Master research Internship

Master Thesis

Knowledge Discovery in Multimedia Content by Diversion of Supervised Machine Learning Techniques

Domain : Machine Learning - Multimedia

Author:
Amélie Royer

Supervisors:
Vincent Claveau
Guillaume Gravier
LinkMedia - Inria/Irisa

Abstract

Knowledge discovery is the problem of extracting new information from large databases, such as recurrent patterns or structural cues. In this domain, the sub-task of *cluster analysis* deals with partitioning a given data space such that two samples in the same cluster are similar, while those in different ones are not. Clustering algorithms exploit an input similarity measure on the samples, which should be fine-tuned with the data format and the application at hand. However, manually defining a suitable similarity measure is a difficult task in case of limited prior knowledge or complex data structures for example.

The purpose of this internship is to investigate an approach for automatically building such a measure by taking advantage of the discriminative abilities of state-of-the-art classification techniques. While classification systems usually require a set of samples annotated with their ground-truth classes, recent works have shown it is possible to exploit classifiers trained on an artificial annotation of the data in order to induce a similarity measure. In this report, after introducing related scientific background, we propose a unified framework, "SIC" (Similarity by Iterative Classifications), which explores the idea of diverting supervised learning for automatic similarity inference. We study several of its theoretical and practical aspects. We also implement and evaluate SIC on three tasks of knowledge discovery on multimedia content. Results show that in most situations the proposed approach indeed benefits from the underlying classifier's properties and outperforms usual similarity measures for clustering applications.

Keywords: unsupervised machine learning, knowledge discovery, multimedia, similarity learning, clustering, data mining

Contents

Introduction

In the domain of data mining, the task of knowledge discovery deals with inferring new and previously unknown information from a dataset. This covers a wide range of sub-problems such as, for instance, extracting frequent patterns in a database, which often convey meaningful structural information (e.g., finding frequent items and items associations in customers' purchases), or fitting a model to a dataset in the form of a function or probability distribution (e.g., regression, text modeling). In this internship, we focus more specifically on discovering structural similarities between samples of a given dataset in an unsupervised manner. In particular, we apply and evaluate the proposed approach in the framework of *cluster analysis* (or *clustering*), which is the problem of grouping data samples such that two points in the same cluster are similar according to an input distance measure (or equivalently, a similarity measure), and conversely, two samples in different clusters are as dissimilar as possible. As it turns out, many knowledge discovery tasks can be formulated as such. For example, finding frequent patterns in a dataset amounts to grouping the samples according to their similarity so that each cluster represents the "equivalence class" of a potential pattern.

Building a good clustering raises two main issues. The first one is selecting an adequate underlying similarity measure between data samples, which is critical for obtaining meaningful clusters. However, this choice is highly dependent on the features used to describe the input data, and on the characteristics we are interested in the cluster analysis to bring into light. The second issue is the clustering process itself, which, taking these distance constraints as input, constructs a partitioning of the dataset. The performance of the process is strongly influenced by the data distribution and application at hand. At the same time, due to the rapid advances in data acquisition in recent years, databases are often very large, thus requiring a further need for computational efficiency, both for distance computations and clustering. In this report, we tackle the first problem, which is to define an input similarity measure between samples when limited prior knowledge is available. As we will discuss in more details later, it sometimes proves difficult to find a suitable input similarity measure for clustering tasks. When usual similarity measures are not suited, users have to manually define an appropriate measure for a particular task, which is challenging in case of complex data structures or lack of prior knowledge for example. To alleviate this construction problem, we investigate a method for automatically building an adequate similarity measure in an unsupervised framework.

The proposed approach relies on the following thought: Intuitively, clustering is very similar to classification, which is the task of associating a data point x with a label y belonging to a set of classes, $\mathcal{Y}$, defined beforehand. However they differ in essence. First, for classification, the class space is known, as well as the number of clusters and what they represent. Secondly, it is a *supervised* task, which means we have access to an annotated database (i.e., a subset of samples with their ground truth class). Thus the usual workflow is to identify characteristics of the classes using this training database in order to later correctly classify new unseen inputs. Clustering works the other way round as it uses (non-annotated) input samples to form clusters, and only then the result is used to determine specific characteristics of the data samples that may have led to this particular clustering. Clustering is a fundamentally *unsupervised* task, in the sense that it does not require any direct prior information about the clusters. The core idea of the internship is to take advantage of the discriminative power of supervised classification techniques, and divert them for the purpose of discovering structural similarities between samples. In recent years, this process has been applied in various domains [**Liu et al., 2000**] such as medical analysis [**Shi and Horvath, 2006**] and knowledge discovery in multimedia content [**Claveau and Ncibi, 2014**] [**Claveau and Gros, 2014**]. More specifically, all these articles propose to infer a similarity measure by diverting usual supervised learning algorithms, under the assumption that two objects often classified together share some resemblance. For simplicity, the approach proposed in this report, which is derived from the previous idea, will be referred to as *SIC* (*S*imilarity by *I*terative *C*lassifications). To assess the quality of the method, we use the obtained similarity measure as input to several knowledge discovery tasks and compare the results to the baseline. In fact, SIC provides several advantages for this purpose. First, SIC abstracts the problem of explicitly defining the similarity measure. Furthermore, SIC infers similarity information directly from the data at hand, hence chances are it is better fitted than a measure defined without any prior insights. On the other hand, a major difficulty is the lack of prior information. For instance, data samples in clustering tasks are unlabeled, thus an adequate *synthetic* supervised framework must be built before applying supervised classification techniques. Besides, in this unsupervised setting we do not know what the ideal *similarity* should be like, which makes it difficult to tune the algorithm (e.g., learning parameters, convergence criterion).

In this internship report, we first give a definition of the clustering task in general, and explain how the current issues motivate the proposed approach. We also review related work more specific to the process of diverting supervised learning techniques for the purpose of cluster analysis. The second section describes the main contribution of the internship, which is a theoretical and practical study of SIC. In particular, it contains a probabilistic model of the obtained similarity function. In the third section, we report experiments using our implementation of SIC on two knowledge discovery tasks on multimedia content, namely, *unsupervised labeling of named entities* in text documents and *motif discovery* in audio content. We choose to use two very different datasets (in terms of data types, learning algorithms, scales. . .) in order to explore the properties of SIC in different settings. These two frameworks are typical situations where it is difficult to define an adequate similarity measure on the data samples because of the lack of prior knowledge. Lastly, we report results in a framework other than clustering, for the problem of nearest neighbors retrieval. The task is to retrieve words that are semantically related in an unsupervised manner. In the last section, we study some additional properties of SIC in practice, in particular its convergence speed.

1 State-of-the-art and motivation

In this section, we introduce the scientific background at the basis of this internship. The first subsection introduces the task of cluster analysis. It contains a description and comparison of several state-of-the-art clustering techniques, as well as definitions for a few classic clustering evaluation measures. We also motivate the goal of the internship with considerations about the influence of the input distance choice on resulting clusters. The second subsection presents the more specific idea of diverting supervised learning techniques for applications to clustering. We review three applicative frameworks and the corresponding related work.

1.1 State-of-the-art of cluster analysis

1.1.1 Problem formulation

Definition. Let the dataset, $\mathcal{X}$, be a set of D points, $\mathcal{X} = \{x_1 \dots x_D\}$, lying in a space $\mathcal{E}$, the data description domain. A clustering of $\mathcal{X}$ is a set of groups of samples $\mathcal{C} = \{C_1 \dots C_k\}$ that cover the whole dataset; i.e., $\forall i \in [\![1; k]\!],\ C_i \neq \varnothing,\ C_i \subset \mathcal{X}$ and $\mathcal{X} = \bigcup_i C_i$. The sets C_i are called *clusters*, and their number, k, is not necessarily known beforehand. Note that the term *clustering* refers to the problem of finding the clusters as well as the resulting partition of the data space.

In the most usual definition, the clusters are required to be disjoint ($\forall i \neq j,\ C_i \cap C_j = \varnothing$), thus each data point belongs to only one cluster. This definition of the problem is sometimes called *hard* or *crisp* clustering. However, this requirement is softened in some frameworks. For instance, *hierarchical* clustering methods form a hierarchy of clusters, which therefore has a tree-like nested structure. Intuitively, each level of the hierarchy yields a different regular hard clustering, and any cluster of the n-th level is included in some cluster of the $(n + 1)$-th. Similarly, in *fuzzy* clustering methods, a degree of membership to each cluster is computed for every data sample, instead of directly returning the cluster the point belongs to. This information is especially useful when a point's membership to a cluster is not obvious, for example, for points lying at the border of a cluster. We illustrate how these variants differ in terms of output in Figure 1. In this section, we stick to the usual framework as we introduce the state-of-the-art for hard clustering. However, the similarity construction process we present in this report, is not subject to such restriction and can be given as input to any type of clustering.

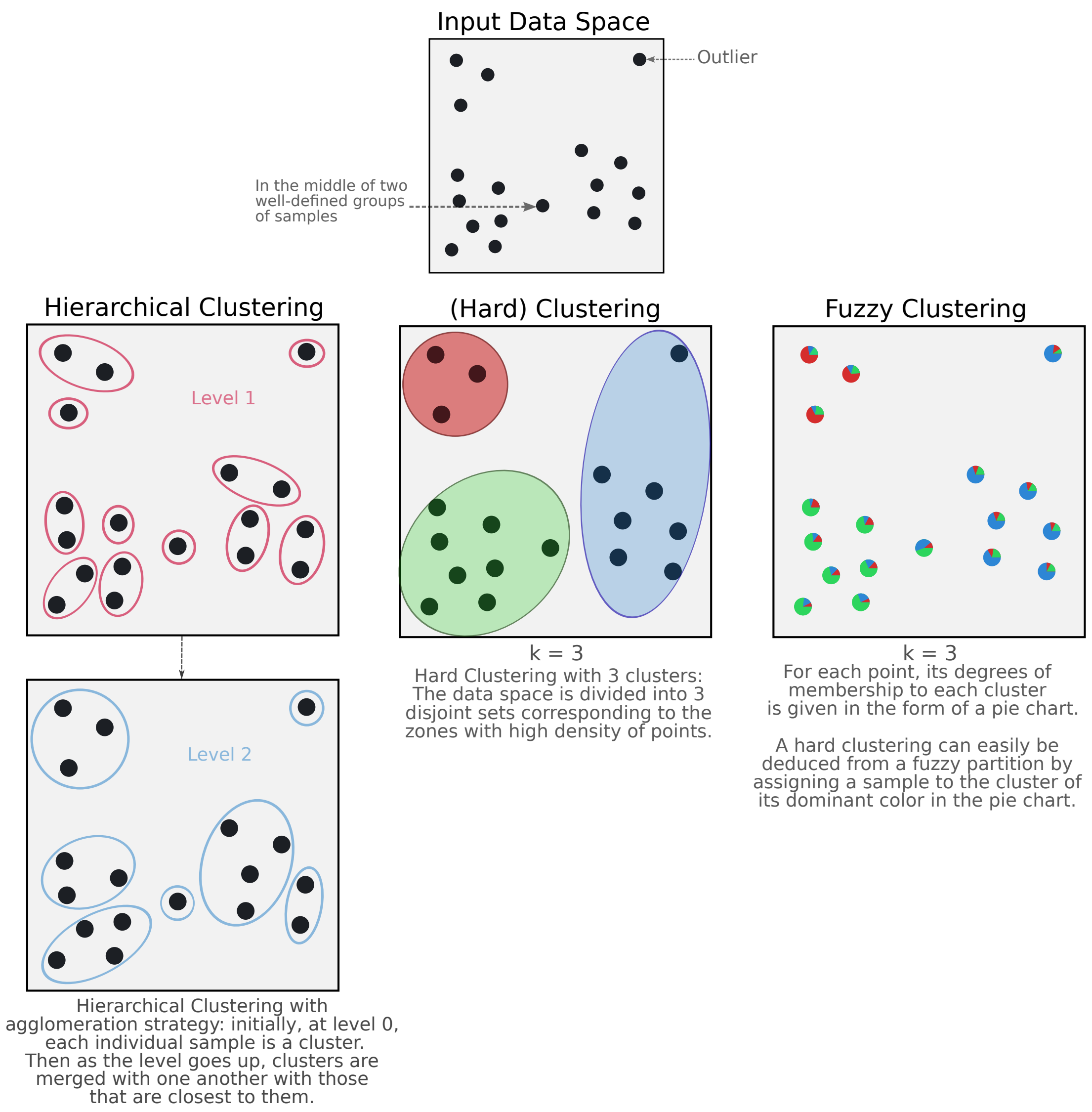

Input Data Space
Outlier
In the middle of two well-defined groups of samples
Hierarchical Clustering
Level 1
Level 2
Hierarchical Clustering with agglomeration strategy: initially, at level 0, each individual sample is a cluster. Then as the level goes up, clusters are merged with one another with those that are closest to them.
(Hard) Clustering
k = 3
Hard Clustering with 3 clusters: The data space is divided into 3 disjoint sets corresponding to the zones with high density of points.
Fuzzy Clustering
k = 3
For each point, its degrees of membership to each cluster is given in the form of a pie chart.
A hard clustering can easily be deduced from a fuzzy partition by assigning a sample to the cluster of its dominant color in the pie chart.

Figure 1: Hard Clustering and its variants illustrated. On the left is a hierarchical clustering built with agglomeration strategy; The middle column is an example of hard clustering with 3 clusters; Finally, the right column presents the corresponding fuzzy partition, with 3 clusters as well.

Goals and issues. Intuitively, a good clustering should minimize the *intra-class* distance (i.e., points in the same cluster should be similar) and maximize the *inter-class* distance (i.e., any two points in different clusters share little resemblance). This notion of similarity is defined by an input distance $d : \mathcal{E} \times \mathcal{E} \to \mathbb{R}$. The first issue is the process of building the clusters, which mostly depends on the data distribution and the expected number of clusters. For example, the dataset may contain outliers, corresponding to regions with few observations. In these cases, the algorithm chooses between adding the point to another cluster (which may lead to unbalanced or distorted clusters since outliers are far away from all other samples), or creating a new separate cluster (which may lead to unnecessary partitions and a surplus of clusters). Secondly, clustering techniques usually do not rely on any (or few) prior information on the clusters. This is an unsupervised framework and this lack of information is one of the challenges to overcome. However, indirect knowledge on the clusters can be introduced trough the parameters of the algorithm. This raises questions such as how to choose (or dynamically determine) the number of clusters, or whether to search for specific-shaped clusters (e.g., ellipses, bounding boxes) or not.

The second main issue is the choice of the input distance, d, which is strongly linked with how the data is described in practice, and should reflect the characteristics we want the final clustering to exhibit. The input distance strongly impacts the resulting clusters and how meaningful they are for the considered application. For instance, let us consider a clustering problem on audio signals described as real-valued vectors. If we are interested in grouping together exact repetitions of the same pattern, then a simple Euclidean distance is befitting. However, in most applications, audio signals are pre-treated and possibly re-sampled, hence measures taking temporal distortions into account, such as the *D*ynamic *T*ime *W*arping *(DTW)* distance, are better suited. Both issues are addressed in further details in the remaining of the section.

1.1.2 Building a partition

A) Taxonomy of cluster analysis algorithms. Cluster analysis is a popular problem that has been studied in many domains, which led to a wide variety of clustering algorithms, each having specific constraints and advantages. Based on a 2005 survey [**Rui and Wunsch II, 2005**], we describe several usual clustering techniques in this subsection.

Error-based optimization. Cluster analysis can be formulated as an optimization problem, where the clusters are refined until some error function reaches a minimum. A well-known example is the *k-means* algorithm, which aims at finding the clustering that minimizes the distances from any sample to the centroid (mean) of its cluster. Note that this algorithm can also be formulated as a steepest descent, and since the function to minimize here is not necessarily convex, the convergence towards a global minimum is not guaranteed [**Bottou and Bengio, 1995**]. Some other clustering algorithms yield a global minimum solution, however they are computationally expensive because they explore more of the optimization space. The core idea of k-means is to iteratively build clusters around k centroids until the whole partition is stable (i.e., no or few changes between consecutive iterations). An example of the workflow of the algorithm is presented in Figure 2.

The main advantage of the k-means algorithm is its simplicity. However, it requires the number of clusters, k, which may be unknown, as input. Furthermore, the initialization of the centroids influences the result and should be carefully tuned. Another short-coming is that the resulting clusters have a constrained shape and tend to be of similar size; In fact, they can be seen as Voronoï cells centered on their centroid. Hence, k-means are best suited for convex clusters. Finally, the input distance, d, has to be defined on the whole domain, $\mathcal{E}$, in order to compute distances between the data samples and newly-computed centroids. Some techniques only need d to be defined

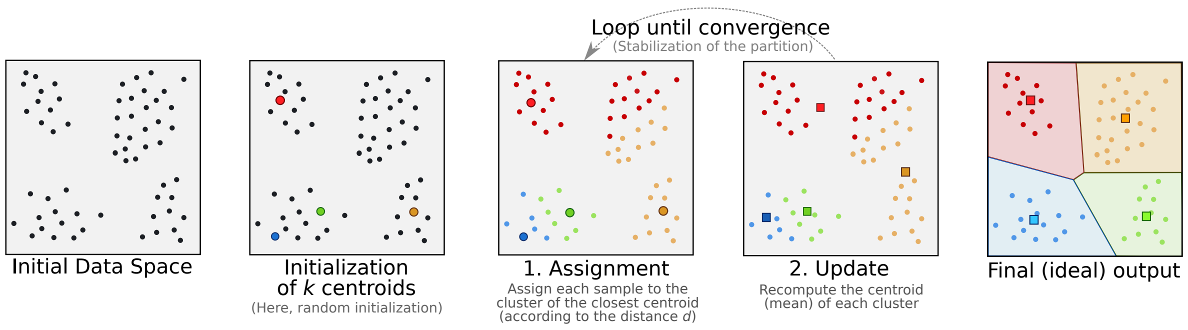

Loop until convergence
(Stabilization of the partition)
Initial Data Space
Initialization of k centroids
(Here, random initialization)
1. Assignment
Assign each sample to the cluster of the closest centroid (according to the distance d)
2. Update
Recompute the centroid (mean) of each cluster
Final (ideal) output

Figure 2: Workflow of the k-means algorithm for $k = 4$; The input parameters are the data samples, the distance d, and the number of clusters k. The algorithm outputs a hard clustering of k clusters.

on the dataset $\mathcal{X}$: This is the case for the *k-medoids* algorithm (or *PAM* clustering: *P*artition *A*round *M*edoids). It is a variant of k-means where instead of choosing a centroid as the cluster representative, a medoid (the most centrally located point in the cluster) is used. These are conceptually equivalent, with the difference that a medoid always belongs to the dataset $\mathcal{X}$.

Density-based clustering. Density-based clustering methods build clusters as regions of the data space containing a high density of points, separated by empty or low-density regions. A classical example is the *DBSCAN* algorithm (*D*ensity-*B*ased *S*patial *C*lustering of *A*pplications with *N*oise). It takes as input a distance ε (size of the considered neighborhoods) and a minimum support m (which controls the clusters' density). The idea of the algorithm is to iteratively expand clusters by browsing each sample's ε-neighborhood (i.e., all points lying at a distance less than ε from the considered sample). Let x be a sample visited during the expansion of a cluster C. If x's neighborhood contains more than m points, it is considered as a dense region and x is added to the cluster C (if not already assigned to another cluster). On the contrary, if the neighborhood contains less than m points, then x belongs to a sparse region and is marked as outlier.

This algorithm has several advantages over the k-means. In fact, DBSCAN does not require the number of clusters as input, and is able to manage and detect outliers. Furthermore, no computations of new distances on the data domain is required, and finally, it is suited for arbitrary-shaped clusters. However, DBSCAN requires information about the size of neighborhoods to explore, ε, as well as the clusters' average density (support parameter m). Both parameters are fixed inputs, which leads the algorithm to search for clusters with similar density.

Graph clustering. Graph clustering algorithms identify clusters as connected components in a graph, such that the total edges' weight is low between two different components and high inside a component. To fit the framework of graph clustering, the dataset can be seen as a graph whose vertices are the data samples, and whose weighted edges represent the distance function d (possibly with a threshold to dismiss edges between very dissimilar points).

Graph clustering algorithms are usually based on random walks on the graph, as for example the *M*arkov *C*luster *A*lgorithm (*MCL*) [van Dongen, 2000]. This algorithm relies on the assumption that if a dense cluster is visited during a random walk on the graph, then the walk will likely stay in the same region until many of its vertices have been visited. The clusters are built by iteratively updating the transitions weights of the walk, forcing the flow to concentrate in high density regions. Intuitively, this family of algorithms is similar to density-based clustering (focus on high density regions, random walk on the samples…) and has similar advantages over the k-means. The main difference is that MCL does not require a minimum density threshold as input parameter, but therefore can not use such a threshold to mark outliers, as it was the case with DBSCAN.

Spectral clustering. From a general point of view, spectral clustering refers to the family of clustering algorithms that transform the original dataset in order to obtain a new space where the clustering problem is easier, and then, apply simpler clustering techniques (such as k-means) on this new space. A classic example is to use the reduced eigenspace of the similarity matrix instead of the original dataset. The intuition is the same as dimension reduction methods that work on the eigenspace rather than the original space in order to exhibit the most important directions of the original dataset (e.g., principal component analysis). Finally, spectral clustering algorithms can be reformulated as graph clustering algorithms, and share the same advantages.

B) About the distance choice. A common point of those algorithms is that they all require an explicit input distance or similarity information. As already mentioned, this measure is an essential factor, as its discriminative power will strongly influence the resulting clusters. Note that instead of true metric distances defined on the whole data domain, $\mathcal{E} \times \mathcal{E}$, some algorithms only require a similarity measure defined on the dataset $\mathcal{X} \times \mathcal{X}$ only (e.g., k-medoids). As a matter of fact, it is always possible to extend such restricted distance to the whole domain, for instance by building a Mahalonobis distance respecting the restricted distance constraints using metric learning [**Kulis, 2013**]. Another solution is to project the data samples on $\mathbb{R}^d$ with respect to the similarity constraints. This can be done using the *MDS* (*M*ulti *D*imensional *S*caling) algorithm for example [**Cox and Cox, 2000**].

In any case, the choice of the input similarity strongly depends on the feature representation of the data samples. When the data domain is a metric space, a natural choice is to rely on its underlying distance. For example, if data samples are real-valued vectors of numerical attributes, the Euclidean distance, or more generally Minkowski distances, are commonly used. There are many other possible choices, and each of them leads to particular geometrical properties on the clusters. For instance, the k-means algorithm is often coupled with the Euclidean distance, which tends to form hyper-spherical clusters. For categorical data, i.e., attributes whose value ranges in a discrete interval, with no total order on it, the task is more complicated as there is no underlying metric, and often depends on the meaning of each attribute. In fact, the problem is not only to compare the different values of a categorical attribute (e.g., is "blue" closer to "green" than "yellow" is to "red"?), but also how to estimate the relative importance of each attribute (e.g., is color more important than edges information when comparing images ?). A classic similarity measure for categorical data is the Hamming similarity, which simply counts the number of categorical attributes for which two samples have the same value. This obviously eludes a lot of information, but little can be done when there is no sufficient prior knowledge. See [**Boriah et al., 2008**] for a more detailed survey of similarity measures for categorical data.

Lastly, in more complex cases (e.g., mixed attributes type, complex data structures, particular application), there is often no conventional similarity measure to use. For example, if data samples are described by histograms, they can be seen as real-valued vectors whose order and relative positions of the coordinates matter. Depending on the application, a bin-to-bin comparison may not be as befitting as a similarity taking into account the inter-bin correlation (e.g., in images, the bins corresponding to "red" and "dark red" are visually similar, contrary to "red" and "cyan"). This example illustrates why we aim at building a similarity which requires limited prior knowledge, and is better suited to the task of clustering than a similarity chosen "by default" by the user.

1.1.3 Assessing the quality of a partition

Internal evaluation. As mentioned before, a good clustering maximizes the similarity between points in the same cluster, and minimizes the similarity between different ones. This property is evaluated through *internal* evaluation, by measures such as the *Dunn Index*. The *Dunn index* for a clustering $\mathcal{C}$, $DI(\mathcal{C})$, relies on two measurements. Δ_i is the diameter of the cluster C_i and represents the notion of intra-class distance. For instance, one can choose Δ_i to be the mean of all pairwise distances in C_i, or the maximum of these distances. Similarly, we define $\delta_{i,j}$, the inter-class measure, as a distance between clusters; e.g., the distance between the clusters' centroids. Given these, the Dunn index is then defined as:

$$DI(\mathcal{C}) = \frac{\min_{i \neq j} \delta_{i,j}}{\max_p \Delta_p}$$

We observe that a high value of $DI(\mathcal{C})$ means large inter-class distances and low intra-class distances, which exactly corresponds to the idea of a good clustering.

However, internal evaluation only ensures that the clustering process divides the space in a satisfying way, not that the resulting clusters are meaningful for the application at hand. Therefore, a second type of evaluation, namely *external* evaluation, is needed. For a given application, a dataset $\mathcal{X}$ and input distance d, we denote the optimal clustering as $\mathcal{C}^*(\mathcal{X}, d)$. $\mathcal{C}^*$ is often given as ground-truth, i.e., annotated data used during the evaluation process only.

External evaluation. External evaluation is the process of comparing the clustering obtained by the algorithm against the optimal one. We present an example of such measures below. See [**Halkidi et al., 2001**] for a more exhaustive study.

The *Rand index* [**Rand, 1971**] is an intuitive mean of comparing two clusterings. Mathematically, it corresponds to the ratio of pairs of data points on which both clusterings agree. The higher the Rand index, the closer the clusterings are. Given two clusterings $\mathcal{C}^1$ and C^2 of $\mathcal{X}$ we define the Rand index $RI(\mathcal{C}^1, \mathcal{C}^2)$ by:

$$RI(\mathcal{C}^1, \mathcal{C}^2) = \frac{\#\{\{x, y\} \mid x \neq y \text{ and } (s(x, y, \mathcal{C}^1) \text{ iff } s(x, y, \mathcal{C}^2))\}}{\binom{D}{2}}$$

where $s(x, y, \mathcal{C}) =$ `True` *iff* $\mathcal{C}$ groups x and y in the same cluster.

A major inconvenient of the Rand index is that its expected value is different for any two different clusterings, thus introducing a bias in the comparison. The corrected-for-chance version is the *adjusted Rand index*, whose expected value is constant of value 0 and whose maximum value is 1. It is one of the most usual evaluation measures for clustering tasks.

Among the large family of external validation measures (see for example [**Pfitzner et al., 2009**], Section 5, for a rather complete comparison and taxonomy of clustering external evaluation metrics), there exists no "perfect" measure. In our experiments we will report four usual measures: The *adjusted Purity* is the sum, for each cluster, of the ratio of correct samples pairs (same ground-truth class) to the total number of pairs in the cluster, averaged by the total number of samples. As its name suggests, a high purity characterizes a clustering whose clusters are "pure". The adjusted *Rand Index* introduced previously is based on pairs counting and takes into account both correctly and incorrectly classified pairs of points. Finally, the *V-measure* and *normalized mutual information* are two other popular evaluation metrics, both based on entropy and information theory notions, rather than pairs counting.

Information retrieval measures. Independently from clustering evaluation, we also introduce two usual information retrieval measures, the *mean Average Precision* and the average *F-measure*, which we use to assess the quality of a similarity measure, rather than the clusters obtained after applying a clustering algorithm. Formally, these two metrics are used in the context of *nearest neighbor retrieval.* For each sample x, we are provided with a ground-truth list of neighbors, $gtn(x)$. This list is then compared with the neighbors retrieved using the similarity we aim to evaluate, through classic measures such as the *Recall* and *Precision*, which assess the quality of the retrieval. If we denote by $\text{rank}_x(y)$ the rank of sample y in the list of samples ordered by decreasing order of their similarity with x, then the Precision and Recall at rank r are given by:

$$\text{Prec}(x)@r = \frac{|gtn(x) \cap \{y \mid \text{rank}_x(y) \leq r\}|}{r} \quad \text{and} \quad \text{Rec}(x)@r = \frac{|gtn(x) \cap \{y \mid \text{rank}_x(y) \leq r\}|}{|gtn(x)|}$$

In extension, the F-measure combines both precision and recall, and the average precision (AP), which is also derived from the precision, additionally takes into account the rank of the retrieved neighbors. In our experiments we report the mean F-measure at rank r ($f@r$) and mean Average Precision (mAP) which are the average of those measures over all samples.

$$f(x)@r = \frac{2 \text{ Prec}(x)@r \text{ Rec}(x)@r}{\text{Prec}(x)@r + \text{Rec}(x)@r} \quad \text{and} \quad AP(x) = \frac{1}{|gtn(x)|} \sum_{y \in gtn(x)} \text{Prec}(x)@\text{rank}_x(y)$$

The mAP and F-measure are more objective than the previous clustering metrics because they do not depend on the clustering algorithm and parameters we use. However they are also less informative as they only take into account the ranks of the samples, and not the similarity values themselves.

1.2 Clustering by diverting supervised learning approaches

As explained in the previous section, defining a suitable similarity measure for clustering problems is often fastidious and very data-dependent. Nonetheless, in recent years, several articles have exploited the idea of diverting supervised machine learning techniques for building a similarity on the data with few prior knowledge. Relying on the performance of existing classifiers allows for similarities with complex internal representations without having to define them explicitly by hand. However, most of the existing related work studies the idea from the point of view of one specific application. One of the contributions of the internship is a more formal and abstract model of the method for identifying the possible issues when applied on new frameworks. This will be developed later on in this report. In this subsection, we first introduce a simple unified framework to explain the general intuition of the method on the basis of recent related work. We then review two applicative frameworks for which such method has already been exploited in recent years.

1.2.1 Defining a similarity from a classifier's output

The task of classification can be seen as the supervised counterpart of cluster analysis. Formally, the goal is to build a classifier f that maps any input sample $x \in \mathcal{X}$ to a class $y \in \mathcal{Y}$, where $\mathcal{Y}$ is a set of classes known beforehand. Furthermore, in the usual framework of fully supervised classification, the user is provided with an annotated database of samples labeled with their ground-truth class. These are used as training set to learn a classifier that minimizes its mistakes on the training samples while preserving good generalization abilities for prediction on new inputs. On the opposite, cluster analysis is an unsupervised task, which means the input data is unlabeled, and the user has no, or very few, information about the optimal clusters. Hence clustering algorithms require an explicit input similarity measure, while classification algorithms are able to learn the needed information through their training step.

Consequently, the output of a classifier can be used to infer a notion of similarity on the dataset. When two objects are assigned to the same class, it implies that the classifier was able to identify some resemblance between them, relatively to the model that was learned. We formalize this idea by defining, for any classifier f, a binary similarity function, s_f, induced by the classifier's output:

$$\forall x,\ y \in \mathcal{X},\ s_f(x,y) = \begin{cases} 1, & \text{if } f(x) = f(y) \\ 0, & \text{otherwise} \end{cases} \quad .$$

The definition could be refined into a continuous one when the classifier yields probabilistic outputs, i.e., for each sample, a degree of membership to each class. Another way of refining this similarity is by computing the mean of several s_f functions from classifiers trained in different settings. This definition is more robust to the bias introduced by the learning process (e.g., choice of the training set) than the one using only one classifier. The main advantage of this similarity construction is that the classifier behaves as a black box and manages the internal data representation by itself. This also allows the user to use more complex data descriptions that would be difficult to manipulate by hand. Furthermore, this similarity measure can be defined on any type of data, as long as a befitting classifier exists. This is especially useful in frameworks where usual metrics are not applicable. For instance in [**Claveau and Gros, 2014**] the authors divert supervised ILP (Inductive Logic Programming) techniques to define a similarity measure on relational data. In fact, most usual distance measures do not apply in this case, as the database does not follow the classical model where a sample is represented by real-valued vectors (attribute-value model).

For these reasons, this similarity construction is a good candidate for clustering applications. The only parameter to determine is the classifier f, which should be straightforward in many cases since clustering tasks often have a supervised classification problem counterpart. We can also rely on the background in supervised learning techniques to refine the measure and investigate its different properties. However, in order to use this construction for clustering, we first need to extend it to unsupervised frameworks. To do so, adequate synthetic data and annotation are generated, and are then given as input of supervised classifiers from which we infer the aforementioned similarity.

In the next paragraphs, we review the two articles that first introduced this idea and applied it to the framework of clustering. Secondly, we review a similar approach which deals with unsupervised labeling of named entities by diverting state-of-the-art text mining techniques. This is also one of the experimental frameworks studied during this internship. Finally, other related works are found in the fields of image clustering [**Perbet et al., 2009**] and data-dependent image indexing [**Joly and Buisson, 2011**]. These two articles exploit classifiers' output in an unsupervised framework in order to yield a partition of the original data space. However, we will not develop them further due to space limitations.

1.2.2 Random forest clustering

One of the first examples of applications of such technique, to the best of our knowledge, appears in a 2006 article [**Shi and Horvath, 2006**] and the corresponding technical report [**Shi and Horvath, 2005**]. The article introduces the "random forest dissimilarity", built by diverting a set of supervised *decision tree* classifiers, for the purpose of cluster analysis. It is then used as input of a k-medoids clustering algorithm, and applied to a medical analysis problem. This work is partially based on an older article [Liu et al., 2000], which also diverts supervised decision tree classifiers for clustering purposes, but does not explicitly introduce a notion of similarity.

In the first paragraph we give a brief introduction to random forests classifiers. Then, we review the approaches proposed in the two articles.

Random forests. A random forest classifier [Breiman, 2001] is a *bagging* process based on decision trees, which are very intuitive supervised classifiers taking samples described by vectors of attributes as input. The core idea of a decision tree is to split the data space on the attribute that separates it the best (the split criterion is evaluated with measures such as the entropy). This process is iterated on the resulting sub-trees, until no attribute is left or all samples in the sub-tree belong to the same class. For categorical (discrete) attributes, the split yields as many sub-trees as possible values; For numerical (continuous) attributes, we use thresholding rules, yielding 2 subtrees each. We illustrate the process with an example of decision tree in Figure 3.

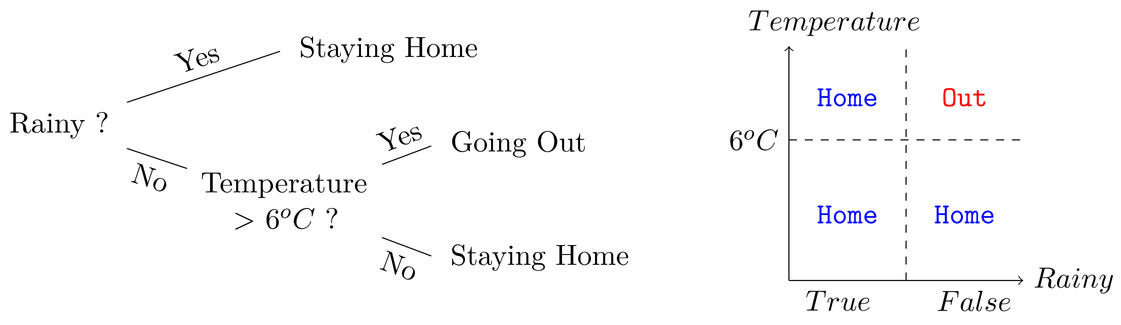


Figure 3: On the left is an example of decision tree with mixed binary and numerical attributes; On the right is the corresponding partitioning of the attributes space.

A *bagging* process is a set of classifiers, trained independently, possibly with different parameters. Each sample is sent through each classifier of the bagging, and their outputs is then combined to yield the final decision. In the case of random forests for example, the classifier's prediction for a new input sample is given by a majority vote on the outputs of the decision trees in the forest.

Diverting the random forest and decision tree classifiers. Both articles rely on the idea of using decision trees to distinguish the original unlabeled data $\mathcal{X}$ from a set of artificial data samples, $\mathcal{A} = \{a_1 \dots a_P\}$. In this particular framework, samples are described by a set of attributes, i.e., real-valued vectors. A class ω_o is assigned to all the original samples, and a class ω_A the the synthetic ones. Then, T decision trees $\{f_1 \dots f_T\}$ are trained on different subsets of $\mathcal{S} \cup \mathcal{X}$ and used for clustering purposes.

In [Liu et al., 2000], a single decision tree is used ($T = 1$), with the goal of distinguishing dense regions from sparse ones in the original unlabeled dataset. The synthetic dataset is generated as P randomly uniformly sampled points on the whole data domain. The authors observe that dense regions in the original dataset should contain more original than synthetic samples, and thus be separated from the sparse regions, which logically contain more synthetic points. Based on this observation, the decision tree is trained on $\mathcal{A} \cup \mathcal{X}$, and the partition induced by the tree is returned as the final clustering (see Figure 3 for an illustration of how decision trees produce a partition of the attributes space). However, the method was only tested on numerical data, and it only generates hyper-rectangular clusters, which are not suited for all applications, especially real-life ones. In fact the only real-life experiment in the original paper is performed on a rather small dataset (4 clusters, 12-dimensional features vectors), and while the results are promising, there are not compared against other clustering techniques. To summarize, this is an interesting clustering algorithm which scales well due to the efficiency of decision trees, however it produces too simple models and does not fit our goal of building a similarity.

In the second article [Shi and Horvath, 2006], synthetic samples are generated by independently sampling each attribute from its empirical marginal distribution in the original data: As the only major difference between the synthetic and original samples is the independence of the attributes, the classifiers focus their splits on variables that are dependent in the original data, which proves beneficial for the application tackled in the article. Several decision trees are then trained on a different subset $T \subset \mathcal{A} \cup \mathcal{X}$, and the rest of the samples is run through each tree, yielding a binary similarity measure for these samples, as defined previously. The final similarity on the original dataset $\mathcal{X}$ is obtained by averaging these measures for the whole forest:

$$s = \frac{\sum_i s_{f_i}}{T}_{|(\mathcal{X} \times \mathcal{X})} \tag{1}$$

This work is very similar to our objective, but the proposed framework is limited. The similarity measure is only applied on numerical data with a simple vector representation. Furthermore, for the generation of the synthetic dataset, it is assumed that the data distribution (or at least an empirical one) is known, which is not always the case. Finally, while decision tree classifiers are good candidates because they are efficient and able to manage mixed-type attributes, they will not be used in our experiments as there are other classifiers more suited for the task.

1.2.3 Unsupervised labeling of named entities

The second example [Claveau and Ncibi, 2014] deals with the task of unsupervised labeling named entities in text documents. Named entities are textual information about the actors, location or time of an event, e.g., names of persons, organizations, places, dates. Given a set of unlabeled named entities and their textual context, the goal is to correctly categorize them, for applications to automatic description of textual content for example.

A) Baseline algorithm. Most approaches to named entities labeling actually use supervision, hence there does not exist many state-of-the-art methods for the unsupervised counterpart. In their contribution, Claveau et al. compare their diversion technique against the manually-defined similarity measure introduced in [Ebadat et al., 2012]. In this work, a named entity is described by several bag-of-words features (a *bag-of-words* is a vector counting words frequencies on a text excerpt) built from the context of each occurrence of the entity (the context is represented by a limited window centered around the word). This description format is then coupled with usual similarities such as the *Cosine* similarity for example. This approach obtains good results on the reported experiments, but it illustrates the fact that defining an adequate similarity is not easy for this task.

B) Diverting conditional random fields to infer a similarity. We now review the approach proposed by Claveau et al. in their article, in which they divert *C*onditional *R*andom *F*ield (*CRF*) classifiers [Lafferty et al., 2001] to build a similarity between named entities, in a way similar to random forest clustering. However, text documents having a sequential structure, random forests would perform poorly on this kind of data, hence the choice of CRFs.

Identifying named entities. The first step for the task is to identify the named entities, which are the samples we want to compare and eventually, cluster. The other, common, words only act as context information for the learning process. Distinguishing named entities from normal words is also a clustering process, however distinct from the task at hand (categorizing the named entities). For simplification, we assume the locations of the named entities in the documents are known beforehand. This information is contained in a *BIO* labeling of the dataset: "O" labels are associated with common words, and for named entities, "B" labels are assigned to the beginning word and "I" labels to the remaining ones, if any.

The step we are interested in is the categorization of named entities, i.e., extending B and I labels with a type, e.g., B-date, I-date... Because we study the task in an unsupervised framework, applying a clustering algorithm is the natural approach here. However, defining a similarity between named entities is not straightforward, as many categorical features should be taken into account (the named entity itself, its grammatical information, but also those of the surrounding words...), and it is not clear how these features articulate; e.g., to identify persons' names the grammatical information is very important (proper noun), while for temporal markers the context plays a bigger role (temporal prepositions). To alleviate the problem, the authors propose to infer a similarity measure by diverting CRFs, which are state-of-the-art classifiers for labeling and segmenting text, making them natural candidates.

Conditional random fields. CRFs are widely used for applications on textual data because they efficiently capture sequential information such as relationships between data samples and their annotation, as well as other samples and annotations around them. A detailed study can be found in the following technical report [Klinger and Tomanek, 2007]. For computation sake, *linear-chain* CRFs are used in practice. This means that for a sentence, at each position, we only express a relation between the current label, the preceding one, and all the words around. These relations are defined by a set of binary feature functions $\phi_j(\mathbf{x}, y_t, y_{t-1}, t) : \mathcal{X}^n \times \mathcal{Y} \times \mathcal{Y} \times \mathbb{N} \to \{0; 1\}$. For instance, the rule ($x_t =$ "*February*" $\wedge\, y_t =$ "*Date*" $\wedge\, y_{t-1} =$ "*Preposition*") means that the word "February", when following a preposition, refers to a date. These rules are automatically inferred from the training set, rather than defined manually. The user provides a pattern, indicating which features should be taken into account for the rule, and it is then matched to all training samples, and combined with the label information (y_t and y_{t-1}) in order to automatically generate a set of rules; e.g., for the previous rule the pattern simply indicates to look at the word at position t which is later combined with the label information. During the learning step, each feature function ϕ_j is associated with a weight λ_j, determined with an optimization algorithm. The final classifier assigns to a sentence $\mathbf{x} = \{x_1 \dots x_n\}$ the labels sequence $\mathbf{y} = \{y_1 \dots y_n\}$ maximizing the following probability:

$$\mathbb{P}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} \prod_{t=1}^{n} \exp\left(\sum_j \lambda_j \phi_j(\mathbf{x}, y_t, y_{t-1}, t) \right),$$

where Z is a normalization constant. Lastly, CRFs are able to manage multiple multi-type attributes descriptions (although most implementations only support categorical features), as well as to detect which features are more or less relevant for the task at hand (through the λ weights). This is a favorable property in the current unsupervised framework since the user may not know *a priori* which features are the most adequate for the target application.

Synthetic data generation. In the random forest example, synthetic samples are generated and used for training. While different from the original data samples, they are still similar enough, so that the classifiers focus on the small, non-trivial, and hopefully meaningful, differences between the two classes. Applying the same idea here would amount to generate grammatically correct sentences, with named entities distributed in a slightly different manner than the original data. This would be too complicated, as the distribution of named entities in natural language is not something straightforward. Instead, Claveau et al. choose to preserve the original dataset $\mathcal{X}$ and to provide the classifiers with a synthetic annotation of the samples. In practice, T CRF classifiers, $\{f_1, \dots f_T\}$, are introduced. Each f_i is trained on a training subset $\mathcal{T}_i \subset \mathcal{X}$ where every sample is uniformly randomly annotated with one of L possible synthetic labels. The remaining samples, $\mathcal{X} \setminus \mathcal{T}_i$, are run through the CRF, which yields a binary similarity s_{f_i} for these samples, as defined previously. The final similarity measure is defined as the average of the induced similarities for all CRF, similarly to Equation (1).

However, the main issue of the method as it is introduced in this article is that it lacks details about the actual properties of the obtained similarity function, and how the latter is influenced by the different parameters and data distribution. In particular, the number of synthetic labels should impact how much information is brought at each iteration since it influences the granularity of the classification. Another issue which is not tackled is the convergence of the method and the number of classification iterations. This however directly impacts the computational cost of the process, which, even if easily parallelizable, requires many training and testing steps.

Nonetheless, the reported clustering results outperform classic similarities, which is promising for the use of this approach in unsupervised frameworks.

C) The (textual) ESTER2 dataset. Before going further into detail, we give a brief introduction of the dataset that we use for the task of unsupervised named entity labeling. In fact, even though the next section focuses on theoretical aspects, we use this dataset to validate our theoretical model in practice. The ESTER2 dataset is originally an audio dataset containing several hours of French news radio shows. For the named entities recognition task, we use a textual representation of a subset of ESTER2, which amounts to 3886 sentences and 5112 named entities occurrences (although a word can appear several times in the dataset, each occurrence is considered to be a single entity). Finally, these named entities are distributed among 8 ground-truth clusters of varying sizes (see Figure 4) which will be used for evaluation.

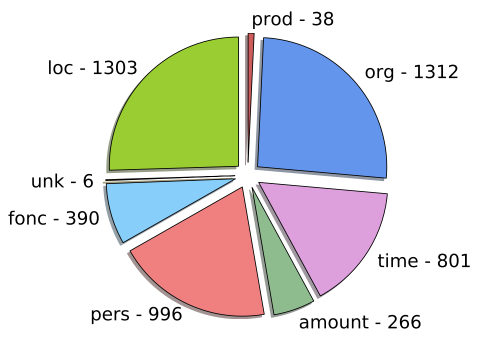


Figure 4: Ground-truth clustering for the ESTER2 text dataset

2 SIC: Similarity by Iterative Classifications

In this section, we propose a more formal study of the idea of diverting supervised classifiers to infer a similarity measures. We call the resulting model SIC. In the first subsection, we define the main SIC approach which uses binary updates, and propose a theoretical model of the obtained similarity distribution. We then show how to estimate the parameters of the model in an unsupervised framework using the Expectation-Maximization (EM) algorithm. In the second subsection, we present a variant of SIC in which we derive a similarity from the estimates made by the EM algorithm. Finally, in the third subsection, we propose to generalize the initial SIC by using weighted updates instead of binary ones.

Framework and Notations. Let $\mathcal{X} = \{x_1, \ldots, x_D\}$ be the set of samples for which we aim to define a similarity $s : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. In order to assess the quality of s, we rely on a ground-truth clustering of the data, $\mathcal{C}^*$, only available during the evaluation phase. In the rest of the section, we denote by k the number of ground-truth clusters, themselves being denoted by ${w_1}^*, \ldots, {w_k}^*$. We also introduce a binary symmetric relation, $\sim$, indicating that two samples are *truly* similar (i.e., they belong to the same ground-truth cluster), and the converse relation, $\perp$.

An iterative construction. The proposed approach relies on the idea that two objects assigned with the same class by a supervised classification algorithm share some kind of resemblance, relatively to this classifier. We propose to build the target similarity s as a sum of local quantities computed over N independent classification iterations. More specifically, at each iteration i, we train a classifier c_i over a training set Tr_i. The classifier parameters and the training set are both randomly chosen anew at each iteration in order to avoid bias towards some specific training settings. We then apply this classifier on a testing set Te_i, which yields a partition of Te_i. From this partition, we derive a score $s_i(x_p, x_q)$ for each pair of samples (x_p, x_q) appearing in the testing set, which finally yields the following expression for the similarity s over N iterations as:

$$\boxed{s^N(x_p, x_q) = \frac{1}{T^N(x_p, x_q)} \sum_{i=1}^{N} s_i(x_p, x_q)\ \mathbb{1}_{\{x_p, x_q\ \in\ Te_i\}}} \tag{2}$$

where $T^N(x_p, x_q) = \sum_{i=1}^{N} \mathbb{1}_{\{x_p, x_q\ \in\ Te_i\}}$ is a normalization constant. It is the number of times x_p and x_q appeared together in the same test dataset over all iterations.

2.1 Binary scores

2.1.1 Definition

We first propose a straightforward way to define the similarity, by counting the number of co-classifications between samples:

Definition 1. *We define $sb_i : Te_i \times Te_i \to \{0; 1\}$, the binary score function at iteration i, as the following quantity:*

$$sb_i(x_p, x_q) = \mathbb{1}_{c_i(x_p) = c_i(x_q)} = \begin{cases} 1, & \text{if } c_i(x_p) = c_i(x_q) \\ 0, & \text{if } c_i(x_p) \neq c_i(x_q) \end{cases}$$

If we now take $s_i = sb_i$ in Equation (2), then the resulting similarity, $s(x_p, x_q)$, is the frequency at which x_p and x_q are classified together over all iterations, and can therefore be interpreted as a rough estimate of the probability for x_p and x_q to be classified together, i.e., $s(x_p, x_q) \simeq \mathbb{P}(c(x_p) = c(x_q))$. This reflects the underlying idea that two similar objects should often be classified together.

2.1.2 Modeling the similarity distribution for binary scores

Ultimately, our goal is not only to build a similarity, but also to use it as input of different applications, hence it is also important to understand the general behavior of the similarity we obtain for future use. In this subsection, we first define an exact probabilistic model of $S(x_p, x_q)$, the random variable associated with the similarity value $s(x_p, x_q)$. However, due to the large number of parameters required, we then fall back to a simpler model in the second paragraph, and explain how to estimate its parameters in the next subsection.

From Bernoulli to Poisson-Binomial. In this paragraph, we propose a theoretical probabilistic model for $S(x_p, x_q)$, which is the random variable associated with the value of the similarity between x_p and x_q. From the expression of the binary scores in Definition 1, the random variable associated to the score at iteration i, $S_i(x_p, x_q)$, is simply a *Bernoulli* variable of parameter $p_i(x_p, x_q) = \mathbb{P}(c_i(x_p) = c_i(x_q))$; i.e., it takes the value 1 with probability $p_i(x_p, x_q)$ and 0 otherwise.

Consequently, the sum $\sum_i S_i(x_p, x_q)$ follows a *Poisson-Binomial* distribution, which is the distribution of a sum of Bernoulli variables of different parameters (it generalizes the usual Binomial distribution in case of non-identical parameters). In particular, its probability density function can be efficiently computed using the Fast Fourier Transform and the closed-form expression proposed in [**Fernandez and Williams, 2010**]. Or, even more simply, it can be approximated accurately by a Poisson distribution of parameter $\lambda = \sum_i p_i(x_p, x_q)$ given that the $p_i(x_p, x_q)$ are small enough [**Hodges and Cam, 1960**].

Additionally, in order to model $S(x_p, x_q)$, we also have to take into account that samples do not appear in the test set at every iteration. In the end, the probability that the similarity $s(x_p, x_q)$ equals the value $\frac{k}{t}$ (i.e., x_p and x_q occurs t times in the same test set and are classified in the same class k times) is given by the following expression: (for simplicity, we sometimes drop the dependency on (x_p, x_q) in the notations)

$$\begin{aligned}
\mathbb{P}\left(S^N(x_p, x_q) = \frac{k}{t} \mid 0 \le k \le t \le N\right) &= \mathbb{P}\left(\sum_i S_i \, \mathbb{1}_{x_p, x_q \in Te_i} = k \mid T^N = t\right) \mathbb{P}(T^N = t) \\
&= \left(\sum_{A \in F_t^N} \sum_{B \in F_k^A} \prod_{i \in B} p_i \prod_{i \notin B} (1 - p_i)\right) \mathbb{P}(T^N(x_p, x_q) = t; x_p, x_q) \\
&= \left(\sum_{A \in F_t^N} \mathcal{PB}((p_i)_{i \in A}))(k)\right) \left(\sum_{B \in F_t^N} \prod_{i \in B} \alpha_i^2 \prod_{i \notin B} 2\alpha_i (1 - \alpha_i)\right)
\end{aligned}$$

where $\mathcal{PB}$ denotes the Poisson-Binomial distribution, F_t^N is the set of all combinations of t elements in $[\![1; N]\!]$ and F_k^A all the combinations of k elements in A. α_i is the probability for any given sample to belong to the test set Te_i (which we assume to be independent on the sample for simplicity). Furthermore, in our experiments, α_i is also independent on the iteration, hence the following simpler expression:

$$\boxed{\mathbb{P}\left(S^N(x_p, x_q) = \frac{k}{t}\right) = \binom{N}{t} \alpha^{2t} (2\alpha(1 - \alpha))^{N-t} \left(\sum_{A \in F_t^N} \mathcal{PB}((p_i(x_p, x_q))_{i \in A}))(k)\right)} \tag{3}$$

From now on, and until the end of the section, to simplify the computations we consider that $\alpha = 1$ (i.e., all samples always appear in the test database and $T^N(x_p, x_q) = N$). In this case, the model amounts to a simple Poisson-Binomial with a multiplicative weight, which yields the following distribution:

$$\boxed{\begin{gathered}
\mathbb{P}\left(S^N = \frac{k}{N}\right) = \mathcal{PB}\left((p_i)_{i=1}^N\right)(k) \\
\mathbb{E}(S^N) = \frac{1}{N} \sum_{i=1}^N p_i \qquad \mathbb{V}(S^N) = \frac{1}{N^2} \sum_{i=1}^N p_i (1 - p_i)
\end{gathered}} \tag{4}$$

We now have a fairly simple model of the similarity distribution for one pair of samples which depends on N parameters $(p_i(x_p, x_q))_i$. However, it would not be feasible to estimate the $ND(D-1)/2$ parameters requires to model the distribution for each pair of samples, therefore we propose a simpler model in the next paragraph to alleviate this problem.

Towards a refined Bernoulli mixture. Instead of distinguishing every single pair of samples as we considered previously, we can group them in only two categories induced by the relations $\sim$ and $\perp$, i.e., the pairs of samples that belong to the same ground-truth clusters and those that do not, even though these relations are not known. This allows us to reduce the number of parameters by assuming that the distribution of similarities for pairs of samples falling in the same category are identical. Formally, we highlight this binary distinction by applying the law of total probability:

$$\begin{aligned}
\mathbb{P}(s_i(x_p, x_q) = 1) &= \mathbb{P}(c_i(x_p) = c_i(x_q)) \\
&= \mathbb{P}(x_p \sim x_q)\mathbb{P}(c_i(x_p) = c_i(x_q) \mid x_p \sim x_q) + \mathbb{P}(x_p \perp x_q)\mathbb{P}(c_i(x_p) = c_i(x_q) \mid x_p \perp x_q) \\
&= \pi_1 p_{1,i} + \pi_0 p_{0,i} \\
\implies S_i = \pi_1 \mathcal{B}(p_{1,i}) + \pi_0 \mathcal{B}(p_{0,i}) & \qquad (5)
\end{aligned}$$

where π_1 is the probability that any two given samples belong to the same ground-truth class (independently from the class), and $p_{1,i}$ is the probability that two samples are classified together at iteration i, given that they have the same ground-truth class.

From its expression in Equation (5), the variable S_i is now a *mixture* of two Bernoulli components of respective parameters $p_{1,i}$ and $p_{0,i}$. These components characterize the similarity distribution for, respectively, the pairs of similar ($\sim$) and unrelated ($\perp$) samples. Consequently, the full similarity S, obtained by summing the S_i and normalizing, is also a mixture of two components, which have the same expression as the one given in Equation (3), except their parameters are now respectively $(p_{1,i})_i$ and $(p_{0,i})_i$. The exact expression of the mixture is given by (generalization of Equation (4)):

$$\mathbb{P}\left(S^N = \frac{k}{N} \mid 0 \leq k \leq N\right) = \pi_0 \mathcal{PB}(\mathbf{p_0})(k) + \pi_1 \mathcal{PB}(\mathbf{p_1})(k) \qquad (6)$$

Finally, note that this model could be refined by considering more categories. For instance, if we distinguished pairs of samples according to each ground-truth class, we would have C^2 Bernoulli components $B(p_{w,w',i})$, representing the pairs of samples such that one is in ground-truth class w and the other in class w', i.e., $p_{w,w',i} = \mathbb{P}(c_i(x_p) = c_i(x_q) \mid x_p \in w; x_q \in w')$. Exploiting the idea further would eventually lead to the initial model where we consider a component for each possible samples pair $(x_p,\, x_q)$. While more precise, these complex models would be more troublesome to handle, and would require a more expensive parameters estimation.

Similarity and relevance. Using this relatively simple mixture model of the similarity distribution, we would like to estimate the *relevance* or discriminative power of the similarity s. For example, finding a threshold α such that if $s(x_p, x_q) > \alpha$ then we can assess, with a certain known risk, that x_p and x_q belong to the same ground-truth class. The 2-components model proposed in Subsection 2.1.2 is well fitted for this idea, since it distinguishes samples based on the $\sim$ relation, and only requires to estimate $2N + 1$ parameters. Therefore, we will focus on this model in the rest of the section. In particular, in the next subsection, we propose two main ways of estimating its parameters in the current unsupervised framework, where the relations $\sim$ and $\perp$ are not known.

2.1.3 Parameters estimation

In this subsection, we propose two methods to estimate the $p_{1,i}$, $p_{0,i}$ and π_0 parameters in the aforementioned 2-components mixture model.

A) Estimates under an independence assumption. Given x_p and x_q, any two samples such that $x_p \perp x_q$, we can assume they have nothing in common and that they are *independently* assigned a label by the classifier. Hence an estimate for $p_{0,i}$, the probability for two truly dissimilar samples to be classified in the same class, would be given by:

$$\begin{aligned}\tilde{p}_{0,i} &= \mathbb{P}(c_i(x_p) = c_i(x_q) \mid x_p \perp x_q) \\ &= \sum_{j=1}^{n_i} \mathbb{P}(c_i(x_p) = j)\ \mathbb{P}(c_i(x_q) = j) \\ &= \sum_{j=1}^{n_i} \left(\frac{|{w_j}^i|}{\sum_{k=1}^{n_i} |{w_k}^i|} \right)^2 ,\end{aligned}$$

where n_i is the number of synthetic classes assigned to the testing set at iteration i, and ${w_1}^i, \dots, {w_{n_i}}^i$ the aforementioned classes. Their cardinals can easily be computed at testing time and used to estimate $p_{0,i}$. While simple, this assumption of independence can not be used to estimate the other parameters (π_0 and $p_{1,i}$), hence we describe a more complete estimation method in the next paragraph.

B) Estimating the parameters with the Expectation-Maximization procedure. In Equation (5), we expressed S_i as a mixture of two Bernoulli components. This can be generalized to the scores over several iterations. More formally, let us define $c^n(x_p, x_q)$ the binary vector containing the scores for the pair $(x_p,\, x_q)$ over iterations 1 to n (i.e., $c^n(x_p, x_q) = (s_1(x_p, x_q) \dots s_n(x_p, x_q))$.Then, as previously, the corresponding random variable can be expressed as a mixture of two *multivariate* Bernoulli components:

$$C^n = \pi_1 \mathcal{B}(\mathbf{p_1}) + \pi_0 \mathcal{B}(\mathbf{p_0}), \tag{7}$$

where, similarly, we have $\pi_1 = \mathbb{P}(x_p \sim x_q; x_p, x_q)$ and $\pi_0 = 1 - \pi_1 = \mathbb{P}(x_p \perp x_q; x_p, x_q)$. The Bernoulli parameters, $\mathbf{p_1}$ and $\mathbf{p_0}$, are now n-dimensional vectors representing the parameters for each iteration; i.e., $\mathbf{p_k} = (p_{k,1} \dots p_{k,n})$. These parameters can be automatically estimated with the Expectation-Maximization algorithm. By considering n iterations instead of 1, we now have more observations, which allows us to compute more precise estimates [Juan and Vidal, 2004].

The Expectation-Maximization (EM) algorithm relies on two sets of variables: a set of observations, $(o_{(p,q)})_{p,q}$, associated with the variable C^n, and a set of hidden (unknown) component indicator variables, $z_{(p,q)}$, which indicates whether x_p and x_q are neighbors or unrelated samples ($z_{(p,q)} = \mathbb{1}_{x_p \sim x_q}$). The EM scheme is an iterative algorithm used for estimating mixture parameters when the complete data is not available (for example, in this case the component membership for each observation is unknown). It works by alternatively estimating a likelihood quantity Q (Expectation step), and maximizing this function against the parameters $\theta = (\pi_0, p_1, p_0)$ (Maximization step) until the likelihood of the data reaches a satisfying threshold. See Appendix A for the mathematical details.

C) Experiments. We conduct experiments on the NER dataset in order to compare the two aforementioned estimation methods. We also use the corresponding ground-truth clustering to compute ground-truth frequency estimates of the parameters as a reference point:

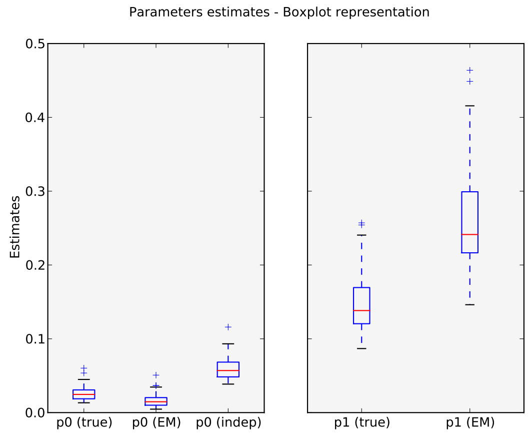
$$p_{1,i} = \frac{|\{p,q\} \text{ s.t. } x_p \sim x_q \text{ and } c_i(x_p) = c_i(x_q)|}{|\{p,q\} \text{ s.t. } x_p \sim x_q|} \qquad \pi_0 = \frac{2 \times |\{p,q\} \text{ s.t. } x_p \perp x_q|}{D(D-1)}$$

$$p_{0,i} = \frac{|\{p,q\} \text{ s.t. } x_p \perp x_q \text{ and } c_i(x_p) = c_i(x_q)|}{|\{p,q\} \text{ s.t. } x_p \perp x_q|} \qquad \pi_1 = 1 - \pi_0$$

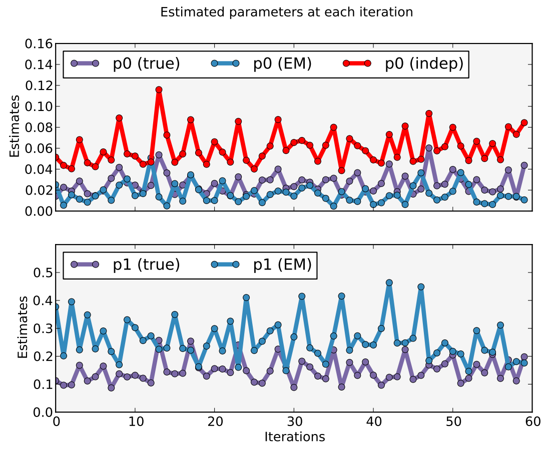
In Figure 5, we present the results of the different estimation methods, applied to an experiment over 60 classification iterations ($N = 60$). The whole database is chosen as the test dataset at each iteration, in order to have $\forall p, q,\ T^N(x_p, x_q) = N = 60$, as we assumed in the theoretical study. Subfigure (a) provides the mean and standard deviation over all iterations of the estimates for the three methods (true parameters, EM estimation, estimation with independence assumption). Subfigure (b) contains the corresponding whisker plots. Finally, Subfigure (c) is a detailed representation of the estimated parameters for each method at each iteration.

	π_0	$p_{0,i}$	$p_{1,i}$
True Parameters	0.798	0.0263 ± 0.01	0.148 ± 0.04
EM	0.859	0.0165 ± 0.009	0.260 ± 0.075
Independence	-	0.0607 ± 0.016	-

(a) Parameters means and standard deviations over the 60 iterations



(b) Box Plot representation



(c) Parameters estimates at each iteration

Figure 5: Result of the three parameters estimation methods over 60 iterations.

We first observe that the estimated $p_{0,i}$ are very close to their true values, especially when using the EM algorithm. Furthermore these probabilities are very small, i.e., samples which are truly dissimilar to each other are seldom classified together. Conversely, a high similarity implies that the two samples are in fact truly similar, which is a good property for the proposed similarity construction. On the contrary, the EM algorithm tends to over-estimate the ($p_{1,i}$) parameters (+0.12) and the proportion of truly dissimilar pairs in the dataset, although by a smaller margin ($\pi_0 + 0.06$). Furthermore, the ground-truth values for $p_{1,i}$ are quite small on average: Only 14%

of the truly similar samples are classified together at each iteration. From these observations, we deduce that pairs of samples with high similarity scores are most likely to be truly similar, however the converse implication (lower similarity $\implies$ dissimilar) is not as clear.

The same conclusion can be made from Figure 6: Following Equation (6), we plot the two Poisson-Binomial components obtained with the true parameters and the EM estimates, as well as the resulting mixture distribution. We observe that the EM algorithm over-estimates the $p_{1,i}$ parameters and leads to two well separated components, which is not the case for the true parameters. We can only conclude the same as before: While very high or very low similarity scores should be trusted, there is an area where the components intersect, corresponding to similarity values more difficult to interpret. A better understanding of this area where the components intersect would

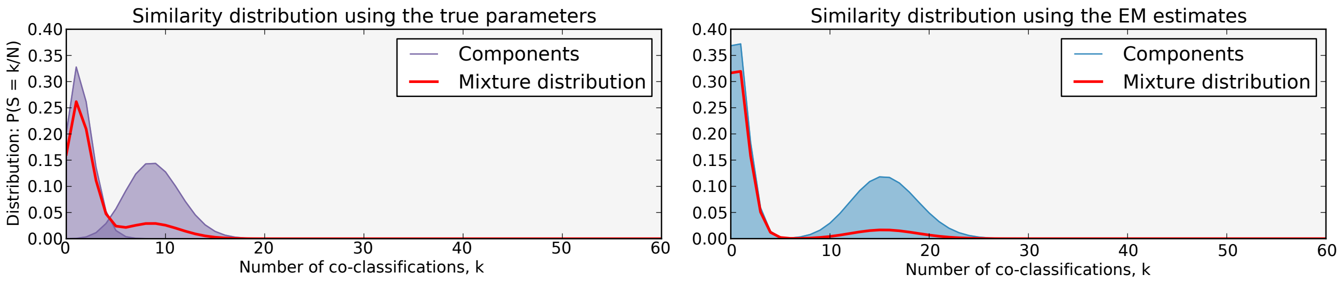


Figure 6: Estimated Poisson-Binomial components (*60 iterations*) and the resulting mixture (*in red*) for the true parameters (left) and EM estimates (right).

allow us to know to which extent we can trust the similarity values obtained with our approach. However this may be difficult in the current framework as the Poisson Binomial distribution has a complex expression. In the next subsection, we propose an EM-based similarity, which leads to a simpler model for the distribution for which the two components can be more easily distinguished.

2.2 Another usage of the EM algorithm

EM Similarity. In Subsection 2.1.3, we described the EM procedure as a way of estimating a Bernoulli mixture's parameters (Maximization step). However, during the Expectation step of the algorithm, we also estimate the quantity $\gamma_{1,(p,q)} = \mathbb{P}(z_{(p,q)} = 1 \mid \hat{\theta})$ which represents the probability that samples x_p and x_q belong to the same ground-truth class. Intuitively, this is also an indication of how similar the two samples are, hence the following similarity definition:

Definition 2. *The* ***EM similarity*** *is given by the value of the hidden component indicator estimated during the expectation step of the EM algorithm applied to the previous mixture distribution:*

$$s(x_p; x_q) = \gamma_{1,(p,q)}$$

Experiments. The main advantage of this approach is that the resulting similarity distributions display well separated components in practice. In Figure 7, we represent the similarity distribution for one given sample x (i.e., one line in the similarity matrix) using the binary scores presented previously (left) and the EM similarity (right). We have drawn in purple the component corresponding to samples with a different ground-truth class than x ($S_\perp$), and in blue the one corresponding to truly similar samples ($S_\sim$). In the EM similarity case, we observe a better distinction between the two components, while for the classic binary similarity, they tend to be closer to each other, making it more difficult to distinguish truly similar from dissimilar pairs of samples. Another interesting point is that in practice this peculiar distribution for EM is obtained for all samples and no matter what the number of classification iterations, N, is. On the contrary, for the binary scores, the distance observed between the two Poisson-Binomial components depends on these two parameters.

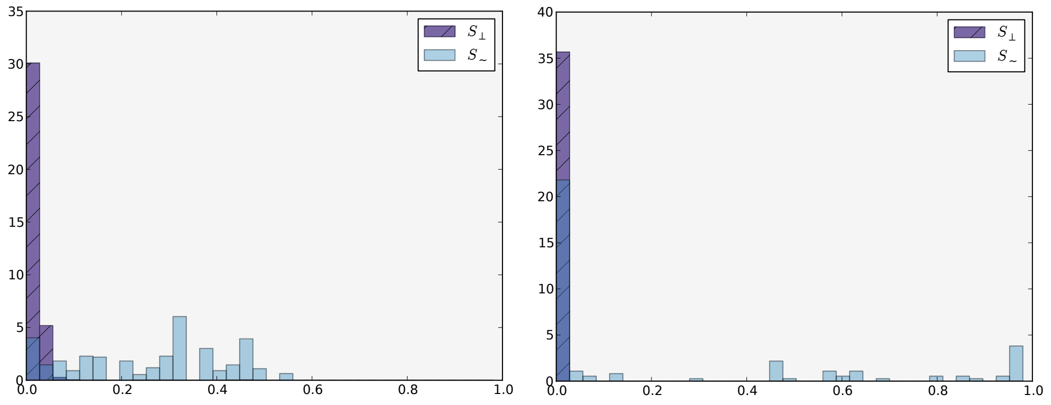


Figure 7: Similarity distribution on 30 iterations for sample "1726-capitaine" for the binary scores (left) and the EM similarity (right)

The EM similarity is thus useful when working with a small number of iterations, as it makes it easier to distinguish good pairs of samples (with a similarity close to 1) from dissimilar ones (similarity close to 0). However when N is large, the binary scores variant gains in precision and the difference between the two distribution shapes tends to be less noticeable. In fact the EM similarity does not influence the samples co-classifications, but rather how the similarities are distributed in the end. For the same reason it does not have a crucial impact on the results for the clustering.

2.3 Weighted scores

In this section, we propose to generalize the previously defined binary scores by introducing real-valued weights. This allows us to constrain the model (at least a part of it) in order to obtain a Gaussian distribution which is easier to manipulate than the Poisson-Binomial one.

2.3.1 Definition

We denote by ai (resp. bi) the score given to all pairs of samples (x_p, x_q) such that $c_i(x_p) = c_i(x_q)$ (resp. $\neq$). The resulting model is a classic reward/penalty scheme.

Definition 3. *We define $sw_i : Te_i \times Te_i \rightarrow \{0; 1\}$, the weighted score function at iteration i, as the following quantity:*

$$sw_i(x_p, x_q) = \begin{cases} a_i, & \text{if } c_i(x_p) = c_i(x_q) \\ b_i, & \text{if } c_i(x_p) \neq c_i(x_q) \end{cases}$$

In particular, these scores are dependent on the iteration, thus can reflect the fact that some iterations are more informative than others. For instance, having two samples grouped together at an iteration with 100 synthetic labels is more meaningful than if there were only 2 possible synthetic classes. Since the number of synthetic classes and other training settings vary over the iterations and may have a strong influence on the classifier results, it is relevant to incorporate this information in the scores. Furthermore, this allows us to distinguish the pairs of entities that are not both in test set Te_i (no update at iteration i), from the pairs of samples that are both in Te_i but in different classes (update b_i). In the previous definition, both cases were considered equivalent and led to no update of the score ($b_i = 0$).

2.3.2 Modeling the similarity between dissimilar samples

With the current score definition, the similarity s does not fit any known probabilistic model anymore. However, it is possible to choose the weight values a_i and b_i in order to partially constrain the similarity distribution, and obtain a model that we can manipulate.

Approximation by a Gaussian distribution. Since we only have two free parameters, we choose to constrain the expectation and standard deviation for the similar distribution for dissimilar samples, denoted by $S_\perp$. The underlying idea is that if we manage to obtain a simple model for $S_\perp$ then we can use it to identify which pairs may correspond to dissimilar samples. We choose a_i and b_i such that $S_{\perp i}$ has a fixed expectation μ and variance σ^2. $S_{\perp i}$ is a Bernoulli-like random variable which takes the value a_i with probability $p_{0,i} = \mathbb{P}(c_i(x_p) = c_i(x_q) \mid x_p \perp x_q)$ and b_i otherwise, hence:

$$\left\{\begin{array}{rl} \mathbb{E}(S_{\perp i}) = & a_i p_{0,i} + b_i(1-p_{0,i}) = \mu \\ \mathbb{V}(S_{\perp i}) = & a_i^2 p_{0,i} + b_i^2(1-p_{0,i}) - \mu^2 = \sigma^2 \end{array}\right. \Leftrightarrow \left\{\begin{array}{rl} b_i = & \mu - \frac{p_{0,i}}{1-p_{0,i}} a_i \\ 0 = & a_i^2 - 2\mu(1-p_{0,i})a_i - \frac{(1-p_{0,i})(\mu^2 p_{0,i}+\sigma^2)}{p_{0,i}} \end{array}\right.$$

$$\Leftrightarrow a_i = \mu(1-p_{0,i}) + \varepsilon\sqrt{\frac{1-p_{0,i}}{p_{0,i}}(\sigma^2 + \mu^2 p_{0,i}(2-p_{0,i})))}, \qquad \text{where } \varepsilon = \pm 1$$

$$b_i = \mu(1-p_{0,i}) - \varepsilon\sqrt{\frac{p_{0,i}}{1-p_{0,i}}(\sigma^2 + \mu^2 p_{0,i}(2-p_{0,i}))}$$

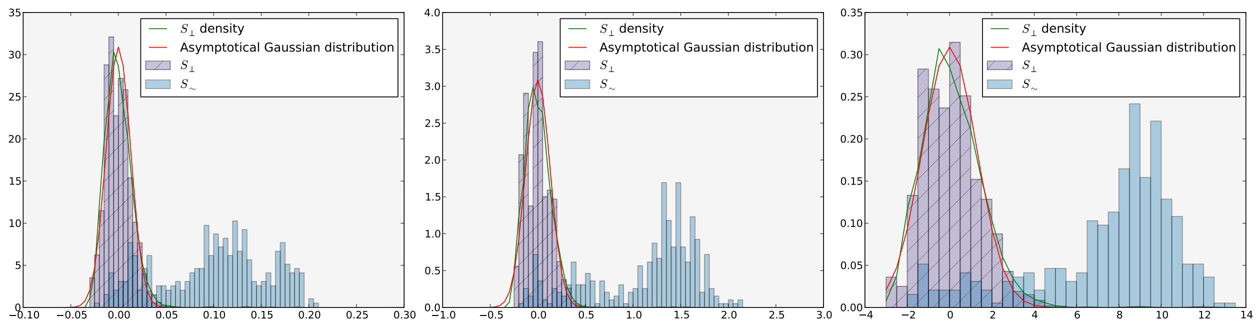
To simplify, we take $\mu = 0$, and we choose ε such that a_i is positive (hence b_i negative) since a high positive value of S means the samples are similar. This results in:

$$a_i = +\sigma\sqrt{\frac{1-p_{0,i}}{p_{0,i}}} \quad \text{and} \quad b_i = -\sigma\sqrt{\frac{p_{0,i}}{1-p_{0,i}}}$$

The variance parameter, σ, reflects the trust we put in the classifier: The greater σ, the greater the difference between the reward and penalty scores.

The full similarity between dissimilar samples, $S_\perp$ is therefore a sum of Bernoulli-like variables with outputs a_i (success) and b_i (failure), expectation 0 and variance σ. It is possible to show that $S_\perp(x_p, x_q)$ converges in distribution towards a normal law $\mathcal{N}(0, \frac{\sigma^2}{T(x_p,x_q)})$ using a generalized version of the central limit theorem. See Appendix B for a proof.

Experimental validation. We conduct experiments on the weighted similarity over 50 classification iterations. To compute the scores a_i and b_i, we estimate the value of $p_{0,i}$ using the independence assumption introduced in Subsection 2.1.3. In Figure 8 we plot the similarity distribution histogram obtained for a given sample x, i.e., one line of the matrix, with the weighted scores variant. In green is the distribution fitting this histogram (restrained to the samples y such that $x \perp y$), estimated with the `scipy` library. Finally, we plot in red the Gaussian Law towards which $S_\perp$ converges in distribution, as explained in the previous paragraph.



(a) $\sigma = 0.1$ (b) $\sigma = 1$ (c) $\sigma = 10$

Figure 8: Weighted similarity distribution (for sample 2610 - président français of class fonc) for different values of the variance σ, over 50 classification iterations.

Despite the low number of iterations, we observe that the practical distribution of $S_\perp$ (green) is indeed close to its asymptotic Gaussian distribution (red). Using classic properties of Gaussian distributions, we can now find a threshold of "confidence", Z, on the similarity $S(x_p, x_q)$ such that when $S(x_p, x_q)$ is greater than Z we can assess with a certain confidence that $x_p \sim x_q$. Formally, Z should be such that the risk $\mathbb{P}(S_\perp > Z) = \alpha$ is low, i.e., using the Gaussian approximation of $S_\perp$, $\mathbb{P}(X \sim \mathcal{N}(0,1) > \frac{\sqrt{T(x_p,x_q)}}{\sigma} Z) = \alpha$ is low. For instance for a risk of $\alpha = 0.05$, the standard normal distribution tables (or Z tables) tells us that we have to take $Z = 1.65 \frac{\sigma}{\sqrt{T(x_p,x_q)}}$. With this reasoning, we optimize the *precision* of the retrieval (if we only keep the pairs of samples with a similarity above Z, they should only contain a proportion α of the truly dissimilar pairs).

Reciprocally we could have led the same reasoning throughout the whole section using the distribution for similar pairs of samples, $S_\sim$, instead. In this case, we could have found a threshold such that $\mathbb{P}(S_\sim > Z) = \beta$ is high, which corresponds to the notion of *recall* (if we keep the pairs of samples with a similarity above Z, then we should retrieve a proportion β of the truly similar pairs). Theoretically, the same kind of criterion could be obtained for the previous Poisson-Binomial model, however the study would be more complicated than for a Gaussian distribution. In particular, the Poisson Binomial distribution depends on the classifier and training settings because of the p_i parameters, thus yields a different model for each run of the experiments.

In the next section, we report results on three experimental settings for applications on clustering and semantic neighbors retrieval. In the reported results, we only consider SIC with binary scores, as we did not have enough time to fully explore the EM similarity and weighted scores variants.

3 Experiments on knowledge discovery in multimedia content

In order to evaluate the proposed similarity construction, we use it as input of several knowledge discovery tasks and compare the obtained results to the baseline algorithms. As we already mentioned, we considered two clustering tasks based respectively on text and audio content, so we can study how SIC behaves with different types of data. Additionally, we exploited SIC as input of a problem of nearest-neighbor retrieval. This task was more challenging than the two others due to the very large scale and heterogeneity of the dataset, and was the occasion to explore a different experimental setting. In our experiments we mainly focus on applications to clustering of multimedia content, however the similarity matrix itself is not constrained to this type of tasks and could be applied on different domains and problems.

3.1 Unsupervised named entity recognition

Given a set of natural language sentences, for which we assume the locations of the named entities are known, the task of *unsupervised named entity recognition* deals with discovering possible structural similarities between those entities in an unsupervised fashion. The task as well as the dataset used were introduced in Subsection 1.2.3. Note that the proposed approached has already been applied in this framework [**Claveau and Ncibi, 2014**], hence we mainly used this task as a mean to evaluate and compare the different approaches explored during the internship.

3.1.1 Diverting Conditional Random Fields

Classifier and Features. In this case, we exploit Conditional Random Field classifiers in combination with SIC to build a similarity. In practice we work with the Wapiti toolkit for an efficient implementation of CRFs [**Lavergne et al., 2010**]. In terms of features, the original dataset, NER, provides 3 common categorical features for each word of the original text document: the word itself, the part-of-speech (PoS: grammatical nature of the word) and the BIO tag (indicates the position of the named entities). This description also conforms to the one of the baseline approach. However, the use of CRFs allows us to easily incorporate more complex feature representations, without having to explicit which role each feature plays in the definition of the similarity. Therefore we also consider an enriched version of the dataset, NER^{rich}, containing 5 additional features indicating if the word belongs to a certain lexical field based on a fixed prior dictionary (e.g., is it a currency, a town's name, a person's firstname . . .). This is closer to a semi-supervised setting than a fully unsupervised one as these new features sometimes provide indirect yet meaningful knowledge about the ground-truth class (Note that the experiments in the previous section were conducted on NER^{rich}).

Other training settings. At each iteration we use 5% of the dataset for training (which on average amounts to 280 named entities for 200 sentences). The rest of the data ($\sim$ 5000 entities and 3600 sentences) is used for testing. When possible it is in fact preferable to choose a low number of training entities (as long as it does not harm the learning process), since it allows for larger test sets. In fact, the more samples are compared with one another at each round, the less iterations will be needed to see each sample a satisfying number of times each at test time.

Once the training data is fixed, we generate the corresponding synthetic annotation. Since in practice we are only interested in the co-classification of named entities, we annotate every other word with a "`null`" label. Similarly, when an entity spans over several words (e.g., "président de la république"), we consider that the class of the entity will be the class given to its first word, and we assign a dummy label "`in`" to the extra words. The `null` label is enforced on the test set too in order to avoid useless computations (no need for the classifier to label a test sample which is not a named entity). Lastly we assign a uniformly randomly chosen synthetic label to each remaining word. See Figure 9 for an example of synthetic annotation.

Since we work on text data, we also introduce a bias in the annotation, by forcing the same synthetic label to all occurrences of the same named entity in the training set. While identical entities do not always share the same ground-truth class (e.g., names of countries are classified as locations or organizations depending on the context of the sentence), it is often true in practice (e.g., "aujourd'hui" is always a temporal indication). This leads to faster convergence because this bias heavily influences the learning step, but also to fewer mistakes for significant markers such as persons names or temporal indications for which the assumption is indeed true.

The last training parameter is the number of synthetic labels, which is a parameter difficult to optimize. First of all, it is desirable to have a sufficient number of samples in each training class

le	**tour**	**de**	**France**	quatrième	étape	remportée	**aujourd'hui**	par
O-null	B-fake11	I-in	B-fake188	O-null	O-null	O-null	B-fake109	O-null
le	norvégien	**Thor**	**Hushovd**					
O-null	O-null	B-fake65	I-in					

(a) Fake annotation on a full sentence of the *training* set

les	technologies	de	l'	information	sont	incontournables	**aujourd'hui**	a
O-null	O-null	O-null	O-null	O-null	O-null	O-null	??	O-null
admis	le	**président**	de	**Microsoft**	**Afrique**			
O-null	O-null	??	O-null	??	??			

(b) Enforced annotation of non-interesting samples on a full sentence of the *test* set

Figure 9: Example of synthetic annotation generated for one iteration

for the learning phase. In this case, the number of synthetic classes is constrained by the size of the training base. We also have to choose a "reasonable" number of classes so that the training phase is not too computationally expensive. Secondly, this parameter influences how fine-grained the resulting similarity is, and should be chosen accordingly. In fact, the more classes there are, the smaller and the more specific they tend to be. A third important point is the distinction between the *possible* number of synthetic classes, the *actual* number of training classes and the number of classes appearing at *testing time*: For instance, for the experiments on this dataset we take a fairly large number of possible synthetic classes ($\simeq 300$ classes for 280 named entities). Obviously not all of them are used because of the random annotation process, which results in about 170 training classes on average. Among those classes, some of them are too specific or correspond to very rare cases, hence their model do not fit any test samples. Consequently we usually only obtain between 50 and 80 classes in the test set, which is much lower than the initial 300 classes.

To conclude, the number of possible synthetic classes at training time should be chosen accordingly with the expected granularity of the cluster and the size of the training database. It is usually good practice to observe the distribution of the test samples among the classes at testing time to determine whether the granularity is too coarse or too fine. However, this parameter usually loses of its influence past a certain threshold, as even if the number of training classes grows, more of them will be dismissed as "too specific" at testing time, hence the number of classes for testing will stay approximately the same.

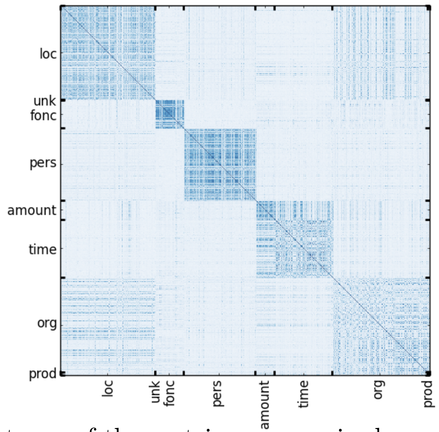
3.1.2 Results

We build a similarity matrix using SIC (with binary scores) over $N = 150$ classification iterations (as we will show in Subsection 4.1.2 about convergence issues, this number is sufficient to obtain satisfying stability of the similarity matrix). At each iteration, 5% of the database is used for training, 95% for testing and there are 300 possible synthetic training classes. This matrix is then used as input of the MCL (*Markov Clustering*) algorithm. We choose to use MCL because of the few prior parameters required and its efficient implementation[1].

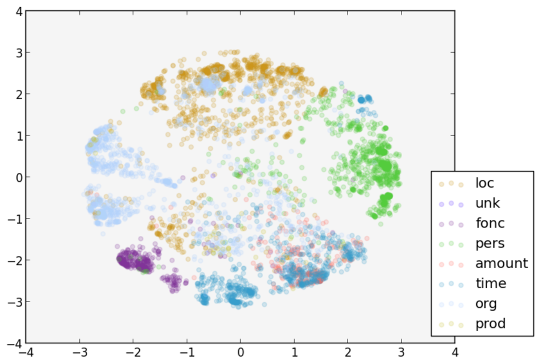
In Figure 10 we present two visual representations of the matrix. The first one is a heatmap (a dark tone indicates a higher similarity) where the samples have been reorganized according to their ground-truth cluster. This can also be seen as a fine-grained confusion matrix between the ground-truth clusters. The second plot is a 2D-projection of the samples, constrained by the obtained

[1] MCL - a cluster algorithm for graphs, http://micans.org/mcl/

similarity, using the Multidimensional Scaling algorithm. The samples are colored according to their ground-truth clustering. Both representations yield the same conclusions: The "person" (green cluster) and "function" (purple cluster) are less likely to be confused with any other class. On the contrary, the "time" and "amount", as well as the "organization" and "location" classes are often mixed together (respectively dark blue and red clusters, and light blue and yellow clusters).



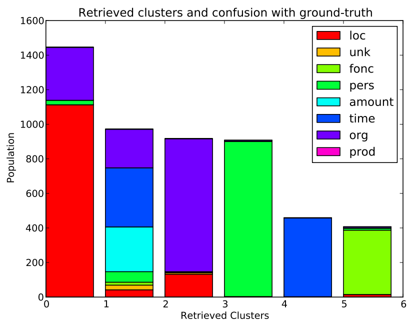
(a) Heatmap of the matrix, re-organized according to the ground-truth clustering.



(b) Projection of the similarity matrix on $\mathbb{R}^2$ using the Multidimensional Scaling (MDS) algorithm.

Figure 10: Two visual representations of the final similarity matrix

Settings / Evaluation	Base. NER	SIC NER	SIC NER$^{\text{rich}}$
Adjusted Rand Index	12.93	20.59	**53.54**
V-measure	-	30.71	62.86
Normalized Mutual Info	-	25.81	62.86
Adjusted Purity	-	51.07	73.63
Number of clusters	-	5	6



(a) Clustering numerical results (left) and final clustering structure (right) using SIC + NER$^{\text{rich}}$

Class	Entities	mAP (NER)	mAP (NER$^{\text{rich}}$)
Fonction	390	28.34	86.53
Person	996	45.36	80.99
Location	1303	42.88	66.53
Time	801	38.38	55.75
Organization	1312	30.99	51.66
Amount	266	24.77	37.88
Product	38	2.51	8.65
Unk	6	0.45	0.26
Total	5112	37.21	**63.37**

(b) Detailed mAP results for SIC on the two NER datasets

Figure 11: Clustering and retrieval results for SIC on the named entity recognition task

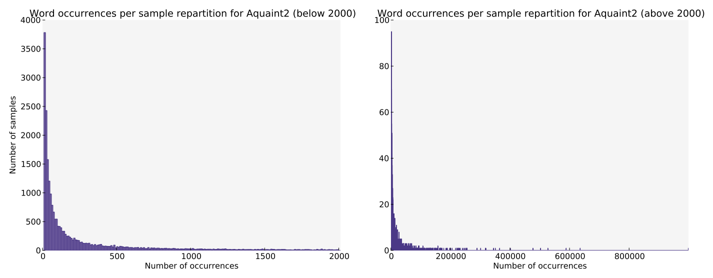
In Figure 11, we report numerical results for several clustering and nearest neighbor retrieval metrics that we introduced in Subsection 1.1.3. All of them are scaled to range between 0 and 100, and a higher value indicates better performance. On the simple NER dataset, with similar feature representations, SIC improves the baseline results, which shows that the similarity construction benefits from the use of CRFs. On the NER$^{\text{rich}}$ database, SIC clearly outperforms the previous results, which was expected, since the additional features provide meaningful information for the task. In particular, the greatest improvements in terms of mAP are obtained for the classes “function” and “person”, for which the additional information is especially relevant (e.g. one of the new attributes indicates whether the word is a firstname) or not. In this case, the main advantage of SIC is that we did not have to manually combine the different features and investigate how they would influence the similarity, as it was managed internally by the classifier.

3.2 Retrieving semantic proximity

The second experimental settings we investigate is the task of semantic relation discovery on the Aquaint2 dataset[2]. The goal is to build a similarity between words which reflect semantic information (e.g., synonymy, hypernomy . . .). The result is evaluated as a neighbor-retrieval task using ground-truth neighbors lists from common word thesauri such as WordNet[3] or Moby[4]. See Figure 22 in the Appendix for an excerpt of the synonymy relations graph.

3.2.1 The Aquaint2 dataset

The Aquaint2 dataset is a large collection of English news articles from various sources. The task samples are all common names occurring strictly more than 10 times in the dataset, which amounts to 26226 words each occurring between 11 and 999210 times in the dataset (see Figure 12). Each word is also provided with 3 categorical features: the word itself (with original formatting, e.g., capitalized letters), its lemmatized version and PoS-tag information.



(a) Distribution of words’ occurrences in the Aquaint2 dataset (below and above 2000 occurrences)

- 825 148 text documents;
- 15 718 376 sentences;
- 42960 unique common names, among which 26226 occur strictly more than 10 times;
- Between 11 occurrences (e.g., “zephyr”) and 999210 (“year”).

(b) Statistics on the dataset

Figure 12: The Aquaint2 information retrieval database

3.2.2 Tuning SIC for Aquaint2

Issues. Compared with the previous dataset, the Aquaint2 dataset presents 3 main problems: first, its large scale, which requires computational adjustments (e.g., the data must be loaded on the

[2] AQUAINT-2 Information-Retrieval Text Research Collection, https://catalog.ldc.upenn.edu/LDC2008T25
[3] WordNet, https://wordnet.princeton.edu/ [4] moby thesaurus, http://moby-thesaurus.org/

fly at each iteration, the full dataset can not be used as test set...). Secondly, each sample occurs several times in the dataset, and the number of occurrences is very heterogeneous as was shown in the previous histograms. This introduces an undesirable bias as samples with many occurrences are more likely to appear at training and test time. Furthermore, each sample can now appear in different classes at test time which we also did not take into account previously. To alleviate the problem, we propose a slightly modified formulation of SIC in Equation (8). In this formulation, we normalize the binary score by the number of times a sample appears in a given class given the total number of times it appears in the testing set:

- $\{x_1, \dots x_D\}$ denotes the *unique* samples;
- $O(x_i)$ denotes the set of all occurrences of the noun x_i;
- For a given set A, the notation $x_i \in A$ means $O(x_i) \cap A \neq \varnothing$;
- $\{w_1^i, \dots w_{n_i}^i\}$ are the classes in the testing set at iteration i.

$$s_i(x_p, x_q) = \mathbb{1}_{\{c_i(x_p)=c_i(x_q)\}} \sum_{k=1}^{n_i} \mathbb{1}_{\{(x_p \text{ and } x_q) \in w_k^i\}} \frac{|O(x_p) \cap w_k^i|}{|O(x_p) \cap Te_i|} \times \frac{|O(x_q) \cap w_k^i|}{|O(x_q) \cap Te_i|} \tag{8}$$

The last issue is that the ground-truth list of neighbors are a lot more fine-grained than the ground-truth clustering in the named entity recognition task. In fact, we have as many clusters as there are entities (26226), and each cluster contains between 3 and 50 entities (38 on average). As we mentioned for the previous dataset, the granularity of the obtained similarity is controlled by the number of synthetic labels. However, we are also constrained by the training speed of the classifier and in this case, it is not reasonable to generate enough labels to match this granularity.

Training settings. As we are working with the same type of text data as NER, the annotation settings for this dataset are the same as before. Due to the large scale of the database, we only use partial subsets for the training set (∼2000 sentences, 8024 samples occurrences) and testing set (∼50000 sentences, 20000 samples occurrences) at each iteration. Finally, we sample the number of synthetic labels in the range [200;400] at each iteration.

For computation sake, we cannot use more synthetic labels to fully represent the granularity of the ground-truth clustering. Therefore we introduce a "One versus all" (OVA) learning setting, which avoids this problem but usually requires a higher total number of iterations in return. Instead of using SIC to build the whole similarity matrix, the idea is to build it line per line (i.e., sample per sample). For each sample x we run the usual SIC, however we only aim to separate the samples similar to x from the others. Hence we only use three training classes: `class1` contains all occurrences of x (positive examples) occurring in the training set, `class2` is for all other Aquaint2 samples which are not x (negative examples), and `class3` contains all the other common words. Then at testing time, every sample falling into `class1` is considered similar to x and receives a positive update. This allows us to build the partial similarity $s(x, \cdot)$, and the process is repeated for each sample to build the whole matrix (Note that with this process the similarity matrix may not be symmetric hence an additional symmetrization step is required).

3.2.3 Results

In Table 1, we report results for SIC (1000 iterations with the number of synthetic labels randomly sampled from range [200; 400] at each iteration) and for SIC with OVA setting (150 iterations for each sample). Note that because of the limited time, we could not run OVA for all the 26226 samples. Hence the results are reported only for a subset of them ($\sim$ 500 random samples). Finally, for comparison we also report baseline results from [**Claveau et al., 2014**] and [**Ferret, 2013**].

Eval. \ Settings	Ferret2013	Claveau2014	SIC	SIC + OVA
mAP	5.6	8.97	0.24	1.83
P@1	22.5	31.05	0.17	6.75
P@10	10.8	13.76	0.14	3.84
P@100	3.8	4.54	0.15	1.90

Table 1: Results for nearest neighbour retrieval task on the Aquaint2 dataset

A first general observation is that all methods yield a low mAP. This is due to the difficulty of the task (in terms of scale, granularity…), but also to the fact that the ground-truth neighbors list derives from independent thesauri, which may not reflect the same semantic contexts as the dataset.

SIC clearly performs worse than the two baseline methods on this task. The disparity in terms of words co-occurrences seems to be the cause: In fact, the baseline algorithms assign each sample to a global internal representation gathering all of its occurrences, and then compute the similarities based on these unique representations: All samples are "on an equal footing" in terms of number of comparisons. On the contrary, SIC works at a lower level, with the original samples. In particular if a word occurs very rarely in the dataset, it also seldom appears in the experiments. Consequently, some pairs of samples are rarely (or worse, never, which automatically yields a similarity of 0) compared together because they seldom appear in the same test set. This unbalance is a strong bias in disfavor of uncommon words. Furthermore, the same problem is probably present in the training set, which affects the learning process. To enforce more homogeneity, we could constraint the number of occurrences of each sample appearing in the test/training sets. However, it would currently be time-consuming to retrieve all occurrences of a sample, as they are scattered across the whole database. A better organization of the dataset, allowing us to quickly retrieve all occurrences of the same word, is required before efficiently implementing this solution.

Secondly, the fine granularity of the clusters could be a problem as increasing the number of synthetic labels would harm the computational efficiency. The OVA scheme was designed to be a bypass of this problem, and it in fact yields slightly better results, yet still low. The main problem here is the regularization of the classifier. In fact, we have to balance the number of positive and negative examples in the training set to avoid overfitting or overgeneralization towards one or the other class. In this case, it appears CRFs are not the best choice as it is difficult to tune their regularization process well, especially since it most likely depends on the considered sample, its number of occurrences…. A possible alternative is the use of decision trees. Because of the tree structure, the regularization is controlled by the pruning level (whether we look at the leaves or higher nodes) which seems more intuitive to tune.

3.3 Audio motif discovery

The last task deals with the problem of *audio motif discovery*. More specifically, our goal is to group together occurrences of the same words given a set of audio samples extracted from natural speech in different speaking conditions.

3.3.1 Supervised Speech Recognition

Presentation of the task. Audio motif discovery is the problem of extracting recurrent patterns in audio signals. These patterns are called *motifs*, analogously to the fields of bioinformatics and genomics, and from a discovery point of view, they carry meaningful information about the data. For instance, in real-life audio signals (e.g., conversations, radio shows), recurring motifs are often linked to the topic of the content, thus may be used for automatic audio summary, or indexation by keywords. This is also an unsupervised framework, as we do not know beforehand what kind of motifs to search for, or their number of occurrences. Furthermore, we are interested in working directly on the audio signals, rather than on a textual representation. This preserves acoustic information (which may be harmful or beneficial depending on the situation), and saves us the effort of translating the audio to another representation.

The task is usually broken down in two steps: the first one is to find the audio pieces that are actually repeated: Those are the motif candidates. The second step is to group these pieces, such that two of them falling in the same cluster means they are occurrences of the same motif. As for the named entity problem, we focus on the second step, which is to assess if two audio signals are occurrences of the same motif. However, the data domain is different from the problem of labeling named entities, and CRFs are not commonly used for audio frameworks. Instead, we propose to use *H*idden *M*arkov *M*odels (*HMM*), often used for speech recognition.

Baseline algorithm. The baseline algorithm for comparing two audio signals is DTW. Among others, it is used for the task of Audio motif discovery in [Park and Glass, 2008] and [Muscariello et al., 2009]. Given two audio samples represented by a sequence of real-valued vectors features (each vector represents a single time unit in the discretized signal), the algorithm aims at finding the best alignment between the samples, based on a loss function (usually, Euclidean distance between their vector representation), and then outputs the loss value in their best alignment. It is also common to add geometrical constraints on this process. For instance, the alignment search can be constrained to only match audio units that are not too far away in time. This is to prevent the alignment path to drift towards too high distortions. Similarly, the extremities of the samples are usually constrained to match, but this assumption is sometimes relieved.

While DTW is very efficient to compare audio samples with temporal distortions, it does not take into account acoustic information, such as the speaking environment (radio studio, outdoors, female/male speaker…). For this reason, DTW is suited for same-speaker speech recognition but performs poorly when the speaking environment varies, which is the case in our dataset. On the contrary, HMM classifiers are not as sensitive to this parameters, and we can expect the SIC similarity to inherit this property.

Hidden Markov Models. For this task we combine SIC with HMM classifiers, which are commonly used for audio processing and speech recognition. A HMM is an *emitting* state machine with two main parameters: the transition matrix (probability of jumping from one state to another) and the emission probability (probability that a given states emits a symbol from a given dictionary). Typically, the states are said to be "hidden", and only the emitted symbols (also called observations) are visible. In our case, the dictionary is the set of all sounds in the spoken language, each state represents one time unit in the discretized signal, and the transitions represents the flow of time.

Given a set of training sequences, the HMM is able to learn parameters for a model of these sequences using the Baum Welch algorithm. Then given an input test sample, the HMM is able to find the most likely state sequences (with a likelihood score) corresponding to this word utterance using the Viterbi algorithm. Finally, in practice we actually use several small HMMs for our task: Each HMM represents the model of one of the training classes. When a new input sample arrives, it is run through all HMMs and the algorithm outputs the class corresponding to the HMM returning the path with the highest likelihood. In practice, we use the HTK library in the implementation, for both the features extraction and the HMM training and labeling.

3.3.2 The (audio) ESTER2 dataset

We assume the motif candidates are already extracted from the audio signal, hence we only focus on building a similarity between the samples. In our experiments, we use the audio counterpart of the ESTER2 dataset. It contains about 8 hours of continuous speech extracted from several French radio shows (africa1, rfi...). There is a high variability between different occurrences of a same word: different speakers (male/female, low/high pitch...) and different environments (radio studio, phone conversation, outdoors ...). This causes the usual DTW approach to perform poorly as mentioned previously, and motivates the use of HMM classifiers to build a similarity.

To build the dataset, we first extract every word occurring at least ten times. The ground-truth clusters correspond to all occurrences of a same word. Hence there are as many ground-truth clusters than unique words in the dataset. For small experiments, we consider a subset of those samples, AUDIO$^{\text{tiny}}$, which contains 19 unique words for 50 occurrences each, amounting to 950 samples (19 words “Afrique”, “autres”, “comme”, “contre”, “depuis”, “dernier”, “encore”, “France”, “gouvernement”, “ministre”, “Nicolas”, “notamment”, “parce que”, “pays”, “place”, “premier”, “Sarkozy”, “toujours”, “ville”). The selected samples provide sufficient variability in the dataset, while keeping some similar words that could confuse the classifiers (e.g., “autres”, “contres”). For bigger experiments we enforce an additional time constraint, as using too short samples would restrain the size of our HMMs (as we need enough content to optimize all the parameters) and would harm their learning abilities. We first remove most of the words that are obviously too short (e.g., monosyllabic words), while retaining some of them to prevent having a too biased evaluation framework (e.g., “il”, “pour”). Secondly we remove all samples with a length inferior to 0.2 seconds. In the end, the full audio dataset, AUDIO, contains 13477 samples for 594 ground-truth clusters. The number of occurrences per unique words ranges between 5 and 267 (see Figure 13). We also introduce a second ground-truth clustering which takes homonyms into accounts (i.e., words pronounced the same way are considered identical). In this case we have 543 ground-truth clusters.

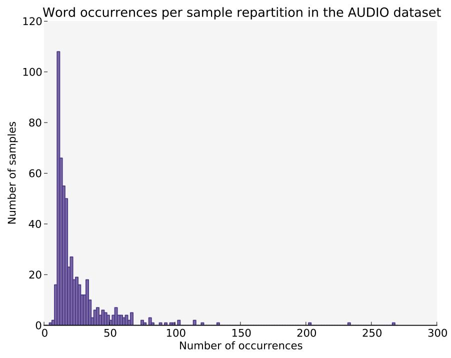


Figure 13: Occurrences per unique word repartition in the AUDIO dataset

Experimental settings. As the dataset originally only provides the audio signals, we first have to extract features to describe the data. We consider four types of audio features: MFCC (Mel-Frequency Cepstral Coefficients) and PLP (Perceptual Linear Prediction), which both show good and comparable performance for speech recognition, as well as LPC (Linear predictive analysis) and LPCEPSTRA (LPC cepstral coefficients), usually not as efficient as the previous two [**Mporas et al., 2007**]. Furthermore, each type of features can be divided into several sets of coefficients (e.g., A: acceleration coefficients, D: delta coefficients, 0: absolute energy ...), and HTK allows us to choose which combination of coefficients to extract. For DTW we use MFCC features (state-of-the-art for speech recognition) with all coefficients except the first one (energy of the signal). This makes the signal more robust to signal amplitude variation. For SIC we will exploit and compare the four types of features. More precisely, at each iteration we first sample one possible type of features (MFCC, PLP, LPCESPTRA, LPC) then we randomly sample a coefficient combination. We then generate the corresponding features and use it for the current iteration.

For the synthetic annotation, we simply stick to a random annotation of the samples in the training set. However, HTK usually requires a minimum number of training samples to learn a HMM (proportionally to the number of parameters there are in the model). For this reason, we also enforce that there is a minimum number of samples assigned per synthetic labels; Any training class failing to meet this requirement is discarded. In order to have enough samples for this restriction, we usually take 25% of the dataset for training for AUDIO$^{\text{tiny}}$ and 40% for AUDIO. The number of synthetic labels used will be precised for each setting in the rest section. As mentioned previously, it is chosen according to the granularity of the target ground-truth clustering. Finally, we once again use SIC as input of the MCL algorithm for the clustering part.

3.3.3 Results

AUDIO$^{\text{tiny}}$. We first present results on the small audio dataset. The dataset contains 950 samples, 25% were used for training at each iteration (with least 10 samples per class). We considered 20 to 40 synthetic training classes, and we used HMM of 7 states (which is the maximum we can use as we are limited by the length of short samples). We first report in Table 2 the results obtained by SIC alone restrained to some combinations of features (i.e., at each iteration we sample one type of features among the ones given and then we randomly sample a subset of coefficients as usual) using 500 iterations for each. Note that for the clustering, we usually tune the parameters so that we obtain a number of clusters close to the ground-truth one (19, here), while maximizing the clustering metrics; hence the retrieved number of clusters is not necessarily the same for each setting.

Setting / Measure	MFCC	PLP	LPCEPSTRA	LPC	MFCC+PLP	MFCC+LPC	ALL
mAP	44.85	**45.43**	38.28	15.83	45.07	34.10	39.24
f@1	**80.53**	79.58	73.89	35.47	81.05	72.53	78.95
f@50	43.06	**43.61**	37.87	17.28	43.06	33.52	38.05
Adj. Rand Index	0.411	**0.421**	0.347	0.102	0.421	0.303	0.360
V-measure	**0.615**	0.608	0.556	0.275	0.623	0.508	0.573
Norm. Mutual Info	0.607	**0.614**	0.546	0.264	0.614	0.509	0.566
Adj. Purity	0.585	**0.617**	0.433	0.293	0.556	0.523	0.561
Number of clusters	23	24	20	23	21	24	24

Table 2: SIC Features influence on AUDIO$^{\text{tiny}}$

When considering only one type of features, we observe that the results are in accordance with the quality of the features. MFCC and PLP are commonly used for supervised speech recognition tasks and indeed yield better results for our task than LPCEPSTRA and LPC, which are usually not as suited for speech recognition. However there is too little difference between the MFCC and PLP results to assess if one is better than the other. Finally, when combining features, the results are close to the average of the different types of features considered. The good iterations with suitable features (e.g., MFCC) are penalized by the bad ones which use bad features (e.g., LPC), yielding average scores (MFCC+LPC).

Finally, in Table 3, we report results obtained for DTW (implemented using the dtw R package) with MFCC features. We compare them against SIC for 177 iterations, using MFCC only too.

Measure \ Setting	DTW	SIC
Adj. Rand Index	0.02	**0.408**
V-measure	0.243	**0.609**
Norm. Mutual Info	0.224	**0.602**
Adj. Purity	0.323	**0.549**
Cluster Number	19	22

Class	DTW	SIC
Afrique	16.57	50.23
autres	15.97	20.70
comme	18.17	34.54
contre	9.63	31.28
depuis	17.08	41.96
dernier	14.70	39.67
encore	14.99	37.95
France	25.44	65.69
gouvernement	16.98	45.75
ministre	16.65	72.91

Class	DTW	SIC
Nicolas	20.94	83.28
notamment	14.82	47.21
parce que	7.51	27.55
pays	23.88	29.83
place	20.29	47.23
premier	23.74	25.54
Sarkozy	32.95	84.24
toujours	17.56	31.88
ville	37.72	48.79
Total	19.25	**45.745**

Table 3: Clustering (left) and mAP (right) results for SIC on AUDIO$^{\text{tiny}}$

We observe that in this case, SIC clearly outperforms DTW. This was expected because, as already mentioned, DTW is not robust to speaking environment changes, while SIC should be because of the use of HMMs. Furthermore, when observing the similarity values obtained by DTW, we notice that the variance is really low, hence it is difficult for the clustering algorithm to find and separate clusters. Secondly, the detailed mAP results show that both methods yield better results for "specific" (or rare) words (e.g., "France", "Nicolas", "Sarkozy"...) while very common words such as "parce que" or "autres" usually have lower retrieval accuracy. A possible explanation is that specific words tend to appear in particular contexts (e.g., politics for "Nicolas" and "Sarkozy") and because of their rarity, when they appear once, they are more likely to appear again in the near future (which is known as the *burstiness* of words). Because of this, it is possible that these words display less variability in terms of speaking conditions and environments, which would explain the better results for SIC as well as DTW.

AUDIO. For the full AUDIO dataset, we use slightly different parameters to account for the granularity of the ground-truth clustering (number of synthetic labels sampled in the range [100;200] at each iteration, with a constraint of at least 20 training samples per class, and 40% of the dataset used for training and all the rest for testing).

The first results presented in Table 14 are a comparison of the different HMM topologies. In fact, contrary to AUDIO$^{\text{tiny}}$, the samples in the full dataset have a minimum length of 0.20s which allows us to use HMM of at most 20 states. We consider two HMM topologies: The first one, `type 1` is a basic linear HMM with loop transitions on each emitting state and only forward transitions from state n to $n+1$ for all n. The second one, `type 2` is `type 1` with additional skip transitions (i.e., direct transitions from state n to $n+2$ for all n). We also consider `type 1/2` which corresponds to randomly choosing one of the topology at each iteration. For space consideration we only report results against the ground-truth clustering that takes homonyms into account for this experiment.

All results were obtained for 2000 iterations of SIC (as in practice we observed that the similarity matrix was usually stable for all settings around this value).

Setting / Measure	Type 1/2 7 states	Type 1/2 10 st.	Type 1/2 12 st.	Type 1/2 14 st.	Type 1/2 20 st.	Type 1 random(7;20)	Type 2 20 st.
mAP	16.49	18.27	19.86	20.61	**20.63**	20.53	20.20
f@1	57.66	59.47	61.79	**62.86**	62.46	62.64	62.44
f@100	14.56	15.89	16.82	**17.28**	17.19	17.13	17.02
Adj. Rand Index	0.130	0.149	0.133	0.136	0.135	0.107	**0.153**
V-measure	0.597	0.612	0.616	0.619	**0.623**	0.619	0.621
Norm. Mutual Info	0.585	0.598	0.601	0.604	**0.608**	0.604	**0.608**
Adj. Purity	0.476	0.524	0.552	**0.556**	**0.556**	0.543	0.539
Number of clusters	547	540	544	542	540	546	545

Figure 14: Comparison of HMM topologies on the AUDIO dataset

For the mixed type `type 1/2`, we observe that the results improve when the number of states increases which makes sense since the HMMs are then able to build more precise models. However the improvement seems to reach a limit around 14∼15 states in the HMM. The fixed `type 1` with its number of states chosen at random at each iteration between 7 and 20 yields similar results. It is interesting to see that despite mixing low number of states with high ones, we still obtain results comparable to those obtained with only high number of states. Finally the fixed `type 2` also yields comparable mAP results despite having more flexibility in theory, however it is slightly better in terms of clustering evaluation (adjusted Rand Index).

To conclude on this dataset, we report in Table 4 the results obtained by DTW with MFCC features and by SIC (using the Type 1/2 topology with 14 states from the previous example). We consider both the case of the ground-truth with and without homonyms.

Settings / Evaluation	DTW	SIC	DTW	SIC
	no homonyms		with homonyms	
mAP	3.00	19.72	3.11	20.61
f@1	13.17	59.41	14.05	62.86
f@100	4.37	16.19	4.65	17.28
Adjusted Rand Index	0.002	0.135	0.003	0.135
V-measure	0.185	0.628	0.177	0.623
Normalized Mutual Info	0.161	0.614	0.154	0.608
Adjusted Purity	0.104	0.553	0.117	0.556
Cluster Number	597	589	542	542

Table 4: Clustering and retrieval results for SIC on the AUDIO dataset

As we already observed on the AUDIO$^{\text{tiny}}$ dataset, SIC performs better than the usual DTW on this task. The global results are also lower than for AUDIO$^{\text{tiny}}$ which is explained by the larger scale of the database which induces more variability, and in a lesser way, the heterogeneity in the size of the ground-truth clusters may also impact the results. Finally the results with ground-truth homonyms are slightly better than without, which is expected as they sound similar, hence it should be impossible for the classifiers (or a human being) to distinguish them without context.

3.4 Implementation

In this subsection, we give some insights on the implementation of SIC developed during the internship. Each iteration being independent from the others, it is the ideal situation for a parallel execution. We implemented the process in Python using the *multiprocessing* library. Each process runs one iteration and their attribution to each core is managed by the main process (see Figure 15).

The similarity matrix (more specifically, only the upper triangle of the matrix) is stored as a flat 1D-array and shared across the different processes for memory efficiency. To prevent any read/write conflict between the process, we split the array into n_locks cells, each managed by a lock. In practice only a few samples are classified together at each iteration (compared to the number of possible pairings in the dataset), hence only a few cases are to be updated. This statistical property should prevent having too many collisions of processes attempting to write in the same cell at the same time, and preserve a satisfying execution time. In terms of resources, we ran the NER experiments on a 24 cores, 64GB RAM machine and AUDIO^{tiny} on a 4 cores, 8GB RAM. For the bigger datasets, AUDIO and Aquaint2, we ran the experiments on the Igrida computing grid at Inria (`http://igrida.gforge.inria.fr/`), using 8 cores, 48GB RAM nodes.

In Table 5 we report the time and memory usages for each setting on 50 iterations. That the first line is the number of cores used for the classification iterations, not counting the main process. The *real time* is the observed elapsed time of the execution. The *(user+sys) time* is the total amount of CPU time (across all processes). Finally, the *max RAM* is the maximum amount of live memory required by the program during its execution. In practice, runs on AUDIO and Aquaint2 are slower than NER because the datasets are larger, and because they require more I/O operations. In fact, for Aquaint2, the data is read from the disk and loaded on-the-fly at each iteration. For AUDIO, the audio features are also generated on-the-fly, and HTK only seems to work with files as input.

Method / Measure	NER (20 cores)	Aquaint2 (7 cores)	AUDIO (7 cores)
Time (real)	6m 12.3s	7h 52mn 22.3s	1h 12mn 34.1s
Time (user+sys)	1h 38m 23.2s	22h 9mn 25.6s	4h 48mn 31.5s
Max RAM	5.31 Gi	34.24Gi	9.53 Gi

Table 5: Execution times and maximum RAM usage for 50 iterations

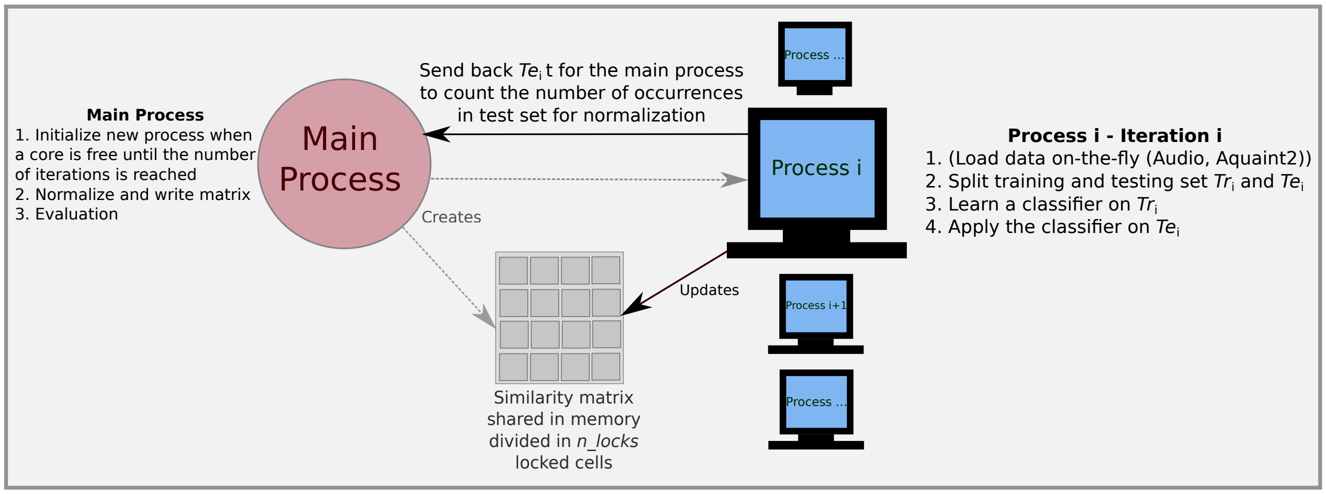

Main Process
1. Initialize new process when a core is free until the number of iterations is reached
2. Normalize and write matrix
3. Evaluation
Main Process
Send back Te_i t for the main process to count the number of occurrences in test set for normalization
Creates
Process ...
Process i
Process i+1
Process ...
Updates
Similarity matrix shared in memory divided in *n_locks* locked cells
Process i - Iteration i
1. (Load data on-the-fly (Audio, Aquaint2))
2. Split training and testing set Tr_i and Te_i
3. Learn a classifier on Tr_i
4. Apply the classifier on Te_i

Figure 15: Illustration of SIC running on one Igrida node

4 Additional properties of SIC

In this section, we propose a practical study of some additional properties of SIC. In the first subsection, we investigate the issue of the convergence of the method, and in the second subsection we define the notion of "confidence score" on the SIC similarity matrix.

4.1 Convergence Rate

Dealing with an iterative method raises the issue of its convergence, i.e., whether the method eventually reaches a stable point, and if yes, how fast, and towards what target value. In this case, the convergence itself and its speed are directly linked to the *consistency* of the classifiers. The more often the classifiers agree over a pair of samples, the quicker the corresponding similarity reaches a stable value. For this reason, it is important to randomly vary the training settings at each iteration, so that the convergence is not due to a bias in the classifiers' decisions. As for the quality of the obtained similarity matrix, it depends on the *accuracy* of the classifiers and data features, relatively to the target application. This property is however difficult to quantify as it is data- and application-dependent. See Section 3 for further discussions about the choice of classification algorithm and training settings. In this subsection, we estimate the convergence rate of the method in practice, and propose two stopping criteria based on this information. The first one is a posterior criterion obtained by studying the correlations between similarity matrices obtained at different iterations. Secondly, we introduce an on-the-fly criterion based on the notion of entropy, which can be used to stop the algorithm when a satisfying threshold is reached. We also study the influence of the threshold value over the quality of the similarity.

4.1.1 Global criterion

Let us consider a normal run of the algorithm over $N > 0$ iterations, from which we derive the similarity S^N. We aim to estimate how close S^N and S^X are (where S^X is the similarity obtained when interrupting the run after the $0 < X < N$ first iterations). To estimate the resemblance between the two similarity measures, we compute a correlation coefficient (e.g., Pearson) between the two corresponding (flattened) matrices. As usual, a value close to 1 (resp. -1) indicates a strong positive (resp. negative) relation between the observations, while a null coefficient indicates there is no correlation.

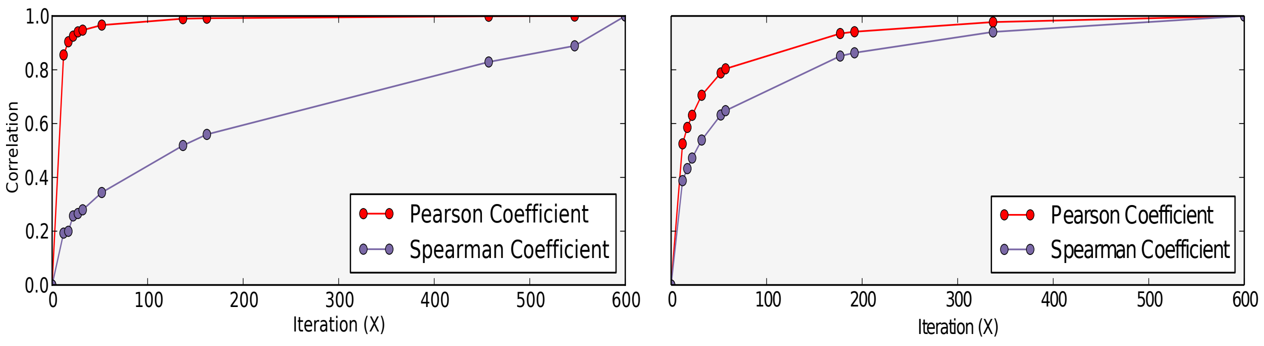
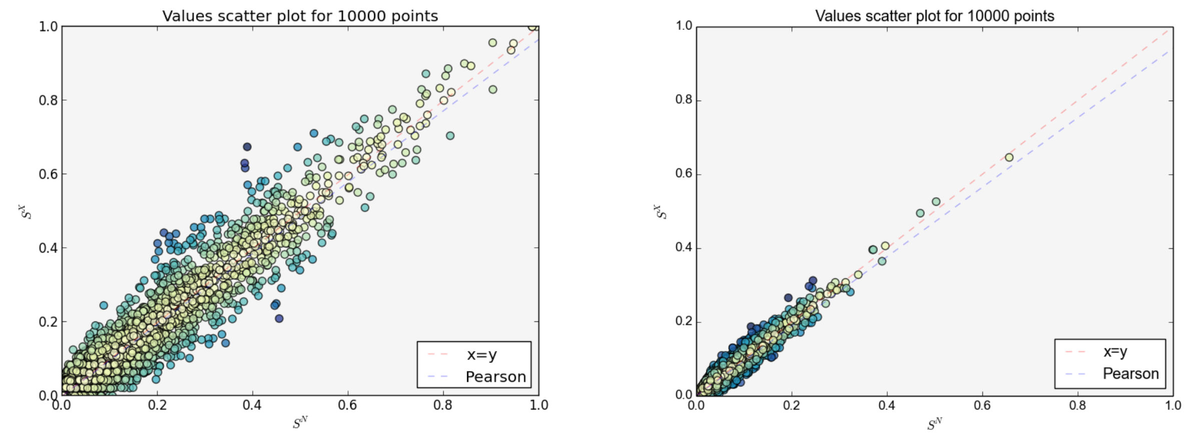


Figure 16: Evolution of the Pearson and Spearman correlations between S^X and S^N ($N = 600$) for the NER (left) and AUDIO$^{\text{tiny}}$ (right) datasets

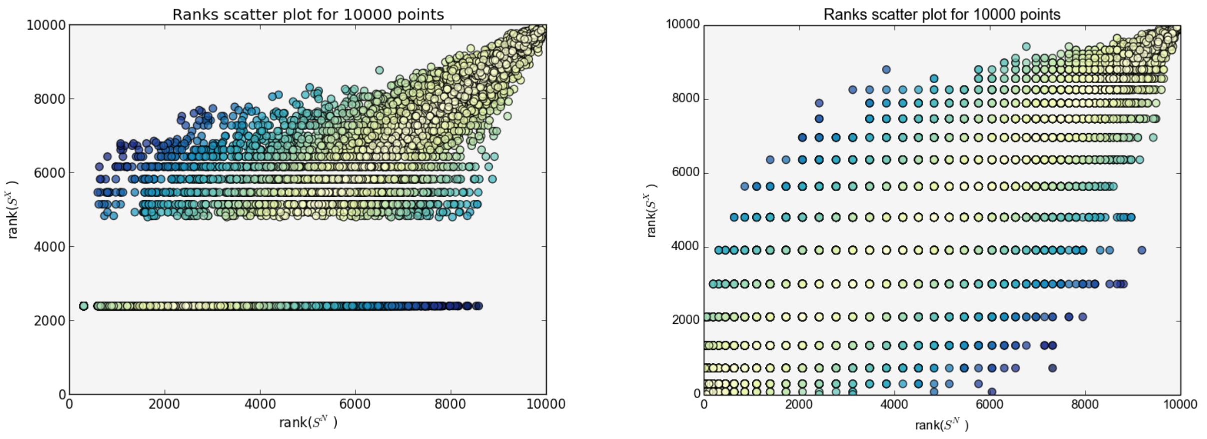
Figure 16 presents the the evolution of the Pearson and Spearman corelation coefficients ax X

grows for the NER and AUDIO$^{\text{tiny}}$ datasets. The *Pearson* correlation is only able to detect linear relations, and is sensible to outliers. On the contrary, the *Spearman* coefficient is based on the ranks of the observations (where the rank is averaged in case of tie situations) rather than their values. Both coefficients are commonly used to assess statistical correlation.

We observe that the Pearson correlation for NER reaches a satisfying target value close to 1 almost immediately ($X \simeq 50$), while it is slower for the audio dataset ($X \simeq 200$). We can verify that the correlation relation between the two similarity measures is indeed linear by plotting the two sets of observations against each other. In Figure 17a, we represent the points $(S^N(x_p, x_q), S^X(x_p, x_q))$ in a scatter plot for both datasets, and we clearly observe that the main axis of the scatter plot is close to the $x = y$ axis (which corresponds to a perfect Pearson correlation of 1) .



(a) Scatter plot of the values of S^N against S^X for 10.000 random pairs of samples for the text dataset (left, $X = 52$) and the audio dataset (right, $X = 192$). The points colors reflect their distance to the $x = y$ axis.



(b) Scatter plot of the *ranks* of the values with the same parameters as the above graphic

Figure 17: Scatter plots of the points (S^N, S^X) (top) and $(\text{rank}(S^N), \text{rank}(S^X))$ (bottom) for the NER (left) and AUDIO (right) datasets.

Surprisingly, compared with the Pearson coefficient, the Spearman coefficient for the text dataset grows very slowly, while it has the same monoticity as the Pearson for the audio dataset. A possible explanation lies in the typical distribution of the similarities for the text dataset. The distribution is often skewed, and the outliers might positively influence the Pearson correlation. Furthermore, in practice, we use more synthetic training classes for the text data and the classifiers used appear to efficiently detect dissimilar samples, hence a large set of samples are never classified together, resulting in many similarities close to 0, especially when we consider only a few iterations (low values of X). This results in a large disparity in the ranks, as shown in Figure 17b: We plot the same scatter plot as for the Pearson, but we use the *ranks* of the similarities (in increasing order) instead of their values (The Spearman coefficient can in fact be seen as a Pearson correlation between the ranks, hence we also would like to observe a main linear axis in these plots). The isolated line in the left graph (NER) represents all the pairs of samples which had a null similarity in S^X ($X = 52$ here). This discrepancy, which does not appear in the case of AUDIO$^{\text{tiny}}$ (right), probably negatively affects the Spearman coefficient for the NER dataset.

4.1.2 On-the-fly criterion

As mentioned in Subsection 2.1.1, $s(x_p, x_q)$ can be interpreted as a frequency which approximates the probability $\mathbb{P}(c(x_p) = c(x_q))$. We propose to use this approximation to compute the entropy of the binary random variable $X_{p,q}$ which takes the value 1 when $c(x_p) = c(x_q)$ and 0 else:

$$H(X^n_{p,q}) = -s^n(x_p, x_q)\log(s^n(x_p, x_q)) - (1 - s^n(x_p, x_q))\log(1 - s^n(x_p, x_q))$$

By definition, $H(X_{p,q})$ assesses how "unpredictable" the variable $X_{p,q}$ is. Low entropies correspond to pairs of samples which are very often, or very seldom, classified together, while the maximum entropy is reached when $X_{p,q}$ is uniform. Therefore, the entropy is initially null and grows as more information is brought at each iteration, eventually reaching a stable point when the information gain is negligible, i.e., new iterations do not change the similarity value very much. We propose to use the mean of the $H(X_{p,q})$ for all (p, q) as an on-the-fly stopping criterion, by ending the algorithm when this measure reaches a stable point. To evaluate the quality of this criterion, we run a normal SIC experiment over 600 iterations, and pause the algorithm whenever the mean of the entropies reaches some threshold. We then compute several evaluation measures introduced in the first section with the matrix at the current iteration and observe their evolution.

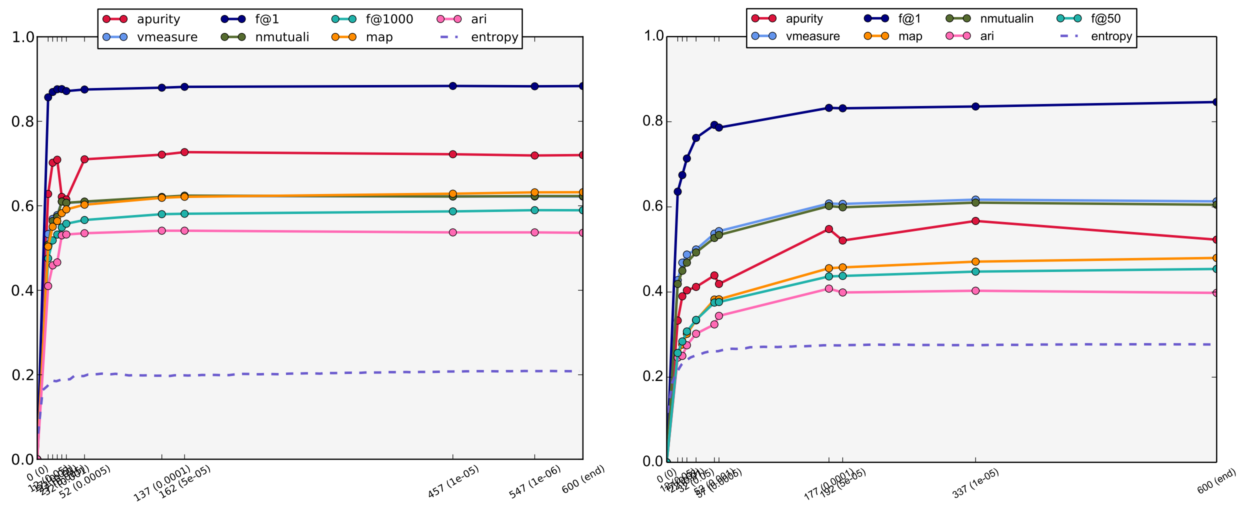


Figure 18: Evolution of the mean entropy criterion and the quality metrics over 600 iterations for the NER (left) and AUDIO$^{\text{tiny}}$ (right) dataset.

The corresponding results are presented in Figure 18. The dotted line represents the evolution of the stopping criterion over the iterations, i.e., the average of the entropies over all pairs of samples. In practice we compute the average entropy every 5 iterations, and mark the iteration (circle marker on the plot) if the difference with the previous measurement was lower than one of the thresholds we considered. Then, for each one of these marked iterations, we evaluate the similarity matrix against the ground-truth. Relatively to the clustering task, we report the adjusted Rand Index, adjusted purity, V-measure and normalized mutual information. The inconvenient of these measures is that they depend on the parameters used for the clustering algorithm. In this case, we tuned the parameters to obtain the same numbers of clusters for each iteration (6 for the text dataset, for 8 in the ground-truth and 22 for the audio one, for 19 in the ground-truth). This makes the different measures more comparable, however because of this constraint, the results reported are not necessarily optimal for each iteration. For this reason, we also report information retrieval results which are more objective since they do not depend on any parameters. We report the mAP, F-measure at 1 and F-measure at 1000 or 50 depending on the dataset (this number is chosen to match the size of the ground-truth clusters).

We first observe that the mean entropy behaves as expected: a fast growth in the beginning until it reaches a stable value. The same general observation applies for the different quality metrics. Secondly, the convergence is clearly faster for the text dataset than for the audio one, as it was already the case with the Pearson coefficient. This is mostly due to the text dataset having more informative features than the audio one for their respective task.

Finally, in Figure 19, we conduct the same experiment but only report the evolution of the mAP (every 5 iterations). The circles markers are the same as in the previous experiment. We also add a color overlay representing the areas where the mAP is less than a certain threshold different from the 'optimal value', i.e., the one found after $N = 600$ iterations, when the similarity matrix is usually stable. By matching the colored areas to the considered thresholds, we observe that a threshold of 0.0001 is a good compromise as it is reached in a low number of iterations and yields mAP values with only a 5 percent different from the optimal one. Using a threshold of $1e - 5$ allows for even better results (1% difference) in return for more iterations.

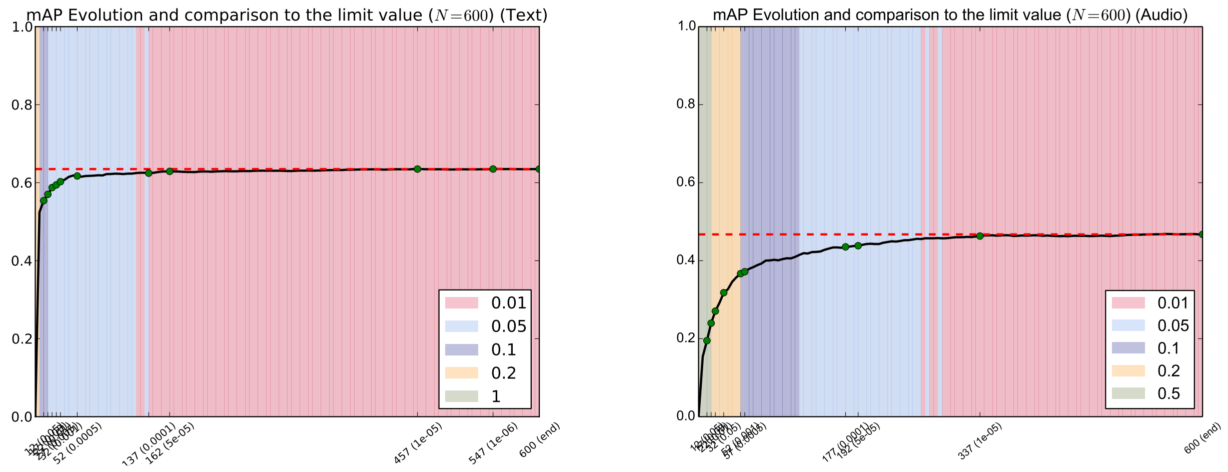


Figure 19: Evolution of the mAP for NER (left) and AUDIO$^{\text{tiny}}$ (right) under the same settings as the previous figure, with a color overlay marking the distance to the optimal mAP value.

4.2 Confidence score

Building on the idea of an on-the-fly criterion, a last issue we consider is the problem of finding a *confidence* score on the obtained similarities, more precisely: can we determine, before comparing against the ground-truth, whether the construction was successful or not for a given sample (i.e., does it have high similarity scores with its true neighbors ?). As a matter of fact, when observing the distribution of the similarities $s(x, \cdot)$ for a given sample x, we can generally distinguish two main categories, as shown in Figure 20: *Category 1* (left) refers to the samples for which the distribution has nicely separated components, while distributions in *category 2* (right) have closer components and more probability mass in the left tail of the distribution. Furthermore, when comparing these distributions to the evaluation results, it turns out that SIC yields better results for the samples in category 1. In fact, the particular distribution shape for this category is probably the result of the classifiers being usually more "consistent" for these samples: As they tend to make the same co-classifications at each iteration, this leads to better separated components.

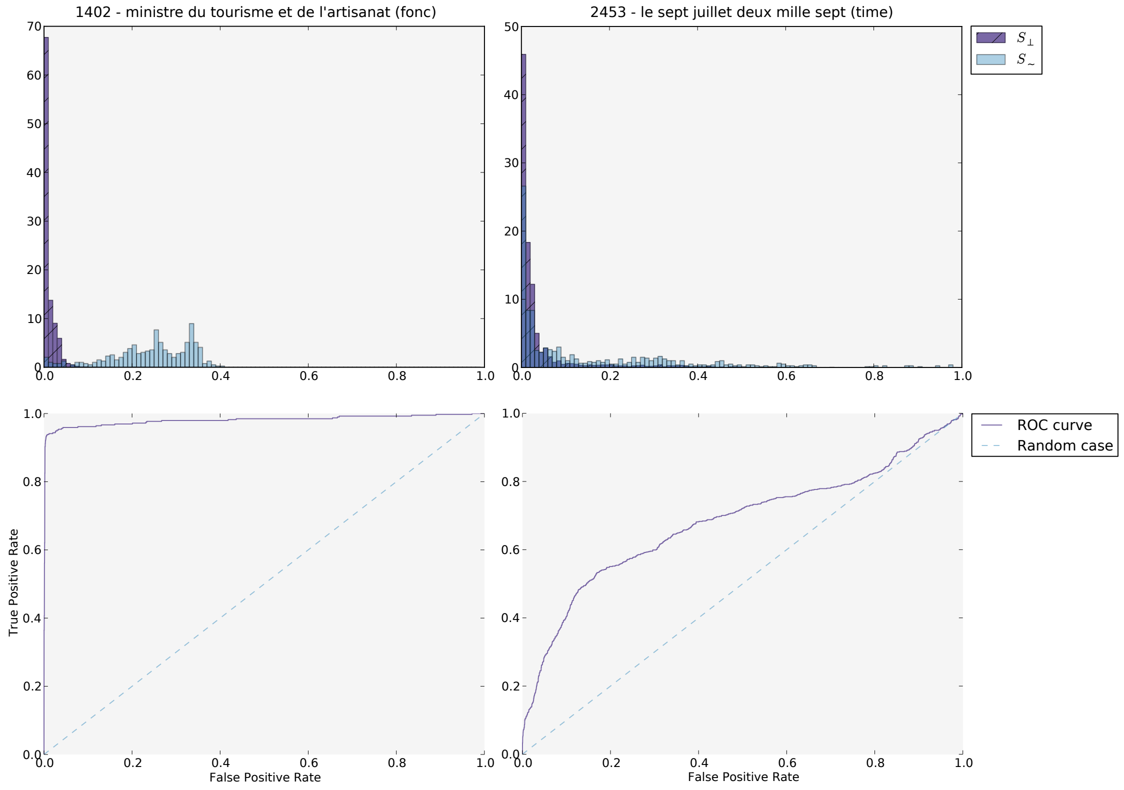


Figure 20: Illustration of Category 1 (left) and Category 2 (right) on the NER dataset. The bottom line represents the ROC curves for the corresponding sample, i.e., the evolution of the True Positive Rate with the False Positive Rate. The farther the ROC curve is from the random case (dotted line), the better it is.

Hence it seems likely that the shape of the similarity distribution could be an indicator of the quality of the similarity, before we even compare it with the ground-truth. To verify this hypothesis, we define a confidence score $c : (x, S) \in \mathcal{X} \times \mathbf{M}_D \to c(x, S) \in \mathbb{R}$, such that high values of $c(x, S)$ means that x likely belongs to category 1. We investigate several usual distribution statistics, based on the distribution shapes observed in the previous figure. For space consideration we only report three of them here: The simplest one, $c(x, S) = max_{y \neq x}(S(x, y)$ characterizes how far the similarity distribution spreads in the range [0,1]. More precisely, $c(x, S) = Var(S(x, y)$ measures how the similarities for sample x spread from the mean. Finally, $c(x, S) = -Skew(\{S(x, y) \mid S(x, y) > mean(S(x, .))\}$ is the *skewness* of the distribution restrained to the values above the mean. A high positive skewness characterizes a distribution with the most of its mass on the right, and a smaller tail on the right, while a low negative skewness characterizes the opposite. In Table 21, we report Pearson and Spearman correlations for each one of these statistics compared to the mAP values obtained when evaluating the corresponding matrix against the ground-truth. We observe that the Max statistics is the best one in terms of generalization, as it gives good results for all datasets. The variance statistics seems to work very well for NER$^{\text{rich}}$ and AUDIO$^{\text{tiny}}$ but indicates no correlation for AUDIO. Finally, the skewness also gives high correlation scores, however with a negative sign for NER$^{\text{rich}}$, which probably comes from the fact that the shape of the distributions are different for the two tasks. To conclude, while none of the proposed measures seems to be a perfect confidence indicator, it is still interesting to see that the obtained correlation values are usually good, which hints to the fact that the shape of the distribution does give some indication about the quality of the similarity.

Setting / Eval.		NER$^{\text{rich}}$	AUDIO$^{\text{tiny}}$	AUDIO
Max	Pearson	0.367	0.470	0.416
	Spearman	0.330	0.455	0.381
Var	Pearson	0.364	0.619	-0.04
	Spearman	0.423	0.567	-0.036
Skewness	Pearson	-0.353	0.144	0.542
	Spearman	-0.367	0.261	0.536

Figure 21: Evaluation of the Correlation between the mAP results and different statistics on the similarity distribution

Conclusion and perspectives

This internship report presents a method for automatically building a similarity on a dataset with limited prior knowledge. The proposed approach is evaluated in the framework of clustering, for which an adequate input similar measure has to be explicitly given by the user. This is a difficult task in case of complex data structures or lack of prior knowledge, and the proposed approach aims to alleviate this problem. In this report, we have presented the main contributions of this internship:

- We proposed a theoretical unified framework for the idea of diverting supervised classifiers for similarity inference, denoted by SIC for simplicity. In particular, we gave an explicit model of the resulting similarity distribution for SIC with binary updates and defined two ways of estimating its parameters.
- We presented an EM and weighted version of the similarity measure They appear to be useful to better distinguish similar from dissimilar samples from their distribution, however we did not have time to exploit them further.

- We developed a parallel implementation of the method and evaluated it on several multimedia datasets. On the more practical side, results show that SIC outperforms usual similarity for the two clustering tasks on audio and text content. While the two settings are very different, in both cases SIC takes advantage of the underlying classifiers' properties which allows the algorithm to indirectly manipulate more complex data representations than the usual similarity, without having to make them explicit. On the contrary, for the task of semantic information retrieval on the Aquaint2 dataset, we reported mAP results lower than the baseline. On of the main causes of these results seems to be the high heterogeneity in the number of occurrences per sample in the dataset. It would be interesting to further investigate SIC in this context but with a smaller and more homogeneous dataset to better identify the issues.

- We also discussed the question of the convergence of the algorithm and how to assess the "quality" of the similarity. While these properties seem to depend mostly on the quality of the underlying classifiers, we proposed an on-the-fly stopping criterion that seems to be efficient in at least two of our applicative frameworks.

Finally, we left out some details and experiments for space considerations; in particular, about the definition of a confidence score and experiments on the Aquaint2 dataset. They will be present in a more complete version soon-to-be available here `http://perso.eleves.ens-rennes.fr/~aroyer/int.html`. In future work, we could study how the method behaves when we consider sequential iterations instead of independent ones: In fact, we can use past iterations to build a temporary similarity and incorporate this knowledge into new iterations (e.g., enforcing the same training class on samples with a high enough temporary similarity). This should accelerate the convergence, but also probably lead to more mistakes if the past results were incorrect. Secondly, it would be interesting to try exploiting the EM and weighted variants further, and study if they can impact the results on the different tasks.

References

[Boriah et al., 2008] Boriah, S., Chandola, V., and Kumar, V. (2008). Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the eighth SIAM International Conference on Data Mining*, pages 243–254.

[Bottou and Bengio, 1995] Bottou, L. and Bengio, Y. (1995). Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems 7*, pages 585–592. MIT Press.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Journal of Machine Learning*, 45(1):5–32.

[Claveau and Gros, 2014] Claveau, V. and Gros, P. (2014). Clustering de données relationnelles pour la structuration de flux télévisuels. In *14 ème conférence Extraction et Gestion des Connaissances*.

[Claveau et al., 2014] Claveau, V., Kijak, E., and Ferret, O. (2014). Explorer le graphe de voisinage pour améliorer les thésaurus distributionnels. In *21ème conférence sur le Traitement Automatique des Langues Naturelles*, Marseille, France.

[Claveau and Ncibi, 2014] Claveau, V. and Ncibi, A. (2014). Knowledge discovery with CRF-based clustering of named entities without a priori classes. In *Conference on Intelligent Text Processing and Computational Linguistics*, volume 8403 of *LNCS*, pages 415–428.

[Cox and Cox, 2000] Cox, T. F. and Cox, M. (2000). *Multidimensional Scaling, Second Edition*. 2 edition.

[Ebadat et al., 2012] Ebadat, A.-R., Claveau, V., and Sébillot, P. (2012). Semantic Clustering using Bag-of-Bag-of-Features. In *CORIA - COnférence en Recherche d'Information et Applications*, pages 229–244.

[Fernandez and Williams, 2010] Fernandez, M. and Williams, S. (2010). Closed-Form Expression for the Poisson-Binomial Probability Density Function. *IEEE Transactions on Aerospace and Electronic Systems*, 46(2):803–817.

[Ferret, 2013] Ferret, O. (2013). Identifying bad semantic neighbors for improving distributional thesauri. In *ACL (1)*, pages 561–571. The Association for Computer Linguistics.

[Halkidi et al., 2001] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145.

[Hodges and Cam, 1960] Hodges, J. L. and Cam, L. L. (1960). The Poisson approximation to the Poisson Binomial distribution. *Ann. Math. Statist.*, 31(3):737–740.

[Joly and Buisson, 2011] Joly, A. and Buisson, O. (2011). Random Maximum Margin Hashing. In *IEEE Computer Vision and Pattern Recognition*, pages 873–880. IEEE.

[Juan and Vidal, 2004] Juan, A. and Vidal, E. (2004). Bernoulli mixture models for binary images. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 3, pages 367–370.

[Klinger and Tomanek, 2007] Klinger, R. and Tomanek, K. (2007). Classical probabilistic models and conditional random fields. Algorithm Engineering Report, Technische Universität Dortmund.

[Kulis, 2013] Kulis, B. (2013). Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364.

[Lafferty et al., 2001] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

[Lavergne et al., 2010] Lavergne, T., Cappé, O., and Yvon, F. (2010). Practical very large scale CRFs. In *48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 504–513. The Association for Computational Linguistics.

[Liu et al., 2000] Liu, B., Xia, Y., and Yu, P. (2000). Clustering through decision tree construction. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, pages 20–29.

[Mporas et al., 2007] Mporas, I., Ganchev, T., Siafarikas, M., and Fakotakis, N. (2007). Comparison of speech features on the speech recognition task. *Journal of Computer Science*, 3(8):608–616.

[Muscariello et al., 2009] Muscariello, A., Gravier, G., and Bimbot, F. (2009). Audio keyword extraction by unsupervised word discovery. In *INTERSPEECH 2009: 10th Annual Conference of the International Speech Communication Association*.

[Park and Glass, 2008] Park, A. and Glass, J. (2008). Unsupervised pattern discovery in speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(1):186–197.

[Perbet et al., 2009] Perbet, F., Stenger, B., and Maki, A. (2009). Random forest clustering and application to video segmentation. In *Proceedings of the British Machine Vision Conference 2009*.

[Pfitzner et al., 2009] Pfitzner, D., Leibbrandt, R., and Powers, D. (2009). Characterization and evaluation of similarity measures for pairs of clusterings. *Journal of Knowledge and Information Systems*, 19(3):361–394.

[Rand, 1971] Rand, W. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

[Rui and Wunsch II, 2005] Rui, X. and Wunsch II, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678.

[Shi and Horvath, 2005] Shi, T. and Horvath, S. (2005). Unsupervised learning with random forest predictors. Technical Report, University of California, Los Angeles.

[Shi and Horvath, 2006] Shi, T. and Horvath, S. (2006). Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics*, 15(1):118–138.

[van Dongen, 2000] van Dongen, S. (2000). *Graph clustering by flow simulation.* PhD thesis, University of Utrecht.

Appendix

A. EM algorithm for a Bernouilli mixture

Given a Bernouilli mixture of the form

$$C^n = \pi_1 \mathcal{B}(\mathbf{p_1}) + \pi_0 \mathcal{B}(\mathbf{p_0}),$$

and a set of observations of the variable C^n, $(o_{j(p,q)})_{p,q}$, we define the hidden component indicator variable $z_{j(p,q)}$, which indicates whether x_p and x_q are neighbors or unrelated samples ($z_{j(p,q)} = \mathbb{1}_{x_p \sim x_q}$), and by Z the random variable associated with the observations z_j. Then the EM scheme to estimate the p_0, p_1 and π_0 parameters unfolds as follow:

Expectancy of the log-likelihood for the complete data.

$$\begin{aligned}
Q(\theta, \hat{\theta}) &= \mathbb{E}\left(\ln(f(\mathbf{o}, \mathbf{z} |\ \mathbf{o}, \hat{\theta}))\right) \\
&= \mathbb{E}\left(\sum_j \ln(f(\mathbf{z}, o_j |\ \hat{\theta})\right) = \mathbb{E}\left(\sum_{j=1}^{\frac{D(D-1)}{2}} \ln(f(o_j \mid z_j,\ \hat{\theta})\ \mathbb{P}(Z = z_j))\right) \\
&= \mathbb{E}\left(\sum_j \mathbb{1}_{\{z_j=1\}}\ \ln(\mathbb{P}(Z=1)\ \mathbb{P}(o_j \mid \mathbf{p_1})) + \mathbb{1}_{\{z_j=0\}}\ \ln(\mathbb{P}(Z=0)\ \mathbb{P}(o_j \mid \mathbf{p_0}))\right)
\end{aligned}$$

$$\begin{aligned}
Q(\theta, \hat{\theta}) = \sum_j & \mathbb{E}(\mathbb{1}_{z_j=1} \mid \hat{\theta})\big[\ln(\mathbb{P}(Z=1)) + \ln(\mathbb{P}(o_j \mid \mathbf{p_1}))\big] + \mathbb{E}(\mathbb{1}_{z_j=0} \mid \hat{\theta})\big[\ln(\mathbb{P}(Z=0)) + \ln(\mathbb{P}(o_j \mid \mathbf{p_0}))\big] \\
= \sum_j & \mathbb{E}(\mathbb{1}_{z_j=1} \mid \hat{\theta})\big[\ln(\mathbb{P}(Z=1)) + \sum_i o_{j,i} \ln(p_{1,i}) + (1-o_{j,i}) \ln(1-p_{1,i})\big] \\
+ & \mathbb{E}(\mathbb{1}_{z_j=0} \mid \hat{\theta})\big[\ln(\mathbb{P}(Z=0)) + \sum_i o_{j,i} \ln(p_{0,i}) + (1-o_{j,i}) \ln(1-p_{0,i})\big]
\end{aligned}$$

Expectation step. Compute $Q(\theta, \hat{\theta})$, which requires $\gamma_{k,j} = \mathbb{E}(\mathbb{1}_{z_j=k} \mid o_j,\ \hat{\theta})$.

$$\begin{aligned}
\gamma_{1,j} = \mathbb{E}(\mathbb{1}_{z_j=1} \mid o_j,\ \hat{\theta}) = \mathbb{P}(z_j = 1 \mid \hat{\theta}) \\
&= \frac{\mathbb{P}(o_j \mid z_j = 1, \hat{\theta})\mathbb{P}(z_j=1)}{\mathbb{P}(o_j \mid \hat{\theta})} \\
&= \frac{\mathbb{P}(o_j \mid z_j = 1, \hat{\theta})\mathbb{P}(z_j=1)}{\mathbb{P}(o_j \mid z_j = 1, \hat{\theta})\mathbb{P}(z_j=1) + \mathbb{P}(o_j \mid z_j = 0, \hat{\theta})\mathbb{P}(z_j=0)} \\
&= \frac{\pi_1 \prod_i p_{1,i}^{o_{j,i}} (1-p_{1,i})^{1-o_{j,i}}}{\pi_1 \prod_i p_{1,i}^{o_{j,i}} (1-p_{1,i})^{1-o_{j,i}} + \pi_0 \prod_i p_{0,i}^{o_{j,i}} (1-p_{0,i})^{1-o_{j,i}}}
\end{aligned}$$

And $\gamma_{0,j} = \mathbb{P}(z_j = 0 |\ o_j,\ \hat{\theta}) = 1 - \gamma_{1,j}$.

Maximization step. Choosing the parameters maximizing $\hat{\theta} \to Q(\theta, \hat{\theta})$:

- Bernoulli parameters.

$$\begin{aligned}
\frac{\partial Q}{\partial p_{1,i}} &= \sum_j \gamma_{1,j}\pi_1 \left(\frac{o_{j,i}}{p_{1,i}} - \frac{1-o_{j,i}}{1-p_{1,i}}\right) \\
&= \sum_j \gamma_{1,j}\pi_1 \frac{o_{j,i}-p_{1,i}}{p_{1,i}(1-p_{1,i})} \\
\frac{\partial Q}{\partial p_{1,i}} = 0 &\Leftrightarrow \sum_j \gamma_{1,j}\pi_1 o_{j,i} = \sum_j \gamma_{1,j}\pi_1 p_{1,i} \\
&\Leftrightarrow \boxed{\tilde{p}_{1,i} = \frac{\sum_j \gamma_{1,j}\ o_{j,i}}{\sum_j \gamma_{1,j}}}
\end{aligned}$$

Similarly,

$$\boxed{\tilde{p}_{0,i} = \frac{\sum_j \gamma_{0,j}\ o_{j,i}}{\sum_j \gamma_{0,j}}}$$

- Mixture coefficients (joint optimization with $\pi_1 = 1 - \pi_0$).

$$\begin{aligned}
\frac{\partial Q}{\partial \pi_0} &= \sum_j \frac{-\gamma_{1,j}}{1-\pi_0} + \frac{\gamma_{0,j}}{\pi_0} \\
&= \sum_j \frac{\gamma_{0,j} - \pi_0(\gamma_{1,j}+\gamma_{0,j})}{\pi_0(1-\pi_0)} = \sum_j \frac{\gamma_{0,j}-\pi_0}{\pi_0(1-\pi_0)} \\
\frac{\partial Q}{\partial \pi_0} = 0 &\Leftrightarrow \sum_j \pi_0 = \sum_j \gamma_{0,j} \\
&\Leftrightarrow \boxed{\tilde{\pi}_0 = \frac{2}{D(D-1)} \sum_j \gamma_{0,j}}
\end{aligned}$$

B. Asymptotical Gaussian distribution for the weighted scores

It is possible to show that $S_\perp(x_p, x_q)$ converges towards a normal distribution $\mathcal{N}(0, \frac{\sigma^2}{T(x_p,x_q)})$ using a generalized version of the Central Limit Theorem, .

Proof. The variables $S_{\perp i}$ are independent, however not identically distributed hence we cannot apply the usual Central Limit Theorem; instead we use the generalized Lyapunov condition. We first define, $\forall \delta > 0$:

$$\sigma_N = \sqrt{\sum_{\substack{i=1 \\ x_p,x_q \in Te_i}}^{N} \mathbb{V}(S_{\perp i})} = \sigma\sqrt{T(x_p,x_q)} \quad \text{and} \quad r_{N,\delta} = \sum_{\substack{i=1 \\ x_p,x_q \in Te_i}}^{N} \mathbb{E}\left(|S_{\perp i} - \mathbb{E}(S_{\perp i})|^{2+\delta}\right) = \sum_i \mathbb{E}\left(|S_{\perp i}|^{2+\delta}\right)$$

Since the expectation is a monotone function,

$$r_{N,\delta} = \sum_i \mathbb{E}\left(|S_{\perp i}|^{2+\delta}\right) \leq \sum_i \mathbb{E}\left(S_{\perp i}^2\right) = \sum_{i=1}^{N} \mathbb{V}\left(S_{\perp i}\right) = \sigma^2 T(x_p, x_q)$$

Hence

$$\frac{r_{N,\delta}}{\sigma_N^{2+\delta}} = \frac{\sigma}{T(x_p, x_q)^{\frac{\delta}{2}}} \xrightarrow[\delta>0,\sigma>0]{T(x_p,x_q)\to\infty} 0$$

Since the Lyapunov condition is verified, we have:

$$\frac{1}{\sigma_N} \sum_{\substack{i=1 \\ x_p, x_q \in Te_i}} (S_{\perp i} - \mathbb{E}(S_{\perp i})) = \frac{1}{\sigma\sqrt{T(x_p, x_q)}} \sum_i (S_{\perp i} - \mu) = \frac{\sqrt{T(x_p, x_q)}}{\sigma} S_\perp \xrightarrow{\mathbb{P}} \mathcal{N}(0, 1)$$

Thus, $S_\perp(x_p, x_q)$ converges in distribution towards $\mathcal{N}(0, \frac{\sigma^2}{T(x_p,x_q)})$.

$\square$

C. Excerpt from the semantic relation graph for Aquaint2 ground-truth

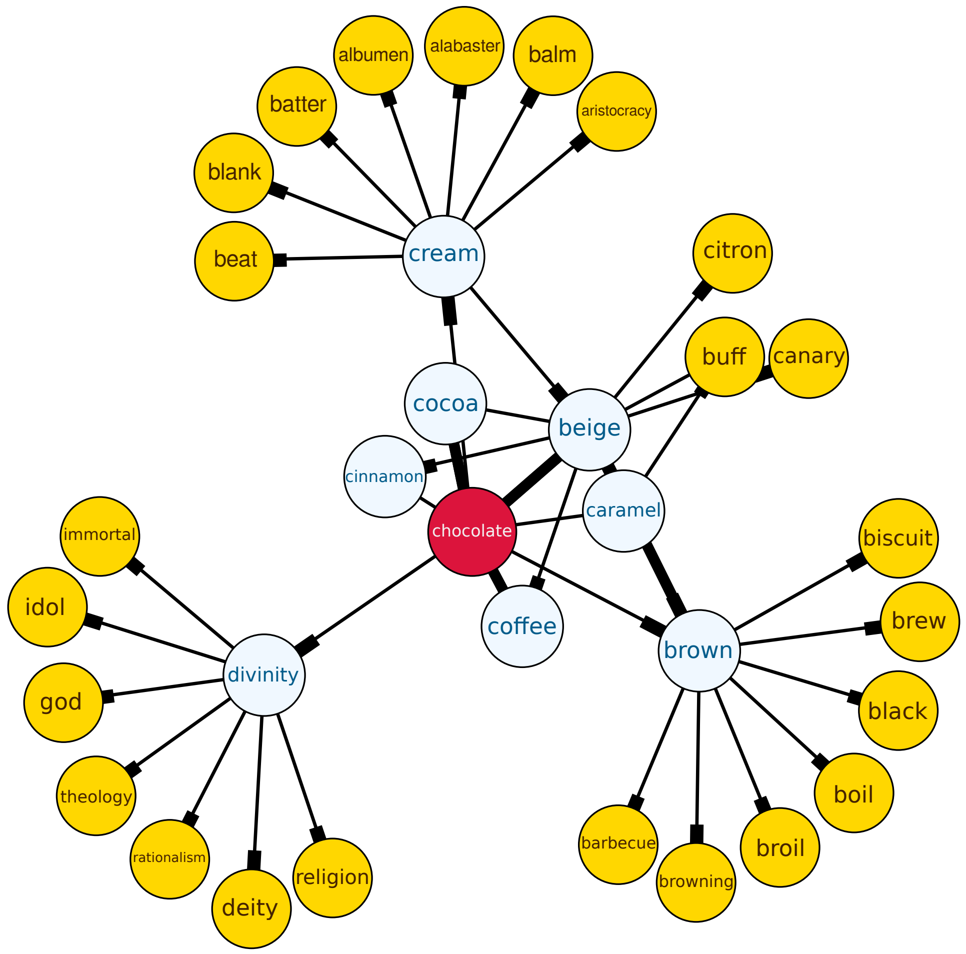


Figure 22: Excerpt of the Aquaint ground-truth neighbors graph starting from node "chocolate" with two levels of neighbors. The edge relation $x \to y$ means that y is in the list of neighbors of x.