Clustering of Document Streams
Xerox XRCE Internship

Cedric Lagnier
Eric Gaussier (LIG), Jean-michel Renders (Xerox)

***Abstract*—With Internet, the number of data published every day increases more and more and a single person is not able to read all documents that are generated. That's why we need clustering algorithms to help us read new documents. In this document, we present the adaptation of some clustering methods (K-means, PLSA, Hierarchical clustering) on document streams. Each method has been developped in order to solve problems of previous methods.**

***Index Terms*—Machine learning, Clustering, Documents stream.**

I. Introduction

In order to deal with the number of new documents every day, we need clustering algorithms: it is easiest to read 100 topics and choose which documents we want to read instead of 1000 documents in a pool. Here, I took an interest in text data streams: a timestamp is assigned to each document which allows us to use a temporal dimension in the clustering.

There are a lot of different text streams: the main streams are news and mails. I will here focus on news, because it is easier to manually detect and understand events, however I have no pre-categorized corpus which I can use to evaluate my algorithms.
The main levels of the hierarchy we can have in the document clustering are:

- **A category** is the first distinction made between articles, it defines the domain of the news. For example, Sports, Politics or Business are categories.
- **A topic** is a distinction in the category. For example, in Sports we can talk about Tennis, Football, ... in Politics we can talk about elections, inner policy, ...
- **A story** is related to a fact which happened. It is characterized by an event succession. For example, Roland Garros 2009 or US presidential elections of 2004 are stories.
- **An event** is the smallest class. It represents a single fact, at a precise time. The defeat of Nadal at Roland Garros 2009 or a krack in Wall Street are events.

In this article, my goal is to group all documents in events. We will see later that only 2 or 3 documents often talk about a same event. Only big events like a tsunami, an election, a world event, ... are defined by a lot of documents.

I will have a look at the main clustering methods and try to adapt them adding the temporal dimension. The main idea in clustering is to find similarity between mathematic representation of elements. An element representation could be a vector, a graph or something else. In document clustering, we represent a document as a vector of words. Values could be binary (presence/absence) or numeric (number of occurrences). A similarity or a distance is then, explicitly or implicitly, computed for each couple of documents.
In documents, we can find words or expressions which are more important than others like person names, place names, dates, ... These words are called named entities. They are very important because in a lot of cases the comparison between two documents is improved when there are named entities. For example, in politics all documents are similar (use the same basic words), but they don't speak about the same persons and then are differentiated by named entities.

There is two main types of clustering methods:

- **Hard clustering methods:** an element belong to one and only one cluster.
  K-means is a hard clustering method.
- **Soft clustering methods:** an element could belong to one, two or more clusters.
  PLSA is a soft clustering method which assign a probability for the membership of an element in each cluster.
  Other methods could simply assign an element to many clusters without any probability.

Here, I am only using full documents, but we can segment each document in little segments which can refer to different events in the same document. The advantage of soft clustering method is then not necessary because we suppose that one segment could only belong to one event. In practice, we see that there are only a few documents talking about more than one event.

In this article, I will first present different works done by other researchers in the domain of event detection and tracking and text clustering using temporal data. Next, I will explain two clustering methods: k-means and PLSA and explain how we adapt them in order to take care of temporal data. In the fourth part, I will present a time-adapted hierarchical clustering method we used to tackle the issues we have with the two previous methods. In the following part, I present an incremental hierarchical clustering method which has the purpose of solving problems of the basic hierarchical clustering method. And finally I will talk about an annotated corpus I made. As I explained before, we didn't have any

annotated corpus and this is a big limitation to evaluate our algorithms. The problem is that we didn't have time to manually annotate a corpus. We decide to, first modelize and manually evaluate some methods and then use a method which obtain better results to help in the corpus annotation.

II. State of the art

The domain of data streams and temporal dynamics has been explored by many people. [8] is a good overview of this domain. Statistical methods are used in [11] in order to detect topics in a three years (330000 documents) corpus using words and named entities, but they don't use the time in their methods. [9] presents an interesting approach of the use of temporal information in the topic detection and tracking problem. They present the concept that the similarity doesn't depend on only term similarity but also on temporal similarity. They define the temporal similarity as the coverage that two events are having.

An other interesting approach that has been observed is the concept of burstiness. In [4], they model the number of occurrences of a word in a time window using a binomial distribution and they compute a burstiness depending on the probability of the number of occurrences observed. Another approach to compute the burstiness of words is presented in [5], where they use a 2-state automaton: bursty and non-bursty states.

The last idea I saw in the domain is in [3]. They define an incremental method which puts each new document in an existing cluster or creates a new cluster if the element is too far from existing clusters using a k-means like algorithm. This method is very interesting because it can be used to cluster documents when they are published and shows the basic idea for event detection and tracking. At each step, the document is the continuation of an existing event (tracking) or defines a new event (detection).
Another article presents a different method but still in the aim of clustering data streams with an incremental method: in [13], they extend the conventional kernel density clustering method in order to make clusters.

III. Time concept introduction in K-means and PLSA

A. Basic methods

The aim of these two methods is to separate data into K clusters (K is a pre-defined number).

1) K-means: K-means compute the K centroids to get the profiles of the K clusters. After the initialization, it iterates a simple 2 phases algorithm until convergence is reached. The 2 phases are assign each element to the nearest cluster and recompute the new centroids.

2) PLSA: The Probabilistic Latent Semantic Analysis method ([6]) is based on a generative model defined by figure 1. The principle is to suppose that a latent class links documents and words. There is a probability that a word will

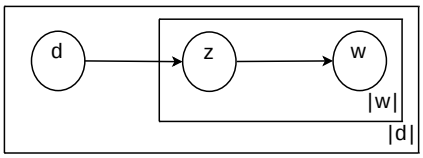

Fig. 1. Generative model for PLSA

be generated in a specific document (in practice, we just have to count occurrences of a word to have its probability). In PLSA, we say that a latent class is generated according to a document and then a word is generated according to the latent class. In our case, the latent classes are events: a document belong to one (or some) event(s) and an event is defined by some words.

Then we can define the joint probability of a word and a document with latent class:

$$p(d,w) = \sum_z p(d) * p(z|d) * p(w|z)$$

B. Updated methods

The two previous method are well-known and don't take the time into account. The next methods are the same, but adapted to take into account the timestamp of each document.

1) K-means: The idea is the same as the basic method: compute K centroids, each defined by a profile. But an element or a centroid profile is not a single vector: there is one vector per feature type. This idea is explained in [10]. We could have N different feature types. In our case, we just use two types: words frequency and timestamps. The idea is to say that 2 documents talking about the same thing but not at the same time have small chances to refer to the same event. The only thing changing in the algorithm is the computation of the distance between a document and a centroid:

$$dist(d,c) = \lambda(dist_{term}(d,c)) + (1-\lambda)(dist_{temporal}(d,c))$$

where d is a document and c is a centroid.

For the temporal distance, we tried 3 possibilities:

- Each document and each centroid have a single timestamp representation:
  $dist_{temporal}(d,c) = |timestamp(d) - timestamp(c)|$
  It is very difficult to tune the parameter λ with this method because the data are very heterogeneous. Moreover, this distance doesn't allow an event to appear in more than one timestamp, even if timestamps are neighbors.
- Each document and each centroid have the same representation for the timestamps and the features. For the documents, it is a boolean value. For example, [0,0,1,0] is the timestamp of a document and [0.3,0,0.7,0] is the timestamp representation of a centroid.
  $dist_{temporal}(d,c) = \sqrt{\sum_t (d(t) - c(t))^2}$
  where t is a timestamp
  This distance allows, for example, an event to appear the

Monday and the Friday of a week, and is very simple to include in the algorithm.

- The last method is the same as the previous but we smoothed the documents time vector representation with a binomial distribution. The idea is to say that instead of a document appears at day t, it has a probability to appear at day $t-1$, $t+1$, $t-2$, ... and these probabilities are defined by a binomial distribution centered on t.
  For each timestamp x: $p(x|t) = \binom{n}{x} p^x (1-p)^{n-x}$ with $n = \frac{t}{p}$.
  where p is a parameter which defines the standard deviation: in this idea, it is the number of days during which the document timestamps have a positive value.

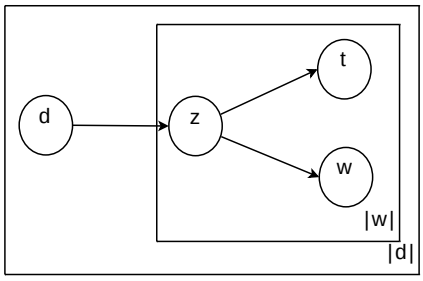

Fig. 2. Generative model for PLSA with time-stamps

2) PLSA: In order to use the time in PLSA, we have defined a new generative model shown on Figure 2. This model generates, in addition to each word, a timestamp. The basic idea is that each document is defined by a timestamp. But we could imagine a document which talks about some events which don't happen the same day and then generate a timestamp for each paragraph. Going further, we choose to generate a timestamp per word even if we know that a word alone doesn't talk about an event, but this is the more generic model using this idea.
We define the joint probability using this new model:

$$p(d,w,t) = \sum_z p(d) * p(z|d) * p(w|z) * p(t|z)$$

We can use an E-M algorithm to compute the values of $p(z|d)$ which define the probability of each document to belong to each cluster and $p(w|z)$ which is the probability of each word to appear in each cluster (the model of the cluster).
E-step:

$$p(z|d,w,t) \propto p(d) * p(z|d) * p(w|z) * p(t|z)$$

M-step:

$$p(w|z) \propto \sum_{d \in D} \sum_{t \in T} \#(d,w,t) * p(z|d,w,t)$$
$$p(z|d) \propto \sum_{w \in W} \sum_{t \in T} \#(d,w,t) * p(z|d,w,t)$$

There is another probability we need to compute in the M-step: $p(t|z)$. We first define a basic model (like the computation of $p(w|z)$ and $p(z|d)$) using a multinomial:

$$p(t|z) \propto \sum_{d \in D} \sum_{w \in W} \#(d,w,t) * p(z|d,w,t)$$

We can add the smoothed time representation of the document in order to help PLSA to detect nearest documents.

The problem is that cluster time representations obtained using this method are not focused on neighbor timestamps and that's why we tried to use a beta distribution ([2]). The idea was to smooth the time distribution of an event and force it to be focused on some neighbor days. The model is the following:

$$p(t|z) = Beta(t, \ _1, \ _2)$$

and

$$Beta(t, \ _1, \ _2) = \frac{t^{\ 1-1}(1-t)^{\ 2-1}}{\int_0^1 u^{\ 1-1}(1-u)^{\ 2-1} du}$$

We used the method-of-moments ([2]) to estimate the parameters $_1$ and $_2$ for each cluster.

3) Evaluation: I evaluated these two methods on a corpus that spans one year: LE MONDE 2004. This corpus is composed of about 43000 documents and 160000 words. Another thing is that this corpus is in French and, in this report, I have translated the different words. This corpus is not annotated with events, so I had to manually inspect clusters to find problems of these methods.

Before showing different results, I will quickly present pre-processing I've done on the corpus. These pre-processing are done for all evaluations I show in this report:

- keep words which appear in more than 2 documents: the goal is to delete words which may be errors of previous step.
- delete days which contain no document: delete week-ends and holidays.
- compute the tf-idf of each feature for each document.

The term distance (distance between documents using only words) used in evaluations is the cosine distance:
$dist_{cosine}(A,B) = 1 - \frac{A.B}{||A||.||B||}$ where A and B are two term vector representations of two documents.

The first thing I evaluated was the k-means method

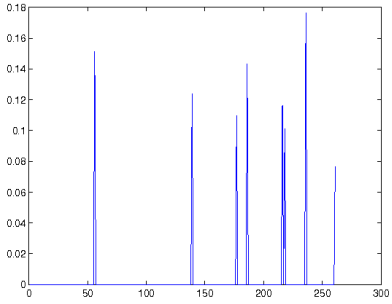

Fig. 3. clusters time representation using k-means or PLSA with the unsmoothed document time representation

money	regime	hundred
to adress	good	George
statement	core	
weapon	Bush	

Fig. 4. clusters words representation using k-means or PLSA with the unsmoothed document time representation

with the unsmoothed time representation method and the PLSA with multinomial time representation method. These two methods gave similar results. Figure 3 shows a common cluster time representation. Observations done on this cluster are the same as these done on others, but I just present this cluster here. We see that we grouped 8 timestamps in the cluster. Many of the documents at these timestamps belong to the cluster whatever words which belong to them. For example, figure 4 shows words which define this cluster. With this observation and the reading of documents which belong to the cluster we see that our clusters don't represent events but just timestamps where documents talk about "similar" things: I mean that the mean document words vector representation of these timestamps are similar.

The second thing I evaluated was the K-means (or PLSA)

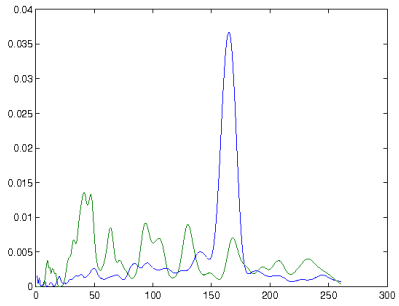

Fig. 5. clusters time representation using k-means or PLSA with the smoothed document time representation

cluster 1

Athens	medal	doping
olympic	champion	record
games	athletics	

cluster 2

wine	Cesare	italian
Battisti	extradition	alcohol
ethic	Italy	

Fig. 6. clusters words representation using k-means or PLSA with the smoothed document time representation

with the smoothed time representation method. The smoothing permits to say that events could be neighbors (in the time). Figure 5 shows the time representation of two clusters and figure 6 shows words which define these two clusters. Unlike the previous method, we see that clusters represent a valid topic: Olympic games and Cesare Battisti extradition for these two clusters. This is not what we defined as an event but as a story. This is a real progress but these methods couldn't be used to find more specific clusters: they are not scalable with a big number of clusters due to the memory cost.

The third thing I evaluated was the PLSA with beta

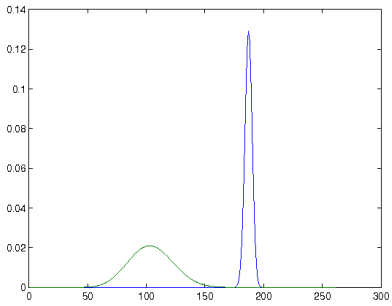

Fig. 7. two cluster time representations using PLSA with beta distribution

distribution as time representation method. Figure 7 shows two cluster time representations using this method. The beta distribution constrains the cluster to be centered on only one timestamp which matches with how we defined an event. But the problem of clusters which group a lot of events still persist and we don't find any story like the previous methods: clusters are very scraggly.

4) Problems: These methods have a big issue in the event detection task. As I said before, we don't know the number of events in our corpus and these two methods need the number of clusters before the start of the clustering. Another problem we encounter was that these methods work with a small number of clusters and in a one year corpus, there is a big number of events, so they are unable to detect fine-grain events in a big corpus.

But we have seen that with good parameters and the smoothed time representation method, we can detect some stories.

In the next part, I have kept these methods while trying to improve them using another way to detect events: the word burstiness.

C. Burstiness concept

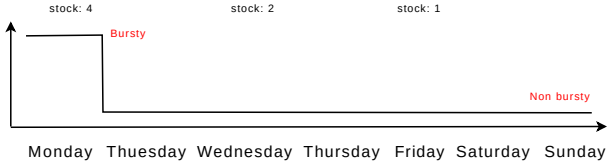

Fig. 8. burstiness example for the word "stock" over a week

1) Burstiness: A word is defined as bursty when it is more present than its normal frequency. The burstiness notion is linked to the time. For example and as illustrated on figure 8, suppose we work with a one week corpus and take the word "stock" which appears 7 times in the corpus: so its mean frequency per day is 1. If it appears 4 times the Monday, we will say that this word is bursty on Monday. The idea is the same in a large scale.

The first method to compute the burstiness of each word is very intuitive, it consists in comparing number of occurrences of a word in a time window with its average frequency, like in my example (the average frequency can be easily computed by count). The problem of this method is that a word can be bursty at day d, not bursty at day $d+1$ and bursty again at day $d+2$. In order to smooth this burstiness, we used an infinite-state automaton method ([7]) for each word.

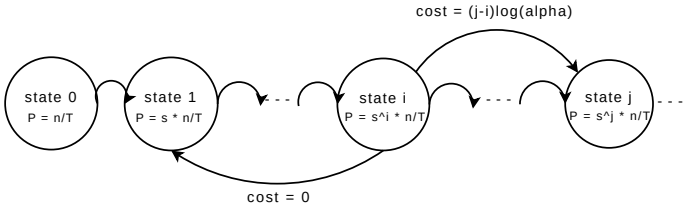

Fig. 9. Find burstiness with an infinite-state automaton

2) Compute burstiness with an infinite-state automaton: We consider here a corpus with T time windows and a word which appears n times in the corpus. Figure 9 illustrates this method. Each state correspond to a burstiness level and a probability is associated to each state. This probability is the probability that the word appears in a document at the particular time window. For example, if the time window contains 10 words, a word that appears two times has a probability of 20%.

- state 0: not bursty ($P_0 = n/T$)
- state 1: bursty level 1 ($P_1 = s(n/T)$) where s is a parameter
- ...
- state i: bursty level i ($P_i = s^i(n/T)$)
- ...

It is not really an infinite-state automaton because the probability assigned to a state can't be bigger than 1, so there is a i_{max} for each word, but this maximum state depend on P_0 which is the occurrence probability of the word in the entire corpus.
Then, we associate a cost to each time window. For a time window t, p(t) is the probability of the word and s(t) is the state:

$$cost(t) = (p(t) - P_{s(t)})^2$$

Then we just have to do a shortest path algorithm such as Dijkstra to find the sequence of states which match better with the sequence of observed probabilities of the word. Figure 10 shows an example of this method. The transition

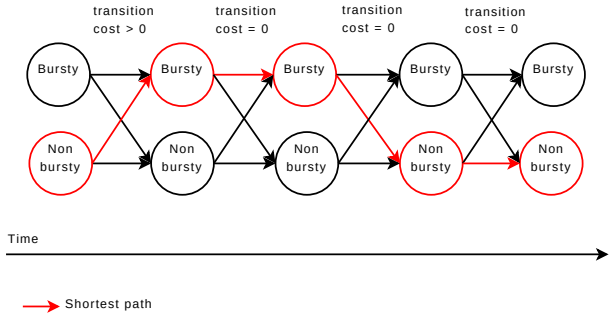

Fig. 10. Infinite-state automaton method with only 2 states

cost when becoming more bursty is shown. But this method is equivalent to the previous one ... it is just a little more complex.

The difference is that we add a cost transition between states. The cost is zero when the state "decreases", but it is defined as γ when the burstiness state increases.

$$\gamma = (j-i)log(\alpha)$$

where j is the new burstiness level, i the previous burstiness level and α a parameter. These transition costs smooth the burstiness: a non-bursty word is not labelled bursty just for one day and a bursty word is not labelled non-bursty just for one day.

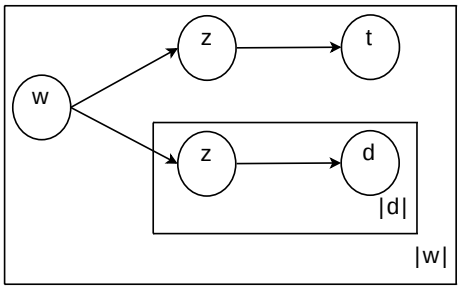

Fig. 11. Generative model for PLSA applied to bursty words

3) New method: After computing the burstiness of each word, we need to find events and then we define an event as a cluster of words which are bursty together. We decided to use PLSA again (and not k-means) because a word could belong to many clusters (for example the word "politic" does not belong to only one cluster) and k-means don't permit that an element clustered belong to more than one cluster. We define the model shown on figure 11. Following, I write the new formulas corresponding to the model. We just include a weight λ for the time dimension:

$$p(w,d,t) = \sum_z p(w) * p(z|w) * p(d|z) * p(t|z)$$

E-Step:

$$p(z|d,w) \propto p(w) * p(z|w) * p(d|z)$$
$$p(z|t,w) \propto p(w) * p(z|w) * p(t|z)$$

M-Step:

$$p(d|z) \propto \sum_w \#(w,d) * p(z|d,w)$$
$$p(t|z) \propto \sum_w \#(w,t) * p(z|t,w)$$
$$p(z|w) \propto \sum_d \#(w,d) * p(z|d,w) + \lambda \sum_t \#(w,t) * p(z|t,w)$$

The parameter λ has to be computed in order to match with news documents.

4) Evaluation: I evaluated this method on the same corpus as previously: LE MONDE 2004. First, I computed the burstiness of each word using these parameters:

- $s = 2$
  The idea is to say that if a word appears twice its mean frequency in a time window, so it is bursty in this time window.
- $\alpha = 0.01$
  We say that, in average, a word is bursty once for one hundred days.

After this computing, we keep only the words which are bursty at least once. We keep only 9000 words.

Next we have to compute the probabilities using the defined PLSA method with parameters:

- K: it defines the number of clusters. The clustering doesn't change so much when we change this value a little. For example, clusters with $K = 40$ or $K = 120$ are similar and the same problems are presents. In the following example, I choose $K = 40$.
- λ: This parameter is used to increase the importance of one of the two dimensions when defining the clusters: words and time.

  - If we decrease λ, we increase the weight of the words and decrease the weight of the time.
    In this case, the time representation of clusters is like the unsmoothed method and cluster documents talk about the same topic but aren't grouped in time.
  - If we increase λ, we decrease the weight of the words and increase the weight of the time.
    In this case, the time representation of clusters is very peaked in the time and cluster documents don't talk about specific topic.

  $\lambda = 3$ is the value giving average results, which make clusters group in time in which documents talk about the same topic. So I used this value for the examples I present in the following.

Figures 12 and 13 show two clusters computed by this method. We can see that these two clusters group documents which talk about a story (like the smoothed method present in the previous section):

- cluster 1 documents talk about a strike of people which work in the shows.
- cluster 2 documents talk about a tsunami which hit Asia countries.

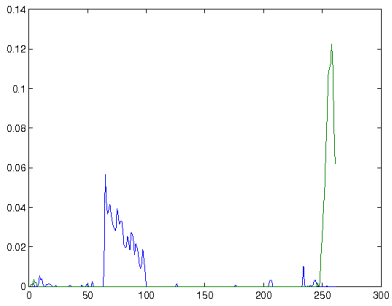

Fig. 12. clusters time representation using PLSA on bursty words

cluster 1

employers	contribution	CFDT
syndicate	discharge	CGT
unemployed	strike	

cluster 2

tsunami	Atjeh	magnitude
Andaman	Maldives	tidal wave
Phuket	seaside	

Fig. 13. clusters words representation using PLSA on bursty words

A thing I can add is that stories found with this method are not the same as the ones found with the smoothed method. The problem of events detection is still unsolved by this method: using PLSA, we are not able to find more and more clusters.

5) Problems: This method gives better results than the previous one, but there are still some issues:

- This method finds only big stories. By "big story" I mean stories which are defined by a lot of documents, for example a hurricane, a war or presidential elections. But we want to be able to find little clusters defined by only a few documents too, and these clusters can't be found with words burstiness because the burstiness of their words is hidden before the words burstiness of big stories.
- An other problem of this method is that it does not find constant stories, like weekly events.
- And the problem we have since the beginning persist: we need to choose the number of clusters.

In the next part, I will present a different way we looked at: the hierarchical clustering methods. Hierarchical clustering methods are associated with cut methods which allow to make a clustering without a fixed number of clusters.

IV. Hierarchical clustering

A. Basic method

The hierarchical clustering method links each document with each other in a tree structure called "dendrogram". At the beginning, each document is a cluster and at each step, the algorithm links the 2 nearest clusters making a new cluster.

B. "cut" principle

The dendrogram is just a linkage of all the elements and don't give any clustering. It shows the distance of each link. So, we need a method which defines where we have to cut the tree in order to have clusters which best match with what we want.

- A classic method is to choose a number K of clusters and cut the tree as soon as we have this number of clusters. This method does not help in my case, because we don't know the number of events and that's why we choose to try hierarchical clustering.
- The first and simplest method I used consisted in cutting the tree at a defined distance level in element groups.
- The second method I used was to cut following the inconsistency criterion defined by Math-works ([1]). The main idea is: for each link, we look at the difference between the distance of the current link and the distances of the previous links and then compute an inconsistency coefficient. If the coefficient is greater than the limit we choose, we cut the tree. By default, the algorithm use only the two links before the current link (for example, at link ABCDE, the two previous links are AB and CDE).

  $$I_k = \frac{z_k - \bar{z}_{k,k-1,k-2}}{\sigma_{k,k-1,k-2}}$$

  where I_k is the inconsistency of the link k, z_k is the distance of link k, $\bar{z}_{k,k-1,k-2}$ is the mean of the distances of links k, k-1 and k-2, and $\sigma_{k,k-1,k-2}$ is the standard deviation.
  For example, we can compute the inconsistency of the link ABCDE as:

  $$I_{ABCDE} = \frac{d_{ABCDE} - \frac{d_{ABCDE}+d_{AB}+d_{CDE}}{3}}{\sigma_{AB,CDE,ABCDE}}$$

  When two documents (leafs of the tree) are linked, the inconsistency of the link is 0.

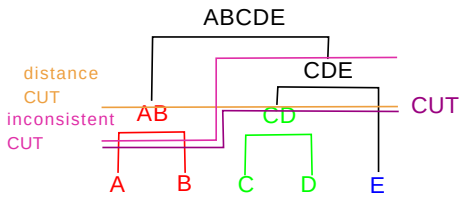

Fig. 14. Cut a dendrogram with 2 conditions: distance and inconsistency

The problem with the inconsistent cut is that when a document is linked with a cluster and the link has a great distance or the new cluster has a poor similarity, the link is not declared inconsistent.
In order to solve this problem, we defined a new cut method (Figure 14) which use at the same time the inconsistency and the distance criteria to check a link. A cluster is valid if the distance which make it is below the threshold and if the link which create the cluster is not inconsistent.

C. Updated method with timestamps

1) Model: Like the K-means model, we want to include the time in the similarity between elements and as for the K-means model, we changed the similarity (distance) function. But we tried here a more complex time similarity function in order to increase the difference between nearest elements and farther elements. We wanted to have a near zero similarity for elements which have a month between them and keep the full textual similarity for documents which have been published the same day.

$$dist(d1, d2) = dist_{term}(d1, d2) * dist_{time}(d1, d2)$$

And the definition of distance functions:

$$dist_{term}(d1, d2) = dist_{cosine}(d1, d2)$$
$$dist_{time}(d1, d2) = e^{\frac{-t}{n}} \text{ where n is a parameter and}$$
$$t = |t(d1) - t(d2)|$$

if we want a quick decrease of the time similarity, we choose a small n and if we want a slow decrease, we choose a large n.

2) Problems: This method can't be used for a big corpus. We need to save and cut the dendrogram in order to create clusters, but this tree is too big when we have a one year corpus. That's why we made the evaluation of the hierarchical method on small time windows, like weeks. At the beginning, we just tried to take time windows which overlap in order to merge clusters of following clusterings, but this method didn't create very good clusters, so in the next part I will explain another method creating clusters with an incremental method.

3) Evaluation: The evaluation of this method have been made on the corpus which creation is explained in section **??** but before annotation. At the moment, I didn't have the time to finish the corpus annotation. I used this evaluation for 2 reasons:

- Test if this method could make a good event clustering.
- Test which linkage function use to help in the creation of the annotated corpus made in section **??**.

At the moment, the corpus is composed by approximately 15000 documents which contain 35000 words and 15000 named entities. The method used is the mix between inconsistency and distance with the two threshold value fixed to 0.8.
I choose these threshold after looking at clusters computed using different values and I choose values which are not optimal, but make average good clusters. The bigger issue we can find here is one event in more than one cluster, solved easily manually.

Figure 15 shows the number of elements in clusters using this method. The average number of elements in a cluster is $2,27$. We can see that clusters contain a few documents which match with the lot of little clusters: one or two documents written on only one webnews on a random event.
We have a problem with big events containing up to 15

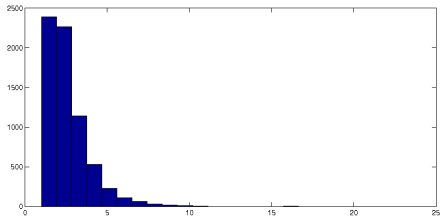

Fig. 15. Number of elements in clusters

documents. We see on the histogram that we only have one cluster with more than 15 documents:this cluster talk about "the victory of Helio Castroneves at Indianapolis 500". But this is not the only event which have a lot of documents in this time period. For example, we have an event which talk about Alia cyclone in India and Bangladesh which have 15 documents. These other big events are separated in many little clusters due to the parameters I choose. These parameters have to be estimated and that's why the creation of the annotated corpus is very important.

In order to try to quantify the quality of the different linkage functions, I have computed two measures on the corpus:

- **cophenetic correlation coefficient:** the goal is to compare the distance computed between two documents $x(i,j)$ before the hierarchical clustering and the distance between the same two documents but in the dendrogram $t(i,j)$: this distance is just the height of the link which group these two documents.
$$c = \frac{\sum_{i<j}(x(i,j)-\bar{x})(t(i,j)-\bar{t})}{\sqrt{\sum_{i<j}(x(i,j)-\bar{x})^2 \sum_{i<j}(t(i,j)-\bar{t})^2}}$$
with $\bar{x}$ the average of all $x(i,j)$ and $\bar{t}$ the average of all $t(i,j)$
This coefficient value is between 0 (bad value) and 1 (very good proximity between the two distances). It checks if the hierarchical clustering keep the distances between documents.
- **spearman's rank correlation coefficient:** it compare the ranks (the nearest is the first) of all the documents for each document before the clustering and after the clustering (distances computed using the dendrogram). Compared to the previous coefficient, it check only the rank and not the distance.
$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$
with $d_i = |x_i - y_i|$ the difference between ranks of the i^{th} value of the two datasets and n the number of elements in the datasets.
We have just to do the mean on documents to have the corpus coefficient.
This coefficient is between -1 (the two list ranks are opposite) and 1 (the two list ranks are equal).

linkage method	cophenetic coefficient	spearman coefficient
single	0.2080	0.1107
complete	0.5793	0.1194
weighted	0.7259	0.1622
average	0.8007	0.3011

TABLE I
DIFFERENT MEASURES MADE ON A HIERARCHICAL CLUSTERING

Table I shows the results obtained by the four linkage methods with these two measures. We see that the average linkage is the method which gives better average results and then I choose to use this method to make the annotated corpus.

D. Incremental Clustering

Right now, this method is just in an experimental phase. I haven't done any evaluation and I'm waiting for the annotated corpus in order to quantify the quality and not just say "clusters look good". Figure 16 shows the main idea:

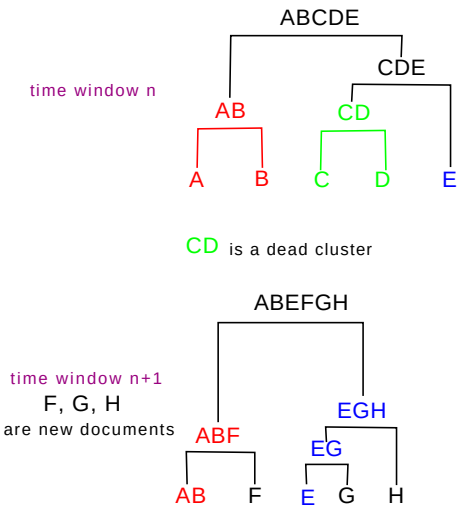

Fig. 16. Idea of an incremetal hierarchical clustering method

- At step n, the algorithm makes a hierarchical clustering (presented in the previous section) and returns clusters which correspond to events. The hierarchical clustering method used has to be tuned in order to detect right events.
- Then, clusters (defining events) being dead (not alive) are put into an archive section: a cluster is alive if recent (less than t days year old) documents belong to it.
- At step $n+1$, the hierarchical clustering algorithm is launched on still alive clusters and new documents.

At any moment, clusters being alive and clusters in archive section represent all events which have been detected so far.

V. BUILDING AN ANNOTATED CORPUS

A. Why an annotated corpus ?

In the aim of evaluate our methods and quantify the quality of our methods, we need a corpus where each document is

annotated by one or some events it describes. Unfortunately, we don't have any corpus with these requirements. So we decided to crawl and annotate news on the web. The problem is that manually annotate an entire corpus is very expensive and/or take a lot of time if we have no tool in order to uncover some underlying structure. That's why we choose to use clustering method to help us to create these annotations:

- First, we use a clustering algorithm (which works pretty well according to a manual evaluation) to create document clusters which are supposed to represent events.
- Next, we manually use the different events discovered by the clustering algorithm to annotate a part of the corpus and manually check for the others.

B. Processing

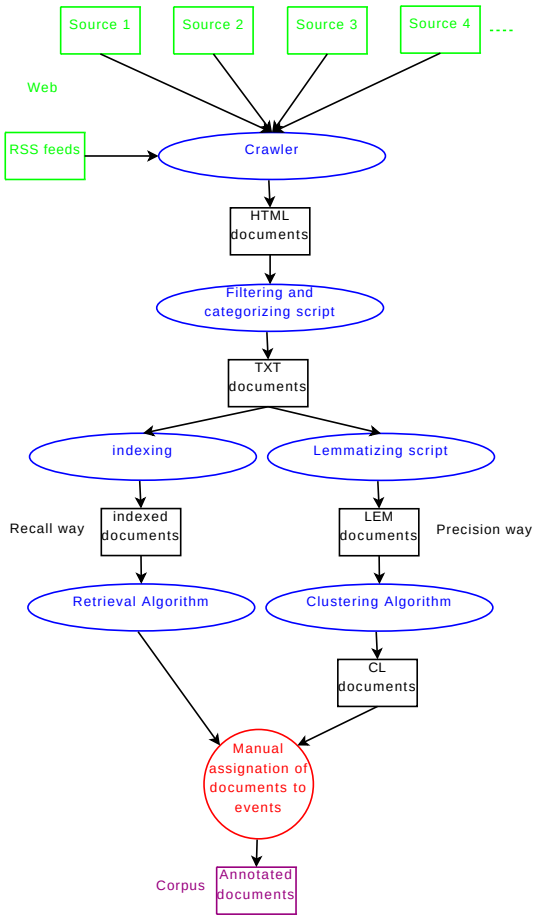

Fig. 17. Creation of an annotated corpus

1) Crawling: The first step consist in choose different sources on the web and crawl the news published by these sources. We choose 11 sources: Associated Press, BBC, CBC, CNN, Fox News, News York Times, Reuters, Ria Novosti, UsaToday, Washington Post and Xinhua. We tried to have variety in the source countries to have national and international news. We have USA news, Britain news, Russian news and Chinese news. The problem in the sources choice is that all the news have to be in the same language (in English) for the clustering. The second choice is the categories we want, and we took: latest, business, world, sports and technology because they are the most present categories in the different webnews.

In the next step, we have to download all the news we choose. In order to do that, each hour the Crawler download the different RSS feeds of the source web sites and crawl news linked in RSS feeds. To avoid downloading many times the same news, the crawler save at each session titles and links of the news crawled.
I choose one hour between two crawls to guarantee that each news is crawled and we don't miss any document: source web sites don't give any information about this subject, but this is the google reader frequency and I tested on many time periods that no news is forgotten with this crawl frequency.

2) Filtering: All news are stored on the hard drive in HTML format and we need to filter these files in order to keep only the title and the main text. This is also in this phase that we find the categories for the latest RSS feed news. The filter script uses a HTML parser and keeps only text in chosen markups. The operation is the same for the category search. Each source have its proper news model (markups which define title, main text or category are not the same in two different sources) and I have made a parser per source.

An other task done in this part is the suppression of identical news. Some sources duplicate some news in the latest RSS feed and in another category, so we have to delete news which appear many times in our corpus. We used an basic method which consists in generating and comparing the md5sum of each filtered file: the md5sum is the result of a hash function called md5. The main characteristic of a hash function is that it is very difficult to find two input data which have the same result.

3) Lemmatization: A Xerox algorithm takes news text files and finds for each news the different words and their role in the sentence: it lemmatize documents. The algorithm is also able to find named entities. A named entity is a word or a group of words having special meaning: it could be a person name, a place name, a date, an organization name, ... As I said in the introduction, these named entities are very important. A lot of events are defined by these named entities and making a difference between normal words and named entities let us use choose weights in order to take more care of named entities than basic words. In the next step, we don't keep all words: some word types are useless to find similarities between documents, for example it's the case for adverbs. We kept only nouns, verbs, adjectives and noun-adjectives.

4) Clustering: The last automatic step of this corpus creation is the clustering of documents. We use the normal hierarchical clustering method presented in previous part on a little set of documents (about 10000 documents). I make the difference between this set and a entire one year corpus which include about 100000 documents. We link the most present words in a cluster of documents as words which

define the cluster. This helps us to know what subject the cluster documents is referring to. Then, we randomly choose N clusters (N depend on the size of the corpus we want) and manually annotate all the documents in this set. For a first corpus, we choose $N = 150$.

5) Manual post-processing: The final step of this operation is the manual definition of events. This step is very important and have to be done with attention. I will explain in the next part the dangers of using the clustering we want to evaluate to help us to annotate documents. There are two main phases during this step:

1) We define events helped by the clustering (we check at all clusters and documents of the little set and find events).
2) We look at documents in all the corpus and add documents belonging to our defined events in our little set.

To summarize, in this phase, we define events using the randomly selected set and incorporate documents from the complete corpus in our events. We have to find all documents which belong to our events in order to make a good evaluation.

C. Problems and Use

As I said before, our method to annotate the corpus is very dangerous. Annotations and the method we want to test have to be independent. For example, if we annotate a corpus with a K-means method, the test of the K-means method will have very good results: that's normal because the corpus is already annotated by this method.
We finally manually annotate the corpus, but we always have to be warned. We have to make the same annotations as if we directly annotate the corpus without using the clusters. The pre-clustering method is used only to help us. It's like if for each document an algorithm present us the K nearest neighbors of the document. We know documents which talk about the same events are in its neighborhood. but a document in its neighborhood is not necessarily talking about the same event.

This annotated corpus then can be used to test our methods and quantify the quality of each. It help us to find parameters values with which we obtain best results and to choose the different function, like linkage function or distance function which better find events.

VI. Conclusion and Perspectives

I have made and tested different models in order to solve the problem of event detection in big corpus and the method which has obtained the best results is the hierarchical clustering principally because we don't have to choose the number of events presents in the corpus: cut methods made good division to shift events. At the moment, I just tested it on a small corpus in a defined time window and I have still to evaluate the different linkage function and the different threshold values to conclude on this method. The next step is to finish the model of the incremental hierarchical clustering method and evaluate it in order to know if the method could be applied on all documents published on the web.

If I had more time, I would have tried to improve this method or to test other angles of view of the problem:

- The first idea to improve the hierarchical clustering method is to use the burstiness. We saw in previous part that the burstiness help to detect events and that's why it could help even with the hierarchical clustering.
- The second idea is to use one step of PLSA on the results of the hierarchical clustering. After finding and define all events, we could use PLSA to re-cluster all documents and then find documents which belong to more than one event.
- The third idea is similar to the previous: segment all documents in paragraphs or sentences and cluster these segments which could belong to only one event. If we segment documents, it's not necessary to use PLSA.
- With the hierarchical method, some documents are linked because they can't be inconsistent, but their distance is sizeable. This is due to the inconsistency definition. The fourth idea is to add a linkage condition on documents: if the distance between two documents is above a threshold, documents are not linked.

References

[1] Mathworks, statistics toolbox. http://www.mathworks.com/access/helpdesk/help/toolbox/stats/inconsistent.html.
[2] Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Beta_distribution.
[3] Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *In 2006 SIAM Conference on Data Mining*, pages 328–339, 2006.
[4] Gabriel Pui Cheong Fung, Gabriel Pui, Cheong Fung, Jeffrey Xu Yu, Philip S. Yu, S. Yu, and Hongjun Lu. Parameter free bursty events detection in text streams, 2005.
[5] Qi He, Kuiyu Chang, Ee-Peng Lim, and Jun Zhang. Bursty feature representation for clustering text streams. In *SDM*, 2007.
[6] Thomas Hofmann. Probabilistic latent semantic analysis. In *In Proc. of Uncertainty in Artificial Intelligence, UAI99*, pages 289–296, 1999.
[7] Jon Kleinberg. Bursty and hierarchical structure in streams, 2002.
[8] Streams Jon Kleinberg and Jon Kleinberg. Temporal dynamics of on-line information. In *In Data Stream Management: Processing High-Speed Data*. Springer, 2006.
[9] Juha Makkonen, Helena Ahonen-myka, and Marko Salmenkivi. Topic detection and tracking with spatio-temporal evidence. In *In Proceedings of 25th European Conference on Information Retrieval Research (ECIR 2003*, pages 251–265. Springer-Verlag, 2003.
[10] Dharmendra Modha and Scott Spangler. Feature weighting in k-means clustering. In *Machine Learning*, page 2003, 2002.
[11] David Newman, Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. Analyzing entities and topics in news articles using statistical topic models. In *In ISI*, pages 93–104. Springer-Verlag, 2006.
[12] Ting Su and Jennifer G. Dy. In search of deterministic methods for initializing k-means and gaussian mixture clustering. *Intell. Data Anal.*, 11(4):319–338, 2007.
[13] Dimitris K. Tasoulis, Niall M. Adams, and David J. Hand. Unsupervised clustering in streaming data. In *ICDMW '06: Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops*, pages 638–642, Washington, DC, USA, 2006. IEEE Computer Society.