Vector embeddings of harmonies in music with deep learning

Elia Anzuoni, Sinan Ayhan, Federico Dutto
École Polytechnique Fédérale de Lausanne - EPFL , Switzerland - 2020

***Abstract*—This project focuses on the exploration of the possibilities arising from the application of NLP word-embedding methods (Word2Vec) to corpora of musical sentences. Specifically, our study moved on two parallel tracks: we analysed the clusters of embedded vectors produced by Word2Vec, in order to probe its ability to learn some common musical patterns; moreover, we implemented an LSTM-based neural network aimed at the prediction of the next chord of a musical section, using the variability in the prediction accuracy to quantify the stylistic differences among various composers, and to detect idiomatic uses of some chords by some composers.**

Our cluster analysis proves able to identify some well-known tonal relationships between chords. The results from chord prediction show how Impressionist composers have a peculiar way of using I:MAJ and V:MAJ, and more generally that classical composers are more easily predicted .

I. Introduction

A. Music theory background

In many musical styles, the chord is the fundamental element of harmony: it is obtained by playing three or more notes simultaneously (harmony itself is concerned with sounds being played together, as opposed to melody, which is about sounds played in sequence). For this project, musical pieces are viewed as sequences of chords: we disregard, for example, any melodic or rhythmic features.

Chords are identified by their *root*, and their *type* (determined by the intervals separating the root from the other notes). In this project, chords can only be of four types: major, minor, augmented, or diminished.

Furthermore, pieces (or, rather, sections of pieces) are characterised by a *key*, which can be thought of as an assignment of "roles" to chords: a chord has a different musical effect in different keys. In this project, two different chords playing the same role in two different keys are considered the same: in this sense, we study the relationships between the roles, rather than between the chords themselves, but we neglect this distinction in the following. The key has the same name as the *tonic*, i.e. its main chord, the one from which musical movement proceeds and to which it is bound to eventually return. We say that a key is a major (minor) key if its tonic is a major (minor) chord: the same terminology applies to sections, which can also be major or minor, depending on their key.

In order to be able to compare sections in different keys, the root of the chords is expressed in relative notation, i.e. as the distance to the tonic: thus, an F major chord will be written as IV:MAJ if the section is in C major, but as III:MAJ if the section is in D minor. The choice of the relative notation is motivated by the fact that the role played by a chord in a piece is largely determined by its type and its position relative to the tonic: in other words, most of what is conveyed by a piece is invariant under a translation of all its notes.

In music theory, several *harmonic progressions* (relatively short sequences of chords) are well known and studied [1]. They are very common since they produce very distinctive and desirable musical effects. The classic example, in major sections, is V:MAJ $\rightarrow$ I:MAJ, the so-called *authentic cadence*: V:MAJ creates a strong feeling of suspense which is perfectly resolved by I:MAJ.

B. Machine learning background

Our project uses three fundamental ML building blocks: word embedding, clustering, and Recurrent Neural Networks (RNNs).

Word embedding is a popular technique in Natural Language Processing (NLP) which learns a mapping of words to vectors in a low-dimensional *embedding space* from a *corpus*, which is supposed to contain sufficient information on the relations between words. The mapping is such that the relative positions of the vectors reflect the relations between words; the precise meaning of this is dependent on the specific method used: we chose Word2Vec [7] for its proven suitability to model the musical language (as in [5], which purely explores the resulting embedding space, or [6] and [3], which exploit it for similar downstream tasks as ours); in Word2Vec, words often appearing in similar contexts are mapped to close points in the embedding space, according to their cosine distance.

Clustering is a well-known unsupervised learning primitive, which works by grouping together close points in a space, and is used to extract information on the points that might be contained in their coordinates. Previous work which might be enhanced by the application of clustering to harmony embeddings may be found in [3]. At first, we used K-means [4] for its simplicity, but we then abandoned it because of its restrictive assumptions (L^2-spherical clusters) which did not apply to our data; we moved to hierarchical clustering [10] to exploit its greater flexibility.

Recurrent Neural Networks are widespread tools in NLP, particularly in the field of word prediction with their Long Short-Term Memory (LSTM) [2] variant. LSTMs are particularly suited to this task because of their structure, involving a *forget gate*, which solves the *short-term memory* problem, typical of traditional RNNs. Similar work, like [6], shows how they can be employed in chord prediction.

II. Data exploration

The dataset consists of 24 csv files, one for each composer studied; each file consists of several lines, variable in number across composers, representing musical sections. The first element of a section is either the word "MAJOR" or "MINOR", followed by a semicolon, specifying whether the section is in a major or minor key (which exact key is irrelevant, because of the relative notation). After the key indication, a sequence of comma-separated chords written in relative notation, constituting the section, is provided. This has required extensive manual labelling work from experts, who had to decide the list of chords corresponding to each section of each composer. This annotation system has been introduced in [11], and is analysed in [8] and [9].

As one can observe in Figure 1, the amount of data at our disposal varies greatly across composers. For several composers (at least from W.F. Bach down), greater care is due in assuming them to be well represented by the data we have: when, later in the paper, we conclude on the intrinsic differences between composers, we implicitly restrict our claims to the data at our disposal. Note that in our data Sweelinck has only chords from a Minor key.

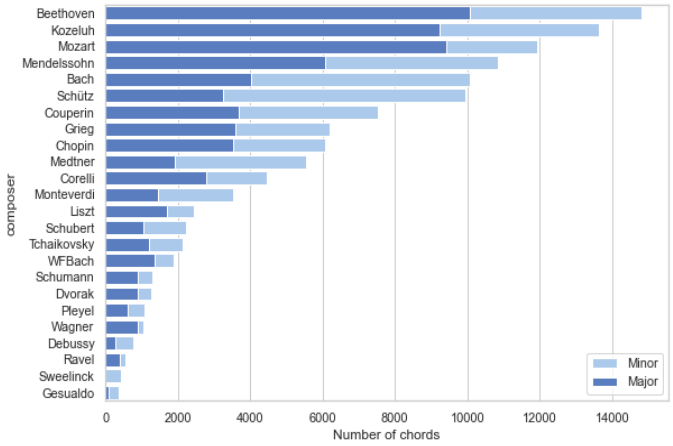

Fig. 1. Total number of (non-unique) chords used by each composer, split between major and minor sections

III. Methodology

A. Word2vec

Our first processing step, serving as a basis for the two downstream tasks of clustering and next-chord prediction, was the application of Word2Vec [7], a popular NLP word-embedding method. From a set of sentences (each a sequence of words), it learns a mapping from words to coordinates (vectors) in an embedding space. Its objective function is designed so that words frequently appearing in similar contexts (roughly, surrounded by the same words), get mapped to close points in the embedding space (with respect to the cosine distance), and vice-versa.

In our case, the "sentences" of the corpus (the training dataset) are musical sections, possibly taken from more than one composer, while the "words" are, naturally, the chords; thus we use the terms section/sentence and word/chord interchangeably. We never "mix" major and minor sections: we never include both types of sections in a single corpus, since no chord in a major section ever appears in the context of a chord in a minor section, so there is nothing additional to learn from this "joint" training. Thus, in the following, when we say "train/test on all sentences/sections of a composer" or "train/test on a composer", we implicitly mean that those sections are all in the same mode (unless otherwise specified).

1) Hyperparameters: Word2Vec has three hyperparameters to tune: `size`, `window`, and `sg`. We tune them in the downstream tasks, to optimise their performance separately.

- `size`. The dimension of the embedding space. To avoid overfitting, this clearly has to be less than the size of the vocabulary, i.e. the number of distinct words in the corpus. In our case, the vocabulary size varies considerably, between 20 and 100 per composer within either of the two modes: depending on this number, we pick a reasonable range of values for `size`, roughly between a tenth and a quarter of the size of the vocabulary.
- `window`. The "width" of the context, i.e. how many chords, to the left and to the right, constitute the context of the current chord. Drawing on musicological considerations, we chose the range [2, 4].
- `sg`. Short for Skip-Gram, selects the training algorithm: it can be either CBOW (guessing the target word from its context), or Skip-Gram (guessing the context given the target word).

An additional hyperparameter, `min_count`, allows to set a minimum absolute frequency a word must have in order to be kept in the corpus. It allows us to polish the result from a lot of spurious, irrelevant mappings of rare words. It does not influence the outcome greatly, but it reduces the size of the corpus, forcing us to reconsider our choice of the other hyperparameters (in particular, `size`).

B. Clustering

A first application of the mapping learnt by Word2Vec is clustering, which is used to detect musical patterns. As is understandable from the properties of the mapping and confirmed by our results, chords appearing in the same cluster are likely to often appear in similar contexts. This is an improvement over [3], where chord vector embeddings were only graphically grouped together, thus not allowing for a clear semantic interpretation. For this task, it was extremely hard to carry out an objective, quantitative model evaluation: therefore, the hyperparameters were chosen based on how much the outcome corresponded to music-theoretical intuitions (for example, we expect, when only training on major sections, that "I:MAJ" and "V:MAJ" end up mapped to close points, since they constitute the most basic musical pattern imaginable, as discussed in [8]).

1) K-means: K-means was our first attempt at clustering, but we quickly disregarded it because of its inherent limitations. First of all, it expects clusters to be spheres of equal radius, which cannot be assumed to hold in our case. More fundamentally, the very fact that K-means works with the L^2 distance clashes with the design of Word2Vec: the embedding space comes equipped with the *cosine distance*, not the L^2 distance. Finally, this algorithm heavily depends on the value of K (the number of clusters), which could only be chosen arbitrarily.

2) Hierarchical clustering: Hierarchical clustering is an alternative method which elegantly solves the problems cited above. In our setting, it works by recursively merging the pair of clusters C_i and C_j (starting from singletons) that are the closest to each other. The distance between two clusters is defined as:

$$L_{ij} = \max_{a \in C_i,\, b \in C_j} d(a,b) \tag{1}$$

where d is a provided metric (the cosine distance is allowed). The recursion stops when the minimum value of L_{ij} is above a given `distance_threshold`.

A first obvious advantage is that this algorithm can work with the cosine distance which, as was mentioned, is preferable to detect similarities in a Word2Vec embedding space. Moreover, it is able to capture clusters of any shape. One might argue, however, that a choice of `distance_threshold` is just as arbitrary as one of K in K-means: this can be obviated by setting `distance_threshold` = 0, and then plotting the dendrogram. The dendrogram (Figure 3) is a depiction of the nested clusters produced by this method: it clearly shows all the mergers $C_i - C_j$ that happened, and the L_{ij} associated to them; one could then choose a reasonable value of `distance_threshold` from the dendrogram (i.e. from the data themselves), for example by setting it to the distance between two chords we would expect to fall in the same cluster (e.g. I:MAJ and V:MAJ).

C. Chord prediction

Another use of the mapping provided by Word2Vec is the LSTM-based next-chord prediction task. LSTMs are an improvement over the classic RNN design that solve its *short-term memory* problem (caused by the well-known *vanishing gradient* problem): this allows them to effectively track long-term dependencies in sequential data. They are commonly used in NLP to predict the next word in a sentence.

We implemented an LSTM-based neural network for next-chord prediction, which trains on a *training corpus* (usually all sentences from a set of *training composers*) and is tested on a *test corpus* (usually all sentences from a single *test composer*). The metric used is the simple accuracy, i.e. the fraction of correctly-predicted chord occurrences, possibly grouped by chord. Model selection (on both Word2Vec's and the NN's hyperparameters) is carried out to optimise the overall test accuracies. We use the overall accuracy results for a single test composer to see how "predictable" he is, from what we learnt from the training composers; we use the same results, detailed by chords, to investigate which chords are easier to predict and which are used more idiomatically. Another idea, which we ultimately dropped, was to investigate the entropy of the context (i.e. the entropy of the distribution of the possible contexts for a chord): we would have expected it to be inversely correlated with the prediction accuracy for that chord, but we actually did not find any evident relation. We leave a full investigation of this to future work.

We implemented this predictor network in two slightly different variants, which we present in order of achieved test accuracy. Both architectures share the same simple underlying structure: a first LSTM layer takes as input the embedding coordinates of one or more chords (called *predicting chords*) surrounding the target chord, and outputs a vector of dimension `hidden_dim` (a hyperparameter); a linear layer maps this vector to a vector of `n_vocab` logits (where `n_vocab` is the number of distinct chords); finally, a softmax layer maps the logits to normalised positive numbers, indicating the estimated distribution of the next chord. The LSTM layer keeps track of what it has previously seen in the sentence through a `state`: we use it in architecture A to exploit the memory of the LSTM, and effectively predict based on the entire past; to the contrary, we do not use it in architectures B. More formally, the training and the testing datasets for this network consist of points (x_i, y_i): x_i is the juxtaposition of the embedding coordinates of the predicting chords for the i-th target chord; y_i is the actual i-th target chord, given as its class index in the vocabulary of chords: this way, we aim at accurately predicting the target chord (alternatively, we could have given y_i as its embedding coordinates, thus aiming at predicting a point in the embedding space close to y_i).

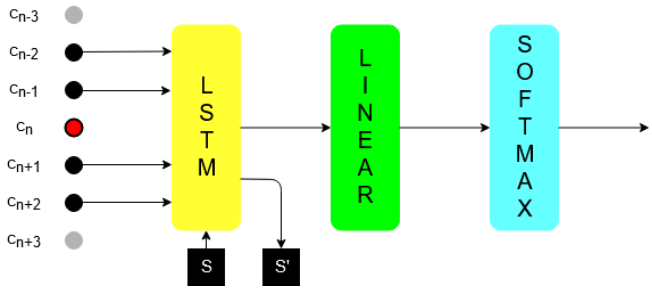

Fig. 2. Diagram for the predictor network

1) Architecture A: In this architecture, we predict a chord based on the entire past, i.e. on all chords that precede it in the section. The only predicting chord is c_{n-1}; however, we also keep the output state S', and feed it to the LSTM at the next iteration (this way, we exploit the LSTM's memory capacity).

2) Architecture B: In this architecture, we predict a chord based exclusively on its surroundings, i.e. the `window_size` chords before and the `window_size` chords after (these are the predicting chords), where `window_size` is a hyperparameter independent of the `window` parameter of Word2Vec. The output state S' is disregarded, since we want to predict a chord given only its context. This architecture achieves a gain in test accuracy of 5-7%, uniformly over all composers, over architecture A. This suggests that future chords are also very useful in predicting the current chord

IV. Results

A. Clustering

Here, we show the results we obtained by applying hierarchical clustering on the corpus containing every composer's **major** sections: we leave the results for minor sections out of this report for space, but they are available with the code[1], and lead to similar conclusions.

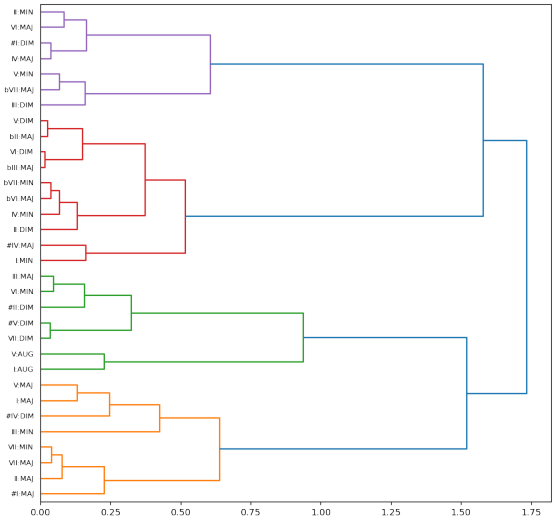

Fig. 3. Clusters dendrogram for major sections

As Word2Vec parameters, we set `min_count`=50 (since the most common chords had absolute frequencies of hundreds, if not thousands): this led to a vocabulary size of 32: the `size` of the embedding space was then chosen to be 5 (alternatives were substantially equivalent); we set `window`=2, again because other values led to similar results; we chose `sg`=1 because it led to more interpretable results than `sg`=0.

The dendrogram in Figure 3 shows that a sensible value for `distance_threshold` might be 0.15, i.e. the cosine distance between I:MAJ and V:MAJ. Using this distance as a threshold leads to a very interesting notion: the resulting clusters group chords that

[1]Code : https://github.com/CS-433/cs-433-project-2-cadrega

are at least as close to each other as the tonic (I:MAJ) is to the *dominant* (V:MAJ).

The resulting clusters mainly pick up on two kinds of tonal relationships: tonic-dominant (the basis for the already-mentioned authentic cadence), and *relative* (two chords are each other's relative if they are the tonics of two keys, a major and a minor, whose diatonic scales are just a rotation of each other). They are:

- **(I:MAJ, V:MAJ)**. The most obvious tonic-dominant pair.
- **(VI:MIN, III:MAJ)**. III:MAJ is V:MAJ in the key of VI:MIN (which is the relative of I:MAJ): this is a "translated" tonic-dominant relationship.
- **(VI:MAJ, II:MIN)**. VI:MAJ is V:MAJ in the key of II:MIN: this is another translated tonic-dominant relationship.
- **(VII:MAJ, VII:MIN, II:MAJ)**. VII:MAJ and VII:MIN are bound by a simple parallel relation, while VII:MIN and II:MAJ are each other's relative chord.
- **(bVII:MAJ, V:MIN)**. They are each other's relative chord.
- **(VII:DIM, #V:DIM)**. These are bound by a different tonal relationship. They share two notes; their "union" is the often-used VII:DIM7, a *fully diminished* chord which nicely resolves onto I:MAJ.

Although it is interesting that our method can isolate *some* such clusters, it is obscure to us why it does not find them *all*, or why it finds these particular ones (e.g., II:MIN and IV:MAJ are also each other's relative chord, but they do not appear in a cluster).

B. Chord prediction

Here, we summarise the results we obtained in chord prediction. An identical analysis as for clustering led us to choosing the same Word2Vec hyperparameters. For the predictor network, we used architecture B with a value of 2 for `window_size`(A value of 1 led to worse performance, but values higher than 2 did not improve it significantly).

As can be seen from Figure 4, predictions generally work best for composers of the classical period, and are worse for authors that are far in time from the classical period. This result is partially explained by having more data for composers of the classical period. However, this is not the only reason, as Baroque composers such as Couperin, or Romantic ones such as Tchaikovsky have a relatively high accuracy as well. Further, even though we have more data from the Romantic period than from the Baroque period, Baroque composers have in general higher accuracy than Romantic composers, indicating that either the Baroque style is close to the Classical style, for which we have more data, or that Romantic authors have somehow a 'higher variance'. This hypothesis is also suggested by musicology.

An analysis of the detailed results, i.e. of the accuracy obtained for each composer for each of the most common chords (figures available in the code), gives even more insight about the idioms common to a specific composer or period. The strongest result, in a major context, is the very low prediction accuracy we scored for I:MAJ and V:MAJ (the absolute easiest chords to predict) when testing on Ravel and Debussy: indeed, they are two Impressionist composers, who are generally known in music theory for their "distinct" harmonies, which rarely (if ever) use authentic cadences. Moreover, we find IV:MAJ and II:MIN to be two "polarising" chords: for most composers, we either predict them very well or very badly, compared to the average. In particular, IV:MAJ is only well predictable for baroque authors, while others (with the exception of Beethoven, Chopin, and Dvorak) use it in a more peculiar way. II:MIN, on the other hand, only becomes hard to predict from the late romantic period. This latter result, albeit neat and striking, is not as easily interpretable as the previous one. In

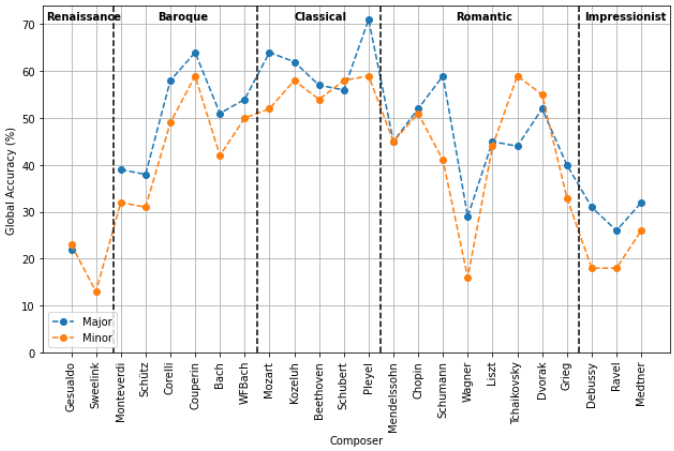

Fig. 4. Global accuracies for all composers, for major and minor sections. Composers are ordered by year of death (from oldest at the left to more recent at the right)

minor sections, a low accuracy on I:MIN (the most common chord together with V:MAJ) for Renaissance authors (Gesualdo, Sweelinck, Monteverdi, Schütz) and for Impressionists, again signals that this chord has played diverse roles across the centuries. We achieve a relatively low accuracy on many of the most common minor chords for romantic and impressionist composers, with the exception of Tchaikovsky. This indicates that he is closer to classical composers in his works in minor contexts (indeed, his only piece in the dataset is Seasons, an extremely traditional piece overall).

V. Conclusions and Future Work

In this paper, we investigated the relations between the chords in a key, as well as the possibility of predicting the chords used by a composer in a section. Word2Vec was our first processing step, which provided useful grounds to base our subsequent analyses on. When applied to the output vectors of Word2Vec, clustering could capture some well-known tonal relationships between chords: 5 out of the 6 clusters we identified could be semantically interpreted as grouping either relative chords or (translated) tonic-dominant pairs of chords. On the other hand, LSTM-based chord prediction yielded fairly high accuracy results in general (at least 50% for most composers), but it also allowed us to use their high variability across chords and composers to draw some conclusions which are supported by music theory. Globally, we found that Classical and Baroque composers use chords in a similar way, while Impressionists and Renaissance composers have a more distinctive style. The Romantic style seems to be complex, as there is a high variance in how Romantic composers use chords.

Future work might include a more refined use of clustering, for instance by applying it to a Word2Vec model trained only on a single composer—or on a group of composers which are known to be relatively similar to each other—in order to detect some special tonal relationship unique to that set of composers. Alternatively, chord prediction could be employed to investigate how rigidly a given composer belongs to a given artistic era: by restricting the training corpus to the other composers in the same era, we would prevent the model from learning totally unrelated idioms, thus achieving a higher accuracy on the test composer (to an extent depending on how similar he actually is to the others in that era).

References

[1] David E Cohen. ""The Imperfect Seeks Its Perfection": Harmonic Progression, Directed Motion, and Aristotelian Physics". In: *Music Theory Spectrum* 23.2 (2001), pp. 139–169.

[2] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM". In: (1999).

[3] André Ghattas, Beatriz Borges, and Irene Petlacalco Barrios. "Enchordings-Harmony Embeddings". In: (). ML4Science report from 2019, not an academic paper.

[4] J. A. Hartigan and M. A. Wong. "Algorithm AS 136: A K-Means Clustering Algorithm". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108. ISSN: 00359254, 14679876. URL: http://www.jstor.org/stable/2346830.

[5] Dorien Herremans and Ching-Hua Chuan. "Modeling Musical Context with Word2vec". In: *CoRR* abs/1706.09088 (2017). arXiv: 1706.09088. URL: http://arxiv.org/abs/1706.09088.

[6] Kristoffer Landsnes et al. "A model comparison for chord prediction on the Annotated Beethoven Corpus". In: *Proceedings of the 16th Sound & Music Computing Conference. Málaga, Spain*. 2019.

[7] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*. Ed. by C. J. C. Burges et al. Vol. 26. Curran Associates, Inc., 2013, pp. 3111–3119. URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[8] Fabian C. Moss et al. "Statistical characteristics of tonal harmony: A corpus study of Beethoven's string quartets". In: *PLOS ONE* 14.6 (June 2019), pp. 1–16. DOI: 10.1371/journal.pone.0217242. URL: https://doi.org/10.1371/journal.pone.0217242.

[9] Fabian Claude Moss. "Transitions of Tonality: A Model-Based Corpus Study". In: (2019), p. 335. DOI: 10.5075/epfl-thesis-9808. URL: http://infoscience.epfl.ch/record/273178.

[10] Fionn Murtagh and Pedro Contreras. "Algorithms for hierarchical clustering: an overview". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.1 (2012), pp. 86–97.

[11] Markus Neuwirth et al. "The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets". In: *Frontiers in Digital Humanities* 5 (2018), p. 16. ISSN: 2297-2668. DOI: 10.3389/fdigh.2018.00016. URL: https://www.frontiersin.org/article/10.3389/fdigh.2018.00016.