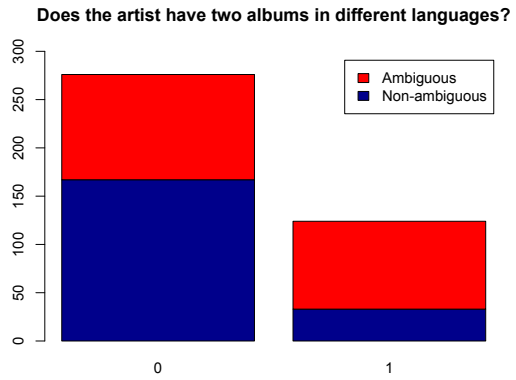

Figure 3.8: Distribution of classes in the test set for the feature **multilingual**

Chapter 4

Experiments and results

In the following chapter we discuss how we chose the optimal feature subsets out of the studied features, for both the Naive Bayes classifier and logistic regression, then show a comparison between the two best classifiers when used to predict artists from two different populations: random and popular.

4.1 Choosing the best feature set

4.1.1 CFS attribute selection

Hall (1998) proposes an algorithm to select a subset of attributes available in a dataset called *Correlation based Feature Selection (CFS) subset evaluation*. A "good feature set", according to this work, should contain attributes strongly correlated to the class but not strongly correlated to each other.

The algorithm uses heuristic search with greedy hill-climbing (Hall, 1998, p. 49) to determine the best feature subset. It is useful for acquiring a first insight of which features are the most relevant for the classification, but it has some drawbacks.

First, it is designed for datasets with a large number of features for which it would be computationally unfeasible to try every possible feature combination, so it may yield suboptimal assessment as to what feature set will produce the best classification. With as little as seven features, it is feasible to train and perform cross-validation on all the possible ($2^7 - 1$, excluding the empty set) feature sets in a few minutes. Secondly, the recommendations this algorithm gives as to what feature vector to use are classifier-independent, and may not always adjust to the classifiers that the experimenters will use in practice.

A run of the `CfsSubsetEval` algorithm using the stepwise bi-directional

search described in the aforementioned paper, with 10-fold cross-validation, was performed in order to identify the most relevant features. This produced the following output:

```
=== Run information ===
Evaluator:    weka.attributeSelection.CfsSubsetEval
Search:       weka.attributeSelection.GreedyStepwise -B -T
              -1.7976931348623157E308 -N -1
Relation:     data
Instances:    400
Attributes:   8
              match-amg match-mbz match-dgs
              country-count name-length label-count
              multilingual ambiguous
Evaluation mode: 10-fold cross-validation

=== Attribute selection 10 fold cross-validation (stratified) ===

number of folds (%)  attribute
          10(100 %)    1 match-amg
           9( 90 %)    2 match-mbz
          10(100 %)    3 match-dgs
          10(100 %)    4 country-count
          10(100 %)    5 name-length
           0(  0 %)    6 label-count
           0(  0 %)    7 multilingual
```

This means that all the features except for **label-count** and **multilingual** were judged as relevant in nearly all of the cross-validation runs of the algorithm.

4.1.2 Evaluating all feature sets

Coetzee et al. (2000, p. 1) describes a simple methodology for optimal feature selection based on the receiver operating characteristic as an *ideal approach*, but dismiss it as unfeasible for moderate-to-large feature sets. However, this approach suits the current study because only a few possible predictor variables for ambiguity have been considered.

Classifier performance can be assessed with several indicators. Cross-validation is used to estimate the rates of true positives, true negatives, false positives and false negatives that a predictive model would yield in practice. The receiver operating characteristic (ROC) is a plot that relates a classifier's

false positive rate with true positive rate. For every classifier, these values are calculated and plotted as a curve in a Cartesian plane as the classification threshold varies, where the x axis is the false positive rate and the y axis is the true positive rate.

A simplified alternative for comparing a large number of classifiers, inspired by Fawcett (2006), is to treat every model as a discrete classifier (only take into account the prediction of ambiguity, not the probability scores), and plot a single point for every classifier instead of a curve. A classifier is potentially optimal if and only if it lies on the convex hull of the set of points in the ROC space (Fawcett, 2006, p. 867): other classifiers can be discarded.

Thus, we propose the following algorithm to choose the best feature sets for both the Naive Bayes classifier and logistic regression.

```
for every subset s in the powerset of features:
    train classifier using s as the feature set,
        using 10-fold cross validation to estimate
        false positive and true positive rates
    plot (false positive rate, true positive rate)
        in the ROC space

calculate the convex hull of the set of points
    in the ROC space

analyze and compare classifiers on the convex hull
```

The classifiers with the best suited feature sets will be on the convex polygon.

Naive Bayes classifier evaluation

Figure 4.1 contains the false positive and true positive rates for every subset of the feature set, when using the Naive Bayes classifier.

In Figure 4.2 the suboptimal classifiers have been excluded and the classifiers in the convex hull have been tagged with their respective feature set for comparison. Four classifiers that are on the northwest-most region of the plot are the most interesting ones, their concrete values for false positive rate (FPR) and true positive rate (TPR) are presented below:

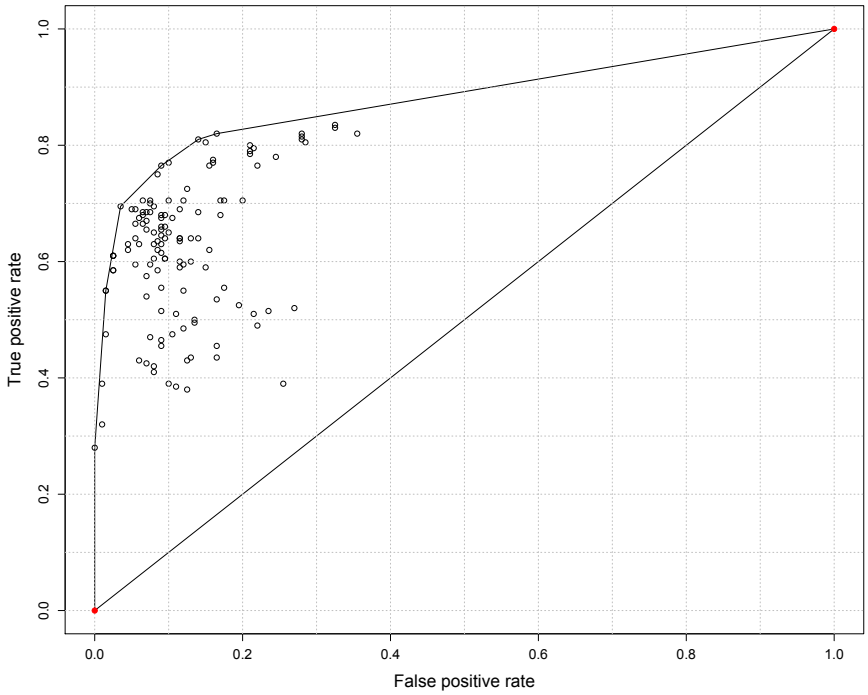

Figure 4.1: ROC space points for every possible feature set, using the Naive Bayes classifier

Subset number	Feature set	FPR	TPR
1	**match-amg** **match-mbz** **match-dgs** **name-length** **multilingual**	0.035	0.695
2	**match-amg** **match-dgs** **name-length**	0.09	0.765
3	**match-amg** **name-length** **multilingual**	0.14	0.81
4	**match-amg** **name-length**	0.165	0.82

The false positive rate of the classifier should be very low: Spotify might embed the classifier in future systems that handle automatic splitting of artists, and it is preferable to miss some real cases of ambiguous artists than

to wrongly report many ambiguous artists. Therefore, the first feature set is chosen as the most suitable for the Naive Bayes classifier.

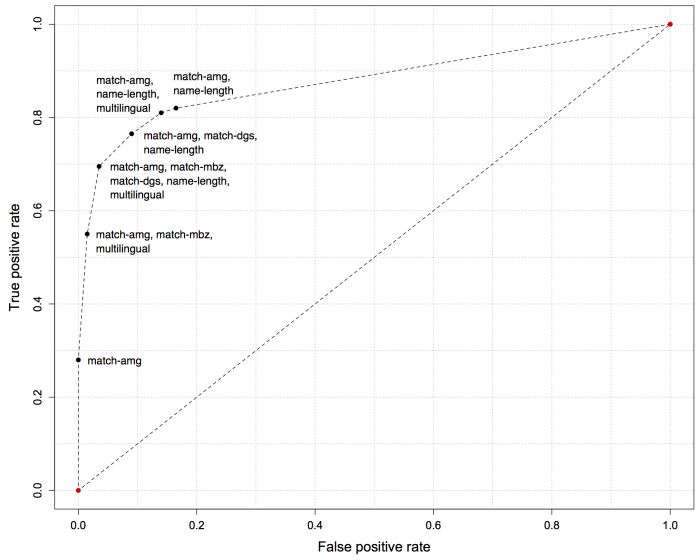

Figure 4.2: Convex hull of ROC space points and their corresponding feature set, using the Naive Bayes classifier

Logistic regression classifier evaluation

The corresponding plot of the logistic regression classifier performance for every feature set in the ROC space is shown on Figure 4.3. Figure 4.4 shows only the optimal classifiers and their feature set in the ROC space.

Subset number	Feature set	FPR	TPR
1	**match-amg** **match-mbz** **match-dgs** **country-count** **name-length** **multilingual**	0.04	0.72

This time, there is one feature set that is clearly superior to the others, achieving a false positive rate of only 0.04 and a true positive rate of 0.72. The optimal classifier by this metric includes all the features except for **label-count**.

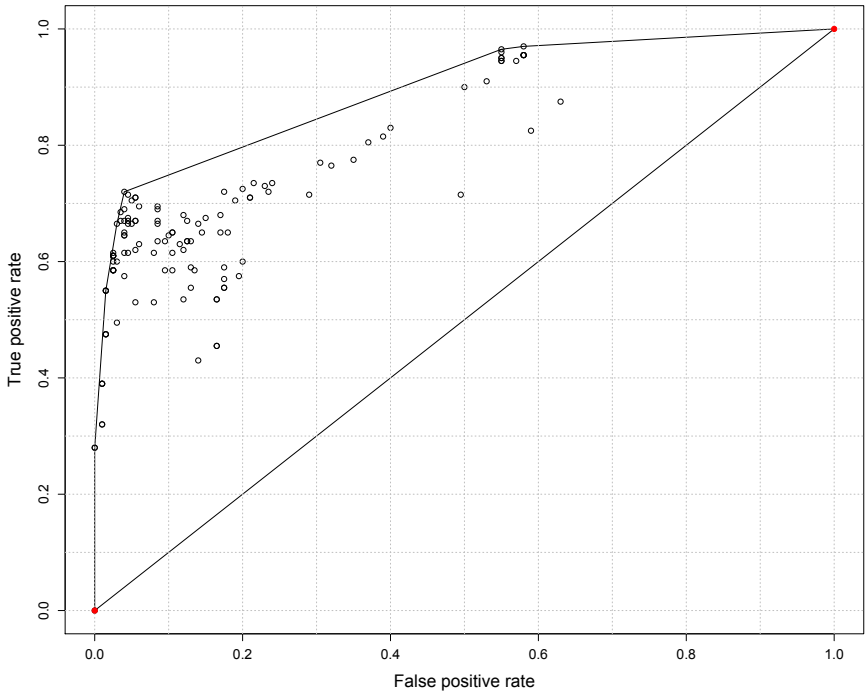

Figure 4.3: ROC space points for every possible feature set, using the logistic regression classifier

4.1.3 Best feature sets

The best performant feature set according to the cross-validation values of true positive and false positive rates was chosen for both classifiers under study, with the aid of ROC space analysis. This exhaustive analysis to choose the right feature set is preferred to the generic approach of `CfsSubsetEval`.

The following table summarizes the two classifiers, whose performance against an unseen data set will be studied in the next section.

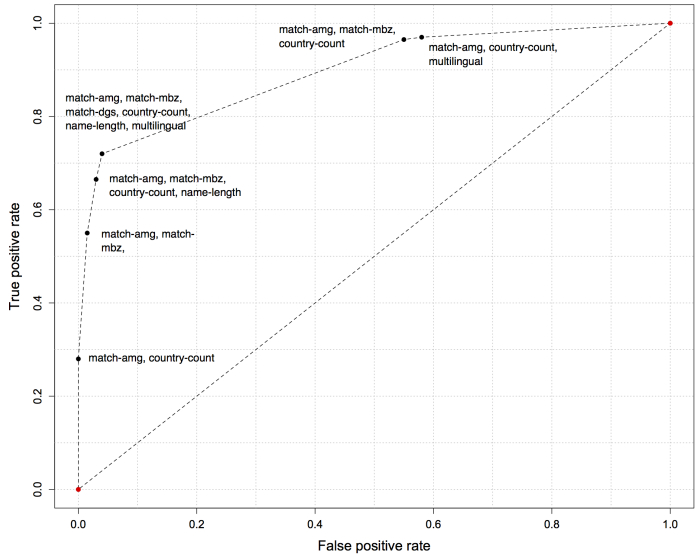

Figure 4.4: Convex hull of ROC space points and their corresponding feature set, using the logistic regression

Classifier	Feature set	TPR	FPR
Naive Bayes	**match-amg** **match-mbz** **match-dgs** **name-length** **multilingual**	0.04	0.7
Logistic regression	**match-amg** **match-mbz** **match-dgs** **country-count** **name-length** **multilingual**	0.04	0.72

4.2 Distribution of ambiguity estimates

We have mentioned that one of the main benefits of using the logistic regression approach to classification is that it is possible to obtain probabilistic scores of *how ambiguous* the learner thinks a particular example is, instead of a binary classification. Figure 4.5 shows all the artists with two or more albums in the catalogue plotted against their probability of being ambiguous,

as calculated by logistic regression under the optimal feature set. Around 4.3% of the catalogue artists were found to be ambiguous using the default threshold of 0.5. It is hard to estimate accurately how many of these total artists were correctly classified as ambiguous. The tagged examples from the training set suggest a false positive rate of 0.04, but it is *too optimistic* to estimate the performance of a classifier based on how well it predicted data from the original training set.

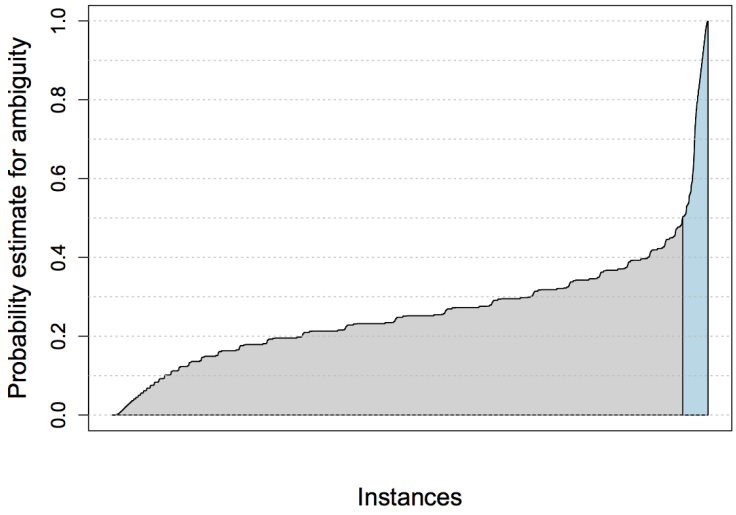

Figure 4.5: Artist ambiguity estimates provided by logistic regression. An estimate greater than 0.5 means the learner considers the artist as ambiguous.

4.3 Classifier comparison

It is mainly interesting to test the classifier performance on two kinds of data: random artists, to assess objectively their generalization capability; and the most popular artists, because their detection and correction is a priority for improving the typical user's experience.

Also, it is good practice to confirm that the machine learning approach is actually yielding some improvement over the easier approach of querying all of the external sources for multiple matches. This oracle can be represented as a decision tree (see Figure 4.6) and its performance on the test set can be evaluated alongside the other classifiers.

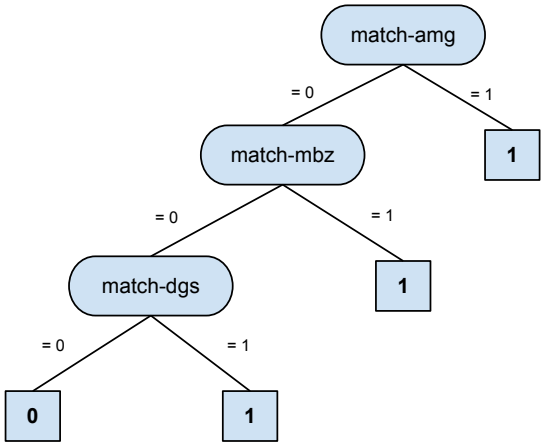

Figure 4.6: Decision tree induced by considering an artist as ambiguous if and only if there exists a multiple matching to any of the external sources.

Two datasets, `random` and `popular`, of 200 artists, previously unseen by the classifiers, were manually tagged. The baseline classifier, logistic regression and Naive Bayes classifier were asked to predict the ambiguity of the examples in the test set.

The **confusion matrix** of every classifying attempt is shown. The arrengement of the elements, according to convention, will be the following:

# true negatives	# false positives	False positives rate: fpr
# false negatives	# true positives	True positives rate: tpr

4.3.1 Dataset `random`

Baseline decision tree:

181	3	False positives rate: 0.02
11	5	True positives rate: 0.31

Naive Bayes classifier:

179	5	False positives rate: 0.03
10	6	True positives rate: 0.38

Logistic regression:

181	3	False positives rate: 0.02
9	7	True positives rate: 0.44

Since tagged artists are selected at random, the proportion of ambiguous artists is inevitably low, as it is suspected to be on the whole Spotify catalogue.

Nevertheless, it is remarkable that a basic, untrained approach has achieved such a low false positive rate: this may suggest that the external sources generally contain reliable information and that the cross-source matching scheme that has been developed is very useful for detecting ambiguous artists. However, many ambiguous artists are left undiscovered by using this method, as the true positive rate shows. There is room for improvement in this aspect.

The Naive Bayes classifier performs better on ambiguous artist discovery, but its use comes with a slight penalty on false positives. Logistic regression is the the strongest option here, maintaining a very low number of false positives while having the best true positives rate.

4.3.2 Dataset `popular`

Baseline decision tree:

179	3	False positives rate: 0.02
6	12	True positives rate: 0.67

Naive Bayes classifier:

173	9	False positives rate: 0.05
5	13	True positives rate: 0.72

Logistic regression:

134	48	False positives rate: 0.36
4	14	True positives rate: 0.78

Once again, the baseline classifier is surprisingly strong, especially on the false positives rate. As before, using the Naive Bayes classifier results in a larger number of false positives, but it is able to detect more ambiguous artists.

The greatly increased false positive rate by logistic regression after only changing the example composition from random to popular artists is inconvenient, because performing well in the detection of ambiguity among the most popular artists is a must.

Analyzing the artists where the classified erred, one notices a common pattern among popular artists that may not be so frequent in the training data, and was not accounted for: artists with long, successful careers such as Elvis Presley and Frank Sinatra have registered songs in many countries, but it is intuitively very unlikely that another artist will ever share their name, so they are not generally prone to being ambiguous. Naive Bayes did not suffer from this irregularity because its optimal feature set did not contain the **country-count** feature.

4.3.3 Adjustment to the logistic regression feature set

If this problematic feature is simply removed and the model is re-trained, we get the following figures for the different test sets:

Dataset `random`

165	19	False positives rate: 0.44
9	7	True positives rate: 0.10

Dataset `popular`

173	9	False positives rate: 0.67
6	12	True positives rate: 0.05

This time, we get less reliable results for the random data set. The classifier obtained by removing the **country-count** feature does not lie on the convex hull, so it is expected that its performance will be worse in general.

4.3.4 Results

Considering the better performance of the **original** logistic regression classifier over Naive Bayes on the random examples, a hypothesis is that, if the best feature set is chosen for classifiers trained with popular artist data instead of random data, logistic regression would also outperform Naive Bayes.

The main result is that popular artists have an inherently different distribution than random artists. The experiments should be repeated when

training with a popular artist dataset, to be able to effectively recognize ambiguous artists from this especially interesting subset of data. Due to a lack of time, a further study could not be conducted in this work.

Chapter 5

Discussion

5.1 Summary

The baseline decision oracle, querying the external sources for ambiguity, would have been in practice much easier to implement and yield the lowest rate of false positives. However, some additional ambiguous artists would remain undiscovered when using this approach. It is better to take into account other, more stable indicators of ambiguity besides data from metadata databases that depend on communities to be maintained and are constantly changing. The best way of aggregating these factors is through the use of classifiers that have been learned from actual data.

Further research is needed to come up with the best classifier for the most popular artists subset, since the data shows that logistic regression outperforms the Naive Bayes classifier when applied to a random population.

From a practical point of view, it is convenient to use the Naive Bayes classifier to predict the ambiguity of popular artists. Other feature sets of logistic regression may also be suitable, but none of them have been found to have a performance similar to Naive Bayes in true positive and false positive rates. When a random population of artists is expected, the logistic regression classifier is superior.

5.2 Future work

This work can serve as a first step towards fixing the problem of ambiguous artists, i.e., splitting the albums of the ambiguous artist and assigning them to two or more new, separate artist entities.

Deciding how to split a particularly artist is a task that Spotify would like to automate. We have started preliminary research on using different

clustering algorithms based on features similar to the ones that have been described here. So far, no particularly interesting results can be reported, but there is much potential for improvement around this idea.

One simpler automatic approach is to split all the albums of the ambiguous artist according to which external artist they match. This is viable if the system has reasonable confidence in the information provided by the external source. A problem with this idea is that the databases are continuously changing. For example, the database state may suggest a division of the artist at some point in time, and some time after, the database maintainers realize they have made a mistake and undo the changes. How can the Spotify metadata keep up with this? The goal is for it to be reliable and as organized as possible; constant changes are difficult to monitor and bring unpredictability to the state of the database.

A more conservative thought is to manually separate a small subset of the artists marked as ambiguous: this can be done in popularity order so as to maximize the benefit in user experience. After this has been done, a simple heuristic system could be put in place so that when a new product identified with an artist name that could belong to any of two or more artists arrives to Spotify, the system may be able to guess which artist is more likely to have produced the album. These heuristics could very well be based on some of the features considered in this project: the country codes of its tracks, the record labels the artists have recently signed with, the languages in the tracks' metadata and so on.

Finally, a crowdsourcing system where users were able to vote on assignment of albums to artists and report mistakes in the classification would be a very interesting follow-up to this project.

Bibliography

AI Access. Probabilities, statistics and data modeling: bias-variance trade-off. Retrieved on 2012-05-16 from `http://www.aiaccess.net/English/Glossaries/GlosMod/e_gm_bias_variance.htm`, 2012.

Michael T. Brannick. Logistic regression. Retrieved on 2012-05-21 from `http://luna.cas.usf.edu/~mbrannic/files/regression/Logistic.html`, 2007.

Frans M. Coetzee, Eric Glover, Steve Lawrence, and C. Lee Giles. Feature selection in web applications using roc inflections and power set pruning, 2000.

Discogs. Using discogs. Retrieved on 2012-06-14 from `http://www.discogs.com/help/about-discogs.html`, 2012.

Miklós Erdélyi and András A. Benczúr. Temporal analysis for web spam detection: An overview. In *Temporal Web Analytics Workshop*, pages 17–24, 2011.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. In *Journal of Machine Learning Research*, pages 1871–1874, 2008.

Tom Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27 (8):861–874, June 2006.

Thomas Gottron and Nedim Lipka. A comparison of language identification approaches on short, query-style texts. In *European Conference on Information Retrieval*, pages 611–614, 2010.

All Media Guide. Allmusic: About. Retrieved on 2012-05-15 from `http://www.allmusic.com/about`, 2012.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, pages 10–18, November 2009.

Mark A. Hall. Correlation-based feature selection for machine learning. Technical report, 1998.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data mining, inference and prediction*. Springer, second edition, 2009.

Václav Hlaváč. Classifier performance evaluation. Retrieved on 2012-05-21 from `http://cmp.felk.cvut.cz/~hlavac/TeachPresEn/31PattRecog/13ClassifierPerformance.pdf`.

David W. Hosmer and Stanley Lemeshow. *Applied Logistic Regression*. John Wiley and Sons, Inc., second edition, 2000.

last.fm. last.fm: Music manager help. Retrieved on 2012-06-14 from `http://musicmanager.last.fm/help/faq?category=Most+Common+Questions&faq=482#482`, 2012.

Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 1966.

David D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European Conference on Machine Learning*, pages 4–15, 1998.

MusicBrainz. Musicbrainz: How editing works. Retrieved on 2012-05-15 from `http://musicbrainz.org/doc/How_Editing_Works`, 2012.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, 1999.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, third edition, 2010.

Bruno Scarpa and Nicola Torelli. Selecting the training set in classification problems with rare events. In *New Developments in Classification and Data Analysis*, pages 39–46. Springer Berlin Heidelberg, 2005.

Spotify. Spotify for ipad. Retrieved on 2012-05-31 from `http://www.spotify.com/us/blog/archives/2012/05/02/spotify-for-ipad/`, 2012.

Wikipedia. International standard recording code. Retrieved on 2012-05-12 from http://en.wikipedia.org/wiki/International_Standard_Recording_Code, 2012.