Auto-Clustering of Financial Reports Based on Formatting Style and Author's Fingerprint *

Braulio C. Blanco Lambruschini[1][0000−0002−5654−0212], Mats Brorsson[1][0000−0002−9637−2065], and Maciej Zurad[2]

[1] SNT - University of Luxembourg
{braulio.blanco, mats.brorsson}@uni.lu
[2] Yoba S.A., Luxembourg
maciej.zurad@yoba.com

Abstract. We present a new clustering algorithm of financial reports that is based on the reports' formatting and style. The algorithm uses layout and content information to automatically generate as many clusters as needed. This allows us to reduce the effort of labeling the reports in order to train text-based machine learning models for extracting person or company names, addresses, financial categories, etc. In addition, the algorithm also produces a set of sub-clusters inside each cluster, where each sub-cluster corresponds to a set of reports made by the same author (person or firm). The information about sub-clusters allows us to evaluate the change in the author over time.

We have applied the algorithm to a dataset with over 38,000 financial reports (last Annual Account presented by a company) from the Luxembourg Business Registers (LBR) and found 2,165 clusters between 2 and 850 documents with a median of 4 and an average of 14. When adding 2,500 new documents to the existing cluster set (previous annual accounts presented by companies), we found that 67.3% of the financial reports were placed in the correct cluster and sub-cluster. From the remaining documents, 65% were placed in a different subcluster because the company changed the formatting style, which is expected and correct behavior. Finally, labeling 11% of the entire dataset, we can replicate these labels up to 72% of the dataset, keeping a high feature coverage.

Keywords: Clustering, NLP, Machine Learning, Unstructured data, Financial Reports.

1 Introduction

In many countries, public companies are required to submit and disclose financial reports promoting transparency and security of business trading. The most commonly required type of document is the Annual Accounts where the company publishes the results of the business performance during one fiscal year[3].

* This work has been partly funded by the Luxembourg National Research Fund (FNR) under contract number 15403349.

[3] We are here describing the situation in Luxembourg but similar structures exist in many other countries.

The annual account of a company consists of a *balance sheet* and an *annex*. The balance sheet is formal and follows the same style and format for all companies as it is filled electronically. The annex, however, describes the company's operation in natural text and can differ significantly from company to company depending on the person or firm drafting the reports. Nevertheless, the information in the annexes contains valuable information that we would like to analyze.

In order to be able to use the annexes in supervised learning algorithms, we need to label the data. Unfortunately, dataset labeling requires a huge effort and we need to find ways to reduce it.

We propose in this paper an automatic clustering method grouping documents (the annual account annexes) in clusters with the same sections/subsections and similar content so that it would be enough to label meaningful information in just a very small subset inside each cluster and then simply replicate automatically these labels in each single document for each cluster. For example, some labels can help us to identify person or company names, addresses, positions or roles, workforce information, financial/accounting categories/subcategories, phrases denoting uncertainty or financial risk, etc. We found that with this methodology if we do manual labeling of 11% of the dataset, we could automatically replicate these labels in 72% of the complete dataset, reducing significantly the labeling effort, covering 92% of the subtitles features.

Furthermore, inside each cluster of reports we do sub-clustering to group even more similar reports (sharing the same document template). This similarity represents the *author's fingerprint* and implies that in a sub-cluster we find all the reports made of a single author to multiple companies. This use case is a common practice in financial reports. The author can here represent a person or an accounting firm and we have confirm that they tend to use the same document template and writing style in all of their reports for different companies.

Our methodology builds upon a *clustering algorithm* fed by format- and content-based features extracted from the annual account annexes that creates as many clusters and sub-clusters as needed depending on the threshold parameters.

We are using public annual accounts submitted to the Luxembourg Business Registers and are able to use both digital native documents as well as the ones where the PDF documents contain scanned pages.

Our work contributes insights on how to learn the similarity of the financial reports based on the formatting style and author's fingerprint and how to cluster them together. Most of the uses cases for applying this algorithm in Financial reports are directly or indirectly related to Internal and External Risk Analysis business processes. This clustering information can allow us to:

- reduce and simplify the labeling effort for training supervised Deep Neural Networks (DNN),
- identify authors (accounting firms) that attend to multiple companies; for external risk analysis like analysis of possible fraudulent behavior,
- train machine learning models that can discard all the common phrases among the documents in a sub-cluster and identify the meaningful information, and to

– analyze the behavior of a company with respect to their accountant (changes over time) for internal risk analysis.

The algorithm is used in a larger project where the aim is to build an automated credit risk platform for small and medium sized enterprises (SME). The algorithm is not limited to be applied to Financial Reports but these documents fit perfectly in the main scenario: thousands of templates and thousands of hidden authors using similar terms for reporting (detailing them with their own writing style). Also considering that different companies can share the same hidden author. It can, however, be generalized and used in any application where it would be useful to find similarities (and differences) in form and content between documents and cluster the documents based on these similarities.

2 Related Work

We have concentrated our research on Document Analysis, which in turn is focused on processing whole documents in contrast to short-text processing. In a longer document you typically find the use of many typesetting features such as different text sizes, various styles such as bold, italics, underline, etc, and whether or not the document will have features such as cover page or table of contents. The position, and style of page numbers, logos, text footers and so on also provide a *Document Class-Specific Knowledge* [2] and constitute a fingerprint of the author of the document as the same author tends to use the same style of writing for all documents of the same kind.

Existing approaches for document clustering like k-means, and k-medoids, all need the number of clusters in advance, therefore they are not applicable for us as we cannot know this beforehand. Other approaches more oriented to Natural Language Processing were proposed like Latent Dirichlet Allocation (LDA) [4], and Document-Term Matrix (DTM) [16]. However, these methods mostly suffer from the same issue. Even though DBSCAN [7] does not require to know the number of clusters in advance; but for this case, the number of features could be infinite (features are generated based on the content and format that can have huge amounts of variations according to the template used by the company and the modifications made by the author). In addition to that, for checking dense regions[4] we need to compare each document with the whole dataset, having a cubic complexity (comparison also at feature level).

The algorithm Hierarchical Agglomerative Clustering (HAC) [5] is a good option for similar clustering tasks, but the main problem is that each new document is added to the closest document (edge growing) and at the end the distance of two edge documents can be big; or if we grow vertically, the resulting dendrogram can be too complex to understand the results.

Xu et al. [16] performed document clustering by using Non-Negative Matrix Factorization (NMF) of the document-term matrix. The algorithm assigns one of the k topics to a document. Here the authors are filtering out stop words

[4] A dense region is a big group of elements that were plotted together in all dimensions.

and applying stemming. In our case, we are only replacing numeric and date information with the [NUMBER] and [DATE] tags.

Most of the current algorithms are focused only on the evaluation of the content for topic analysis. As we are dealing with clustering within a single topic, we need to consider format features additionally to the content ones.

The use of graphs is another approach proposed by some authors. Hamza et al. [9] use graph-matching in order to cluster administrative documents. Unlike other approaches, they create as many clusters as required and is not topic-clustering oriented. This algorithm uses content and relative position of each document to create a set of features. They use graph-probing distance to evaluate if a document is going to be added to an existing cluster or if it will create its own cluster. It is also important to mention that this algorithm adopts incremental learning, hence, there is no need for retraining. Inspired by this approach, our algorithm also compares a document with a set of existing clusters based on a list of text-oriented features. Compared to Hamza et al [9], we added three posterior phases, which allow us to work with large datasets, to remove insignificant features and to perform a second clustering step inside each cluster. Our algorithm considers that each feature can contribute distinctively, depending on how many documents shares each feature. Each time a cluster accepts a new document, it becomes more robust because of the shared features.

The selection of a distance function or similarity score is one of the most important aspects of any clustering algorithm. Common distance functions are Manhattan, Euclidean, Cosine, Mahalanobis, etc. These functions need numeric values for calculation. In our case, the comparison of features are text-based features, consequently, these functions are not suitable for us. Transforming the features into embeddings or to do a one-hot enconding are also not feasible as the number of features can be infinite. The graph-probing distance proposed by Hamza et al. [9] computes the distance between two graphs based on the frequency (freq) of the edges between the nodes of each cluster. Our approach uses a similar metric that is the feature-set similarity score, which considers the contribution of each feature according to a confidence score. This confidence score highlights the feature's importance (which is automatically updated by the algorithm during the process). See section 4.1 for more information.

As the first evaluation method, we are going to check a small part of the dataset and manually assign the corresponding cluster and sub-cluster. Once we have the labeled dataset, we need to specify the metrics to be used for evaluation, as shown by Palacio-Niño & Berzal [13], the commonly used distance metrics are Jaccard, Recall, Precision, Accuracy, Hubert statistics and Rand Index [15]. We use Rand Index (RI) because it measures the correct assignment of an element into a cluster as shown in Equation 1, where a is the number of pairs of elements belonging to the same cluster, and b is the number of pairs of elements belonging to different clusters, according to the training dataset (T) and labeled dataset (L) with a total of n elements. The second evaluation method consists in adding more reports from existing companies (previous years) and analyze the true-positive ratio of the clustering.

$$RI(T, L) = \frac{a + b}{n(n-1)/2} \tag{1}$$

3 Dataset

Our data was obtained from the Luxembourg Business Registers[5], which is a public database for company data in Luxembourg. The information available at the LBR contains annual accounts and their modifications, court orders (like bankruptcy, judicial dissolution, temporal administration assignation, etc.), registration and so on.

As mentioned before. the Annual Accounts consists of structured or table-oriented financial statements like balance sheets and profit and loss statements, but also include an annex in free-form text that is not subject to a specific document template. The annexes can be submitted in French, German or English where French is the most common consisting of about 85% of the documents. Examples of annexes are shown in Figure 1 and Figure 2.

The format of the annexes can be quite different. However, we have empirically found that some of the annexes are more similar than others and we conclude that they are prepared by the same author or accounting firm which uses a template for all (or several) or their customers. A clustering algorithm can group similar annexes, this allows us to label a small sub-set of documents in a cluster and then replicate these labels to the entire cluster.

We have downloaded 53,210 Annual Accounts from 2015 on-wards from LBR. In table 1 we can see some statistics about our dataset. The total of working pages is obtained removing empty pages and financial statements pages from our dataset.

Table 1. Full dataset statistics.

	Full dataset	Last year dataset
Number of business types	452	452
Number of companies	38,644	37,999
Number of reports	53,210	38,455
* Number of reports in French	46,191 (86.8%)	32,943 (85.7%)
* Number of reports in German	4,214 (7.9%)	3,209 (9.3%)
* Number of report in English	2,805 (5.2%)	2,303 (6.0%)
Total of pages	375,544	277,798
Total working pages	195,546 (52.1%)	144,895 (52.1%)

The *last year dataset* is considering only the most recent Annual Accounts presented by a company. A company can present two Annual accounts the same year in case there is a need for a rectification.

For validating our algorithm's performance we need to work with a labeled dataset. There are many labeled datasets available for Document Analysis, like DocBank [11], PubLayNet [18], DSSE-200 [17], ICDAR2015 [3], DocVQA [12],

[5] `https://www.lbr.lu`

Web Document dataset [14], TDT2 [8], TTC-3600 [10], and Reuters-21578 [1]. More datasets can be found in UCI Machine Learning repository [6]. These datasets could be used for different kinds of tasks like text extraction and labeling [3, 11, 17, 18]; for question answering based on images [12]; for document clustering [1, 8, 14] and so on. Most of them only have the text available and not the document itself.

None of the previous datasets can be used to identify clusters based on the formatting style and/or author's fingerprint. For this reason, we have labeled a small part of our dataset in a semiautomatic way (explained in the *Evaluation* section). This labeled test dataset is composed of 1,000 documents and more details are given in table 2.

Table 2. Test Dataset statistics.

Number of documents	1,000
* Number of documents in French	905 (90.5%)
* Number of documents in German	91 (9.1%)
* Number of documents in English	4 (0.4%)
Number of business types	193

4 Auto-Clustering algorithm

Our algorithm is a *two-level auto-clustering algorithm* for financial reports. The first level is a template-oriented clustering and the second level is an author's fingerprint-oriented clustering.

There is no need to specify the number of clusters but a few threshold parameters needs to be specified, see table 3. The code for the clustering algorithm and associated tools are publicly available in github[6].

Table 3. Threshold parameters used in the algorithm.

Threshold Parameter	Notation	Explanation
Clustering	κ_c	Min similarity for appending a document into a cluster.
Merging	κ_m	Min similarity for considering merging two similar clusters.
Sub-clustering	κ_s	Min similarity for appending a document into a sub-cluster.

Preprocessing For each document, the algorithm needs the extracted text and some meta-data for each page. We have used a tool we developed for this process[7] which uses PyTesseract[8] to extract text from PDF documents (including PDFs based on scanned pages) and most of the meta-data.

[6] https://github.com/Script-2020/autoclusteringFinReports
[7] FDExt: Available in the repository
[8] `https://pypi.org/project/pytesseract/`

Algorithm 1 Main autoclustering algorithm.

function MAIN_CLUSTERING(D, S_c, S_m, S_s) ▷ D:List of documents,κ_c: clustering threshold,κ_m: merging threshold,κ_s: sub-clustering threshold
  $C, Id_{max} \leftarrow \{\}, 0$ ▷ Phase I: Generation of candidates
  for each $d \in \mathcal{D}$ **do** ▷ Can be done in parallel
    $C, Id_{max} \leftarrow add\_document\_to\_cluster(d, C, \kappa_c, Id_{max})$
  end for
  $C \leftarrow do\_feature\_cleaning(C)$ ▷ Phase II: Feature Cleaning
  $C \leftarrow merge\_clusters(C, \kappa_m)$ ▷ Phase III: Cluster Merging
  for each $c \in \mathcal{C}$ **do** ▷ Phase IV: Sub-clustering
    $Sc, Sc_{idmax} \leftarrow \{\}, 0$
    for each $d \in c.documents$ **do**
      $Sc, Sc_{idmax} \leftarrow add\_document\_to\_cluster(d, Sc, \kappa_s, Sc_{idmax})$
      $c.subclusters.add(Sc)$
    end for
  end for
return C
end function

Our algorithm consists of four phases: (I) Generation of cluster candidates, (II) Cluster feature cleaning, (III) Cluster merging and, (IV) Sub-clustering.

Algorithm 1 shows in detail how these phases are sequenced.

4.1 Phase I: Generation of cluster candidates

The algorithm to generate cluster candidates uses a set of format-based features and a set of content-based features. Our OCR extractor tool provide us the language, page orientation, text bounding box and the text. The format-based features are:

- Language (French, German or English).
- Page orientation (portrait or landscape).
- Horizontal position of each line (frequency ≥ 2).
- Text line width and height (frequency ≥ 2).
- Enumerator patterns (eg. 1. [arabic point], I) [roman parenthesis], etc.).

We include only format features for lines that have at least two values being the same (frequency ≥ 2). Language and Page orientation features are used for clustering while position, text width and height are used for sub-clustering. For enumerated text lines, we replace the enumerators with their corresponding pattern (e.g. arabic, cardinal, letter). The enumerators are extracted using regular expressions to identify alphanumeric, roman or mixed sequences and its corresponding separators like dots, colons, dashes, or parenthesis. Enumerators patters are used for clustering.

For the *content-based feature set*, the algorithm adds subtitles as features. A text line is identified as a subtitle if it starts with an enumerator pattern. The

Algorithm 2 Add a document into a existing cluster or create a new cluster

```
function ADD_DOCUMENT_TO_CLUSTER(d, C, κ_c, max_id)  ▷ d:document, C: Map of clusters, κ_c: clustering threshold, max_id: Max id in the map C
    f_d ← get_features(d)
    found ← False
    for each c ∈ C do
        f_c ← c.features
        χ ← calculate_similarity(f_d, f_c)
        if χ ≤ κ_c then
            c.features ← merge_features(f_d, f_c)
            c.confidence ← update_confidence(c)
            c.documents.add(d)
            found ← True
        end if
    end for
    if ¬found then
        max_id ← max_id + 1
        C[max_id].features ← f_d
        Cmax_id].documents.add(d)
    end if
return C, max_id
end function
```

next consecutive line following a sub-title is added also as a text-line feature. Sub-titles are used for clustering while text-line features are used for sub-clustering.

The algorithm begins iterating document by document, to assign a proper cluster or create a new one if the document does not fit into any existing cluster (Algorithm 2). At first, as there are no clusters, the first document creates a cluster of one document (itself), the document's features are copied to the cluster. The iteration continue with the following document.

The candidate document is going to iterate through the existing clusters and the document's features are going to be compared with the cluster's features. If the feature-set similarity score (χ) of both set of features is bigger than the clustering threshold (κ_c), the document is included into the cluster and the cluster's features are updated by merging in the features of the document.

The feature merging (method $merge\_features$ in Algorithm 2) consists of adding the new document's features to the cluster's features and update the *confidence score* of each cluster's feature. When the cluster is created, each feature has a confidence score equal to 1 (transferred directly from the single document). When a new feature is added to the cluster, this confidence score is updated. For example, if a cluster of one document has a initial confidence score of 1, and the new document also has the same feature, the confidence score will remain 1, but in case the new document does not have this feature, the new confidence score is equals to 0.5. For a bigger cluster, the feature's confidence score before merging is equals to 0.2 in a cluster of 20 documents (this means that 4 of these documents share the feature). In case the document is including an unseen fea-

ture to this cluster, the feature's confidence score is 0.048 (1/21). The feature's confidence score (σ) is part of the similarity score (χ). The feature's confidence score allows the cluster to focus in the most important features that are more common to most of the documents.

Our similarity metric is the *feature-set similarity score* (χ). Is based on the graph-probing distance but considering the feature's confidence score (σ). The *feature-set similarity score* (χ) is shown in equations 2 and 3. χ is used for comparing the similarity of one cluster with a document (phase I) or between two clusters (phase III).

$$avg_{features}(A, B) = (|f_{(A)}| + |f_{(B)}|)/2 \tag{2}$$

$$\chi(A, B) = \sum_{i=0}^{|f_{(AUB)}|} \begin{cases} \sigma_i / avg_{features} & , \exists f_i \varepsilon A \wedge f_i \varepsilon B \\ 0 & , \text{otherwise.} \end{cases} \tag{3}$$

Where $|f(A)|$ is the number of features in the cluster A, $|f(B)|$ is the number of features in the document or cluster B; σ_i is the confidence score of the i^{th} feature in $f_{(AUB)}$ (the union of features of A and B).

When a new document is added to the cluster, the features are merged and the confidence score of each feature is updated as shown in equation 4.

$$\sigma_{(f_{t+1})} = (\sigma_{(f_t)} * n_t + 1)/n_{t+1} \tag{4}$$

The new confidence score for each feature in the cluster $\sigma_{(f_{t+1})}$ is calculated by multiplying the current confidence score $\sigma_{(f_t)}$ with the current number of documents (n_t) plus 1, all divided by the number of new documents (n_{t+1}).

Additionally, the sparsity measure of the cluster is calculated. This metric is the cluster confidence (τ_c) and as shown in Equation 5, is the average of all its features' confidence score (σ).

$$\tau = \sum_{i=0}^{n} (\sigma_{(f_i)})/n \tag{5}$$

4.2 Phase II: Cluster's features cleaning

In this phase, each cluster is analyzed and the less confident cluster's features are pruned. Only features with a confidence score $\sigma_{(f)}$ less than a forgetting threshold (κ_f) are kept. κ_f allows to remove features that are only in one or two documents inside the cluster. This cleaning phase is applied only in clusters with more than 4 documents. We empirically found that setting $\kappa_f = 1/n_{docs}$ for clusters with 10 documents or less and $\kappa_f = 2/n_{docs}$ otherwise, provided good results. As some features could be removed in this process, the cluster's confidence (τ_c) is recalculated.

Algorithm 3 Cluster merging

```
processed_pairs ← {}
function MERGE_CLUSTERS(d, C, κ_m)          ▷ d:document, C: Map of clusters
    adj_matrix ← get_similarity_matrix(C, κ_m)
    high_confident_pairs ← get_highconfident_pairs(adj_matrix)
    for each pair ∈ high_confident_pairs do
        if pair ∉ processed_pairs then
            recursive_pairs ← get_recursive_pairs(high_confident_pairs, pair[1])
            for each pair_r ∈ recursive_pairs do
                if pair_r ∉ processed_pairs then
                    C[pair[0]].features ← merge_features(pair[0].features, pair[1].features)
                    C[pair[0]].confidence ← update_confidence(C[pair[0]])
                    C[pair[0]].documents.extend(C[pair[1]].documents)
                    C[pair_r[1]] = Null
                    processed_pairs.add(pair_r)
                end if
                processed_pairs.add(pair)
            end for
        end if
    end for
return C
end function
```

4.3 Phase III: Cluster merging

In this phase we improve on the candidate clusters from phase I, merging similar clusters. Documents that are similar to other documents but were placed in other clusters because of the parallel processing are merged together into a single cluster. For doing this instead of comparing the features of one document with one cluster, features are compared between two clusters and the merging threshold (κ_m) is used as parameter.

In this phase we are creating k_m groups of clusters to be distributed. Each group with N clusters is going to create a $N \times N$ similarity matrix, where each cell is the feature-set similarity score (χ) between a pair of clusters. Then all the high-confident pairs are retrieved ($\tau_c \leq \kappa_m$). Then, each pair is analyzed, and per each pair, all its high-confident clusters are also retrieved in a recursive way. This allow us to get a chain of similar clusters to merge. Afterwards, we merge all the chain of clusters, merging their features and the corresponding cluster confidence (τ_c). We use equation 4 but instead of adding one, should be the product of the feature's confidence of the second cluster times the number of documents in that cluster.

4.4 Phase IV: Sub-clustering

In this phase we repeat the algorithm used for the generation of candidates, but this time is done cluster by cluster and only considering the documents inside the cluster. The parameter is the sub-clustering threshold (κ_s).

5 Evaluation

The clustering task usually falls under the unsupervised Machine Learning category, but in this case, for our first evaluation method, we are going to use a subset of the data to evaluate the performance of the clustering algorithm and measure the success of the algorithm. For doing this we need to label a test dataset as mentioned before.

Our labeling method consists of running our algorithm with lower threshold parameters (κ_c & κ_m) in order to get similar documents together and then do manual check. This labeling was done manually by a single person to achieve consistency. The cluster and sub-cluster are assigned to each document (the assigned sub-cluster number is subjective because is not too strict, allowing for certain differences between documents). The main rules for considering to put two documents into the same cluster are: i) if both are using the same document template, and ii) they use the same words in the subtitles, left alignment and so on.

Two documents in the same cluster are not going to be in the same sub-cluster if they do not share the same format features such as: the same cover page, table of contents, size, and location of the text etc. Some details can vary if it is clear for the labeler that the same person/company used that template. To make this labeling effort easier, we generate a report of cluster's similarity, where we include for each cluster the list of cluster which were not so similar to be merged but the similarity is around 50% (not merged clusters with κ_m less than 0.50), this allow us to check manually clusters discarded by the algorithm that should be merged. This manual assignation is going to be considered as our *ground truth*.

The final labeled dataset has 1,000 documents divided into 77 clusters and 175 sub-clusters in total. The biggest cluster contains 86 documents divided into 19 sub-clusters. The biggest sub-cluster has 27 documents. On average each cluster has 12 documents. 66% of the sub-cluster contains less than 6 documents and 13% of the sub-clusters contains more than 10 documents. 50 clusters with only one sub-cluster with an average of 8 documents each. Other metrics are shown in table 4.

Table 4. Metrics for ground truth labeled dataset.

Measure	Max	Min	Average	Median	Mode
Clusters (doc/cluster)	87	4	12.7	8	5 (14)
Sub-clusters (doc/sub-cluster)	27	1	5.6	4	1 (45)

The Rand coefficient or Rand Index is the metric used for comparing similarity of correct classified pairs over the total of pairs, and this can be used to compare the ground truth with the results of our algorithm.

For our second evaluation method, we are going to train the model considering only the Annual Accounts presented by the company in the last year (Last

year dataset) and then order by cluster size and do a manual checking of the first 1,500 documents. The results of this evaluation is the true-positive ratio for clustering. Later, we are going to append documents to the existing model, which contains previous reports presented by the company (remaining reports from the Full dataset) and then check if all the documents from one company are in the cluster and sub-cluster. This is also measured as true-positive ratio for sub-clustering or author's fingerprint.

For labeling purposes, based on the final results, we define the number of documents to be labeled (n_{lab}), and the $feature\_coverage$] (in total how many features can be labeled with respect to the total of features). The selection of documents to be labeled is first done by selecting the document that contains most of the shared subtitle features. The next selected document in the cluster is the next one which contains most of the remaining features. This iteration continues until all of the features were covered or a minimum threshold ($min\_feature\_coverage\_ratio$) has been reached or a maximum number of iterations were done ($max\_iterations$). Only are considered clusters with more than a minimum number of documents ($min\_docs\_threshold$).

The $feature\_coverage$ formula is shown in equation 6. It is calculated as the average of the number of labeled features divided by the total number of features in all clusters that has equals or more documents than $min\_docs\_threshold$.

$$feature\_coverage = n\_labeled\_features \div n\_features \tag{6}$$

6 Experiment and Results

We ran the algorithm with our dataset with 38,455 reports. The following values were used as threshold parameters: κ_c (clustering) = 0.20, κ_m (merging) = 0.30, κ_s (sub-clustering) = 0.17.

Different threshold values were tested, these were the best comparing the 1,000 reports in our test dataset and the Rand Index (RI). As shown in Table 5, the best Rand Index for clustering was 87% and 55% for sub-clustering. The lower RI for sub-clustering is mainly because some documents should be placed in the same sub-cluster according to the labeler because it looks like the same, but in terms of content there are some mistypos or more additional information.

Table 5. Rand Index (RI) results

Metric	Clustering	Sub-Clustering
Rand Index	87%	55%

As shown in Table 6, 6,119 clusters were found, 35.4% corresponds to clusters with 2 documents or more, representing 81.3% of the dataset. 10.3% creates a cluster of one (itself) and 8.4% documents were discarded because they have less than 5 features (usually scanned documents with few lines of readable text). If

we consider only clusters with a cluster confidence τ_c bigger than 0.85, we got that 66.9% of the documents were clustered with a high confidence.

Table 6. Auto-Clustering Results

Total clusters	One document	More than 2 docs		discarded	High confident clusters	
		clusters	documents		clusters	documents
6,119	3,953	2,165	31,264	3,238	2,070	25,735
	10.2%		81.3%	8.4%		66.9%

It is important to notice that because of the content features and language features, all the documents within a cluster are the in same language. At this point and for our goals, there is no need to translate into a single language and then process it, even more because our main language is french and the current tools are not well developed in French.

In figure 1 we see an example of two reports that belongs to the same cluster but in different sub-cluster. The information in the two documents is quite similar, but there are some differences. Also for this specific example, the first document has a cover page and the second one does not.

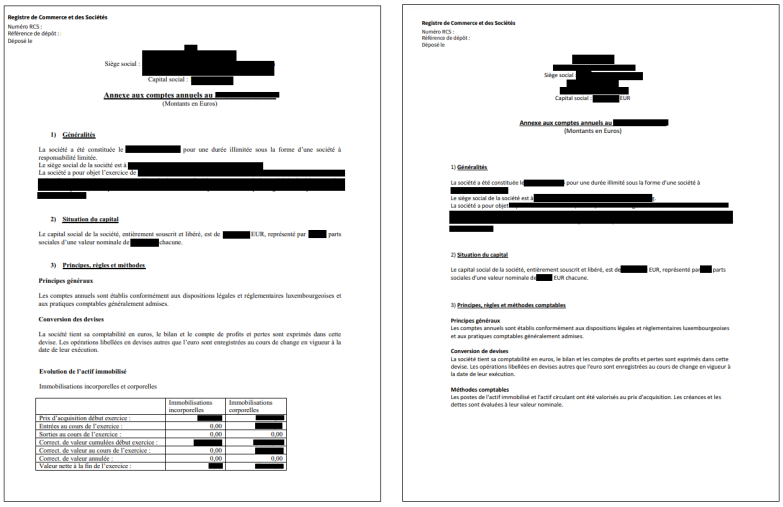

Fig. 1. Two reports from the same cluster but different sub-cluster.

On the other hand, in figure 2, we can see that all the reports belongs to the same cluster and the same sub-cluster, this is the author's fingerprint. Basically what is changing is some part of the content (different specific company-related

information), but the layout and words to present the information is quite the same.

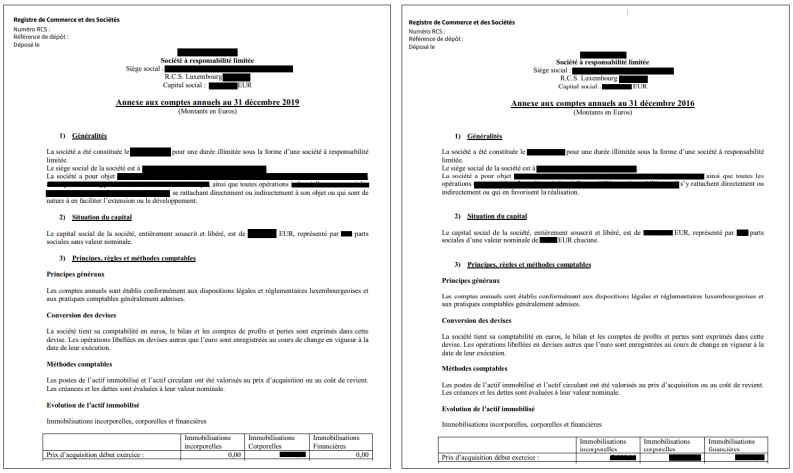

Fig. 2. Two reports from the same sub-cluster: author's fingerprint.

Checking only the clustering level, we have checked manually 1,322 reports distributed into 37 clusters where the biggest cluster has 850 reports and the smallest cluster has only 4 reports. We found that only 9 reports were placed in the wrong cluster, having a true-positive ratio of 99.3%

We have checked manually, at a cluster and sub-cluster level, 502 reports where we found 122 sub-clusters with an average of 4 reports per each sub-cluster. The algorithm could place correctly 378 reports in its corresponding sub-cluster, reaching a true-positive ratio of 75.3%. We identified that in some cases two sub-clusters had to be considered in the same sub-cluster because analyzing the report visually, looks alike, but usually one of the groups does not change the format but increase the financial concepts to be described, when this changes overpasses the threshold value, it splits into two different sub-clusters.

Considering the dataset which includes only the last presented report (38,455 documents), we use different threshold parameters for obtaining the documents to be labeled. The results are shown in Table 7. We specified $max\_iterations = 10$, but none of the clusters need more than 4 documents to be labeled.

In consequence labeling $n_{lab} = 4{,}190$ documents (10.90%) will allow us replicate up to 27,799 documents, that represents (72.29% of the entire dataset), having a very good feature coverage (92.80%).

Finally we append to the existing model 2,500 new documents from 799 companies. One document from each of these companies were fed into the model during the training phase. From all these new documents, 67.3% of the financial reports where placed in the right subcluster. For the remaining documents, we

Table 7. Number of documents to be labeled

min_docs_threshold	2 (81.3% of the dataset)				3 (72.29% of the dataset)			
min_feature_coverage_ratio	0.0	0.1	0.2	0.3	0.0	0.1	0.2	0.3
n_{lab}	7,694	6,556	5,902	5,749	6,360	5,091	4,365	4,190
% dataset (38,455)	(20.01%)	(17.05%)	(15.35%)	(14.95%)	(16.54%)	(13.24%)	(11.35%)	(10.90%)
feature_coverage	99.34%	97.51%	95.65%	95.03%	99.04%	96.39%	93.70%	92.80%

analyzed manually 100 reports chosen randomly, and we found that 65% of them were placed in a different subcluster because the company had changed the formatting style over the time.

For analyzing the contribution of each phase in the model's outcomes, we kept only the first phase and remove a phase for each test case. Removing the cleaning phase does not change too much the results, only affects the quality of the features, removing noisy features(15%). Removing the cluster merging phase we end up with 3.3 times more clusters. The contribution is significantly but can be removed if the first phase is not distributed, in this sense the contribution of this phase is zero because the number of clusters are almost the same[9]. Removing the sub-clustering phase does not have any impact in the clustering because it operates inside each cluster. The contribution at template-level clustering is zero but very high for author's fingerprint-level clustering.

7 Conclusions and following steps

We have demonstrated that our algorithm, using content and layout features, is able to cluster different financial documents and even to identify the author,s fingerprint. There is no need to define the number of clusters.

With a semi-manual labelling of about 11% of the dataset, we can use the clusters from the algorithm to replicate them up to 72% of the dataset, effectively reducing the labeling effort.

The sub-clustering results when a company has different documents in different clusters can be later analyzed the whole dataset with a Temporal Machine Learning Model to classify if the author has changed the formatting style or if the company has changed of author.

This algorithm can be generalized to other domains differentiating the content-based features with the format-based features, where the first group should have a bigger weight. This allows to cluster documents focused on the text more than the format. For example, applying in internal documents provided for the different departments in a company, to cluster documents inter-departmental documents to label for training machine learning models. Similarly can be applied to reports from external consultants to look for hidden patterns between them.

[9] This test was done only in a small dataset because it requires more memory resources, for larger datasets the contribution of the phase is very high

References

1. UCI Machine Learning Repository: Reuters-21578, Distribution 1.0, Text Categorization Collection Data Set, https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection
2. Aiello, M., Monz, C., Todoran, L., Worring, M.: Document Understanding for a Broad Class of Documents **5(1)**. https://doi.org/10.1007/s10032-002-0080-x
3. Antonacopoulos, A., Clausner, C., Papadopoulos, C., Pletschacher, S.: ICDAR2015 competition on recognition of documents with complex layouts - RDCL2015 . https://doi.org/10.1109/ICDAR.2015.7333941
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
5. C. D., Manning, P.R., H., S.: Introduction to information retrieval. In: kdd. vol. 1 (2008)
6. Dua, D., Graff, C.: UCI machine learning repository, http://archive.ics.uci.edu/ml
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: kdd. vol. 96, pp. 226–231 (1996)
8. Fiscus, J.G., Doddington, G.R.: Topic detection and tracking evaluation overview. In: Topic detection and tracking, pp. 17–31. Springer (2002)
9. Hamza, H., Belaid, Y., Belaid, A., Chaudhuri, B.B.: An End-to-End Administrative Document Analysis System. In: 2008 The Eighth IAPR International Workshop on Document Analysis Systems. pp. 175–182
10. Kılınç, D., Özçift, A., Bozyigit, F., Yıldırım, P., Yücalar, F., Borandag, E.: TTC-3600: A new benchmark dataset for Turkish text categorization **43**(2), 174–185
11. Li, M., Xu, Y., Cui, L., Huang, S., Wei, F., Li, Z., Zhou, M.: DocBank: A Benchmark Dataset for Document Layout Analysis, http://arxiv.org/abs/2006.01038
12. Mathew, M., Karatzas, D., Jawahar, C.V.: DocVQA: A Dataset for VQA on Document Images, http://arxiv.org/abs/2007.00398
13. Palacio-Niño, J.O., Berzal, F.: Evaluation Metrics for Unsupervised Learning Algorithms, http://arxiv.org/abs/1905.05667
14. Sinka, M., Corne, D.: A Large Benchmark Dataset for Web Document Clustering
15. William M. Rand: Objective criteria for the evaluation of clustering methods **66**(336), 846–850. https://doi.org/10.1080/01621459.1971.10482356
16. Xu, W., Liu, X., Gong, Y.: Document clustering based on non-negative matrix factorization. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval. pp. 267–273. SIGIR '03, Association for Computing Machinery
17. Yang, X., Yumer, E., Asente, P., Kraley, M., Kifer, D., Giles, C.L.: Learning to Extract Semantic Structure from Documents Using Multimodal Fully Convolutional Neural Network, http://arxiv.org/abs/1706.02337
18. Zhong, X., Tang, J., Yepes, A.J.: PubLayNet: Largest dataset ever for document layout analysis, http://arxiv.org/abs/1908.07836