Classification of Unstructured Documents into the Environmental, Social & Governance (ESG) Taxonomy using Spark NLP

by Dr. Alina Petukhova

There is an immense amount of unstructured data generated every day that can affect companies and their position in the market. As this information continuously grows, it's a critical task for decision makers to process, quantify and analyze this data to identify opportunity and risk.
This White Paper demonstrates how to automatically solve this issue for unstructured data sources.

Table of Contents

1. Abstract

There is an immense amount of unstructured data generated every day that can affect companies and their position in the market. As this information continuously grows, it's a critical task for decision makers to process, quantify and analyze this data to identify opportunity and risk. One of the important indicators in this kind of analysis is ESG (environmental, social and governance) rating, which identifies issues for a company in these critical areas. This White Paper does this automatically for documents continuously ingested from over world news. The models have been deployed in production as part of a big data analytics platform of a leading data provider to the financial services industry.

To perform this analysis effectively and process a massive number of data sources, John Snow Labs' Spark NLP has been used to automatically analyze incoming documents and detect material ESG events. The goal of this machine learning pipeline is to automatically identify ESG material events in unstructured data records and tag them correctly.

2. Introduction

John Snow Labs' customer for this solution provides timely insights to the financial services industry. The company's platform continuously ingests information from the Internet.

Fund managers and other financial analysts have begun to focus more on ESG criteria and the preference for ESG-positive stocks is high. However, there are not defined standards and rules for ranking companies for ESG currently.

ESG criteria identifies critical issues for a company's operations that socially conscious investors use to screen potential investments. Environmental criteria consider how a company performs as a steward of nature. Social criteria examine how it manages relationships with employees, suppliers, customers, and the communities in which it operates. Governance deals with a company's leadership, executive pay, audits, internal controls, and shareholder rights. ESG criteria help analysts and portfolio managers to make investment decisions on companies.

3. The Challenge

The goal of ESG classification is to automate the process for analyzing data records to identify ESG issues. Natural Language Processing (NLP) techniques are used to automatically assign ESG tags to the target unstructured data records. Artificial Intelligence (AI) models with properly curated datasets can accomplish this task. The analysis yields data record keyword distribution over the three main criteria: environmental, social and governance. All three are important: Keeping them in mind is important when measuring the sustainability and ethical impact of an investment or choosing who to do business with.

The main stages of the project workstream were:

1. Create a taxonomy of signals and sub-signals that capture when ESG related events happen.
2. Work with content experts to find content for each signal or sub-signal combination for training and validation datasets. Borrow or expand concepts from other sources for the taxonomy and work with content experts to come to a final taxonomy. Create a negative dataset with documents not containing ESG signals.
3. Identify keywords and phrases that can be used to aid content searching to capture data for each signal and sub-signal.
4. Train machine learning models for identifying and tagging title and content body (unstructured text) with actual signals or sub-signal items from the taxonomy.
5. Provide precision and recall metrics for each signal or sub-signal and iteratively optimize the model to achieve desired results.
6. Deploying the models into production so that they become a hardened, reliable, and scalable component of the big data analytics platform.

Due to the large data scope of the project, and the need to deploy the models quickly to production reliably and at scale – John Snow Labs' Spark NLP framework was chosen to train the models. The data annotation, preparation, model creation, experimentation, and deployment were also done by the John Snow Labs team.

During the project, we defined target ESG taxonomy to train AI models. We also labeled the existing dataset and trained machine learning models for identifying and tagging title and content body (unstructured text) with actual labels from the taxonomy.

4. The Dataset

Having enough text data records per tag is critical to accurately train a model. To create labeled datasets, we used publicly available unstructured data records for the last 5 years. Additionally, we added negative datasets containing articles not relating to ESG.

4. 1 The Challenge

Overlapping sub-signals - In general, tagging articles is very subjective to an individual's perspective. In the ESG taxonomy, some sub-signals are very close meaning and can be mislabeled not only by a model, but by a content expert as well. For example, these two sub-signals:

- "Social.EmployeeStandards.DiversityInclusion"
- "Social.HumanRights.Discrimination"

Could be equally justified as per the content record. Because of that, it's important to have detailed instructions from content experts to make sure different annotators share the same ideas on content during the labeling process. Otherwise, model metrics are going to be low and mismatch of tagged articles will be high because of the specific sub-signal interpretation. The main goal during the labeling process is to exclude individual bias for the better model performance.

- **Solution** - To eliminate this problem, articles should be tagged correctly for all sub-signal. For the models, training should use a weighted average of the relevance value for each sub-signal or asn adaptive weighted value if annotators have different level of qualification. The aggregated result will give a better picture of model accuracy.

Low Quality Data - Another challenge for this project is to deal with sub-signals with less tagged articles and articles with corrupted bodies. If models were to be trained on those articles, they will not perform with expected accuracy.

- **Solution** - The data preprocessing consisted in removing short articles and articles with corrupted bodies. Additionally, we analyzed incorrect model predictions to make sure annotators labeled the initial record correctly. To make sure we are not overfitting the model, the random state of train/test/validation split was changed on every data regeneration.

4. 2 Data Exploration

Figure 1 show an analysis of the tagged dataset statistics. Specifically, the number of samples per category and average number of words per record body.

One of the main concerns when developing a classification model is whether the different classes are balanced. This means that the dataset contains an approximately equal amount of each class. In Fig. 1a we can see that the Negative tag created imbalance in the dataset. Additionally, two poorly represented sub-signals:

- "Environment.Sustainability.MaterialsSourcing"
- "Social.CommunityResponsibility.AccessAffordability"

had less, then 50 articles in the dataset, which could affect model results. However, from Fig. 1b we see that the number of words for each sub-signal are much more homogeneous (except for the Negative sub-signal). Overall average is 270 ± 164 words.

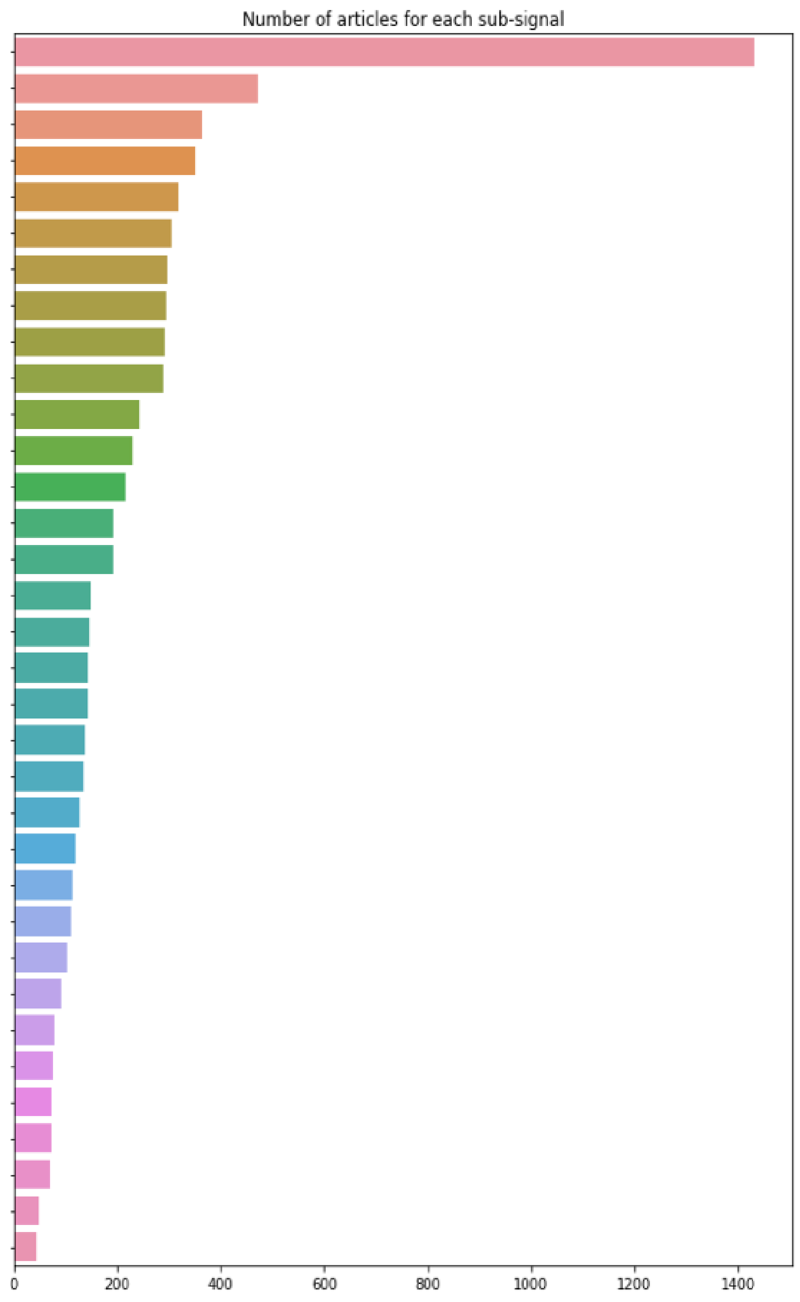


Fig 1a: Statistical analysis of dataset: Number of articles for each sub-signal

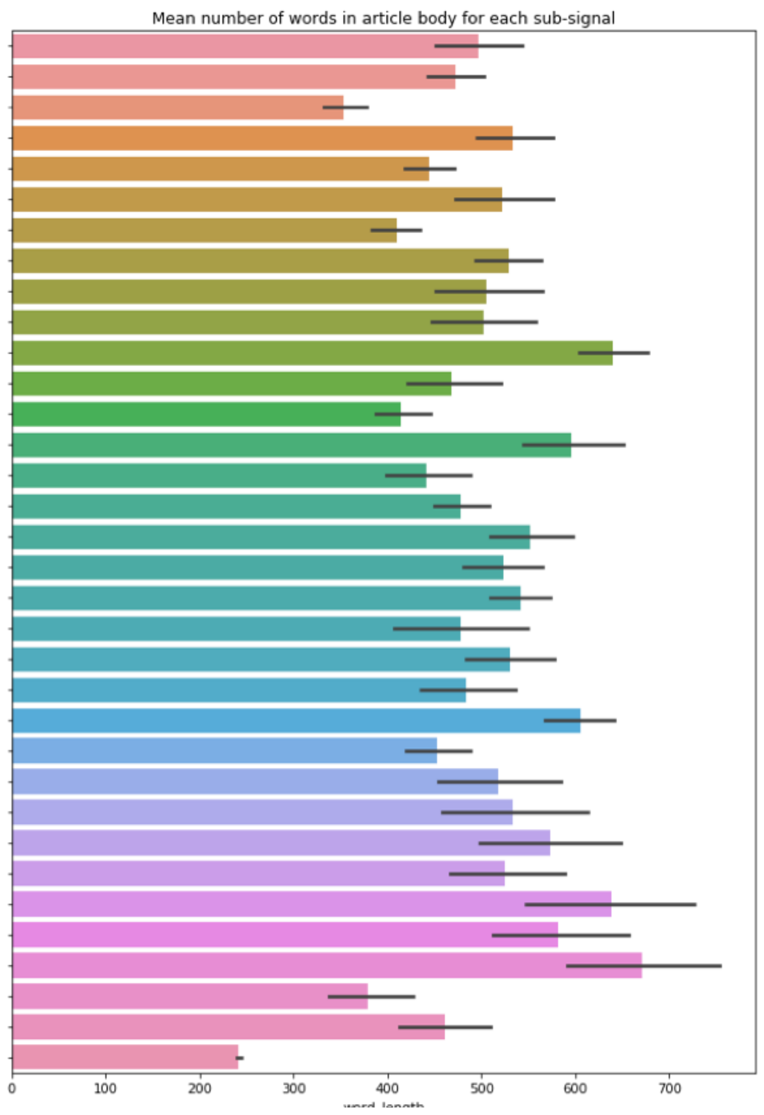


Fig 1b: Statistical analysis of dataset: Mean number of words for each sub-signal

4. 3 Data Preprocessing

Data preprocessing included the following steps:

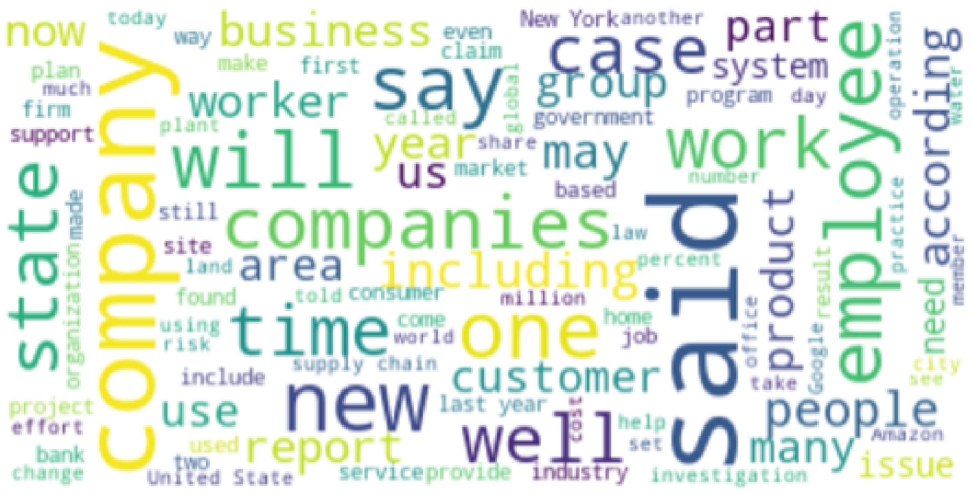
- Processing samples with a body size greater than 600 words - this left us with a total number of samples equal to 7,462 and 33
- ESG sub-signals labels + 1 Negative label.
- Removal of stop words as well as punctuation and HTML tags.
- Lemmatization of each word.

Stop words: commonly used words such as "and" or "the" won't have any predictive power since they are used in all of the articles. A common practice is to remove them to reduce noise in the dataset. Spark NLP provides stop word removal out-of-the-box, and this was extended by adding days of the week, months and numbers.

Lemmatization: Lemmatization is the process of reducing a word to its standardized lemma, which takes into consideration the morphological analysis of the word. In the pipeline we have used public pre-trained models offered by Spark NLP [8].

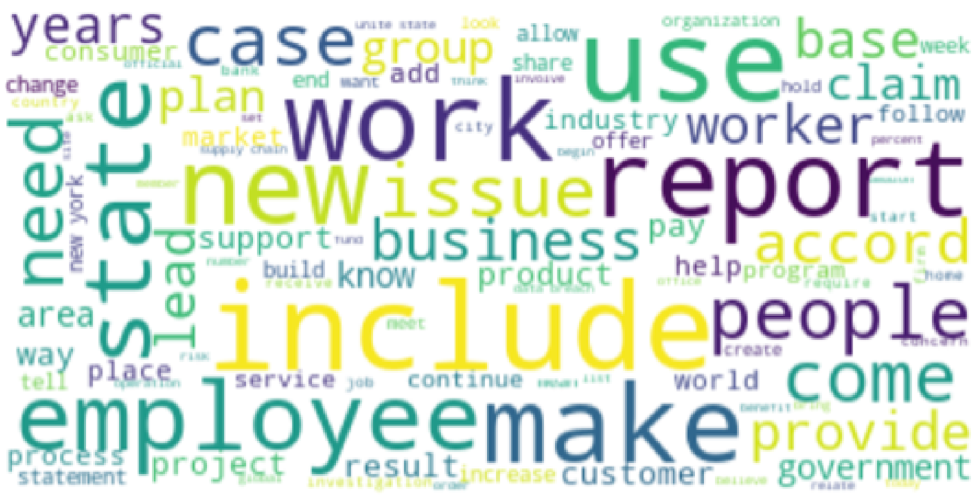
To determine relative prominence of the most prominent terms for each sub-signal and to form a stop words list, we built Word cloud charts. A Word cloud is a collection or cluster of words depicted in different sizes. The idea behind this approach is that each word is placed in the chart based on the frequency of occurrence in a record. The bigger and bolder the word appears, the more often it's mentioned within a record and the more important it is. This approach provides us with some interesting observations regarding the distribution of sub-signal keywords in the labeled collections

Word cloud for dataset overview, Before pre-processing:



All sub-signals

After preprocessing:



All sub-signals

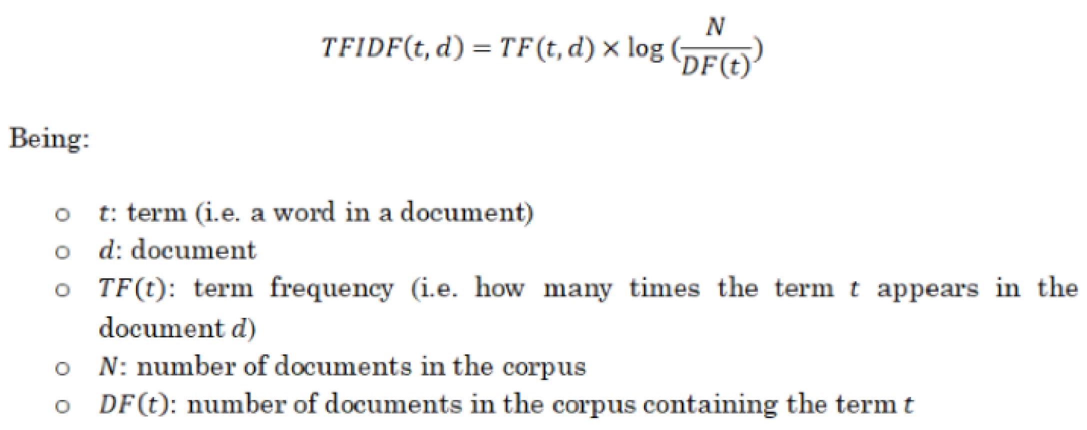
5. Data science

5.1 Feature Engineering & Embeddings

The end goal of the models is to predict the multilabel classification problem and the properties of a data point that are not mutually exclusive, such as sub-signals that are relevant to an unstructured data record.
First, using the preprocessed record titles and descriptions, we created a dictionary of words. The total number of unique words is around 87,000. We then extracted the following word features for the classification task:

• **Word count features:** For count features, we used the first 5,000 most common words for the body field and 7000 for the title field to define the dictionary and then encoded the record titles and bodies as vectors with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

• **Word TF-IDF scores:** TF-IDF Vectorizer transformed the text into features vectors learned from the vocabulary and documented frequency of each word in the training data. For the TF-IDF method we used a dictionary size of 5,000 for body and 7,000 for the title feature.



$$TFIDF(t,d) = TF(t,d) \times \log\left(\frac{N}{DF(t)}\right)$$

Being:

- t: term (i.e. a word in a document)
- d: document
- $TF(t)$: term frequency (i.e. how many times the term t appears in the document d)
- N: number of documents in the corpus
- $DF(t)$: number of documents in the corpus containing the term t

• **Doc2Vec embeddings:** Doc2Vec is a model that represents each article as a paragraph of vectors learned from the training data. It's based on Word embeddings, which is a family of NLP techniques aiming at mapping the semantic meaning into a lower-dimensional vector space using a shallow neural network [1].

Word embeddings produce a set of word-vectors where similar meaning vectors are located close together and word-vectors having a differing meaning are distant to each other. To create embedding for all document we used the Le and Mikolov in 2014 algorithm, which usually outperforms simple averaging of Word2Vec vectors. Doc2Vec has two main implementations:

1. Paragraph Vector - Distributed Memory (PV-DM)
2. Paragraph Vector - Distributed Bag of Words (PV-DBOW)

The main difference between PV-DM and PV-DBOW is the way of calculating the vector of the target word. In PV-DM it's based on an average of both context word-vectors and the full document's doc-vector, but PV-DBOW just uses the full document's doc-vector. Both implementations are obtained by training a neural network.

In our pipeline we combined PV-DM and PV-DBOW implementations as input features for the models. This approach is suggested by the authors [1] and during the evaluation it helped us to improve the final metrics. We trained Doc2Vec embeddings on the vocabulary from the collected dataset. Additionally, we considered only words with a minimum count of 5 for the body and 2 for the title fields and used dimensionality of the feature vectors of 50 and 100, respectively.

We also tried applying pretrained GloVe [2] embeddings (with frozen Embedding layer) but the accuracy in this case was lower than when learning embeddings directly from the data.

In the first part of our work we experimented with traditional machine learning techniques: multinomial logistic regression, Naive Bayes, kernel SVM and Random Forest.

For our implementation, we trained pipelines with several architectures (model types, amount of iterations, penalties, normalization) as well as with different parameters such as an embedding dimension, maximum sequence length and maximum number of words (for words tokenization).

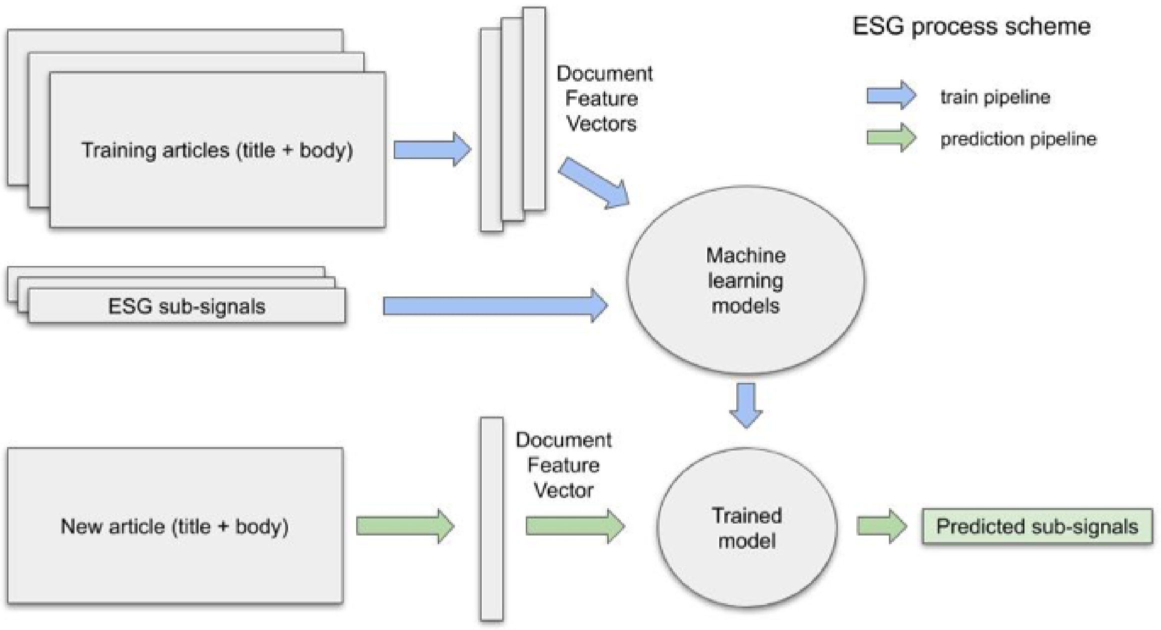


Fig 2: Typical pipeline architecture

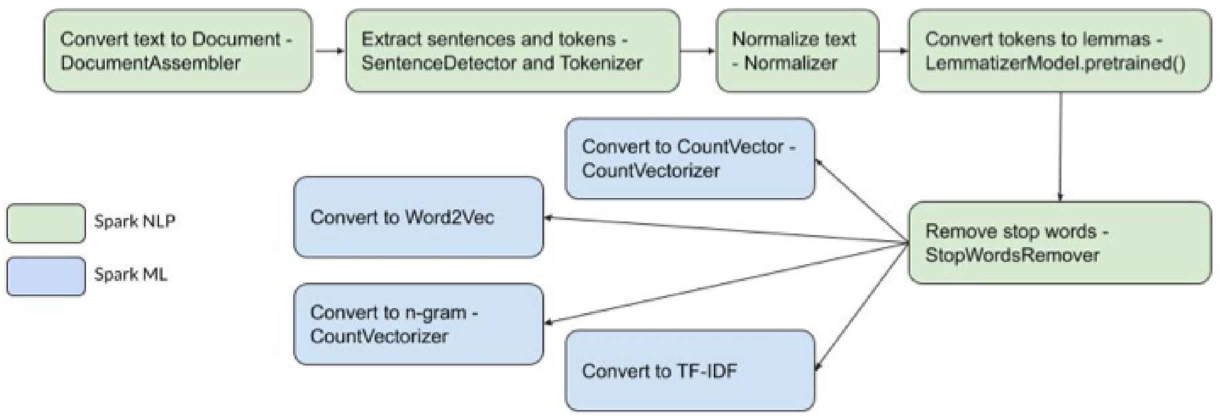

Convert text to Document - DocumentAssembler
Extract sentences and tokens - SentenceDetector and Tokenizer
Normalize text - Normalizer
Convert tokens to lemmas - LemmatizerModel.pretrained()
Convert to CountVector - CountVectorizer
Convert to Word2Vec
Spark NLP
Spark ML
Remove stop words - StopWordsRemover
Convert to n-gram - CountVectorizer
Convert to TF-IDF

Fig 3: Spark NLP and Spark ML stages for text classification

5. 2 Document Classification

Multinomial Logistic Regression was used with cross-entropy loss and L2 regularization, penalizing the model to minimize the cost function [3]:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

Multinomial Naive Bayes used in the training implements the naive Bayes algorithm for multinomially distributed data with an additive smoothing parameter of 0.8.

Random Forest used in the training fits several classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. We used the Gini criterion as a function to measure the quality of a split and 10 estimators and regularized each tree in terms of maximum depth.

Kernel SVM used in the training implements the “one-against-rest” approach with multi-class SVM [4] and Linear kernel. Additionally, we tested the “one-against-rest” decision function with RBF kernel, but this approach didn’t show good results.

All fitting parameters were found by 5-fold cross-validated grid search.

	Count features			TF-IDF features			Word2Vec features		
	Train	**Test**	**Val**	**Train**	**Test**	**Val**	**Train**	**Test**	**Val**
Logistic Regression	0.990	0.807	0.800	0.940	**0.821**	0.794	0.885	0.767	0.749
Naive Bayes	0.912	0.770	0.740	0.712	0.623	0.611	N/A	N/A	N/A
Random Forest	0.987	0.606	0.595	N/A	N/A	N/A	0.873	0.493	0.484
Kernel SVM	0.819	0.739	0.730	0.883	0.826	**0.810**	0.802	0.761	0.744

Table 2. Results for initial architecture with f1 score metric

5. 3 Hierarchical Classification

To improve model accuracy during the project we decided to apply a hierarchical approach [5] for the ESG problem, since it's naturally cast as a hierarchical classification problem where the classes to be predicted are organized into a tree class hierarchy. To implement this approach, we represented every ESG signal as combination of top-level tag - "area", second level tag - "signal" and third level tag - "sub-signal".

In the initial models, we implemented predicting only the leaf nodes of the tree. This approach behaves like a traditional classification algorithm during training and testing. However, it provides an indirect solution to the problem of hierarchical classification, because when a leaf class is assigned to an example, one can consider that all its ancestor classes are also implicitly assigned to that instance. Opposite to that, we applied the Local Classifier Per Parent Node Approach, which trains a multilabel classifier for each node of the class hierarchy (leaf nodes).

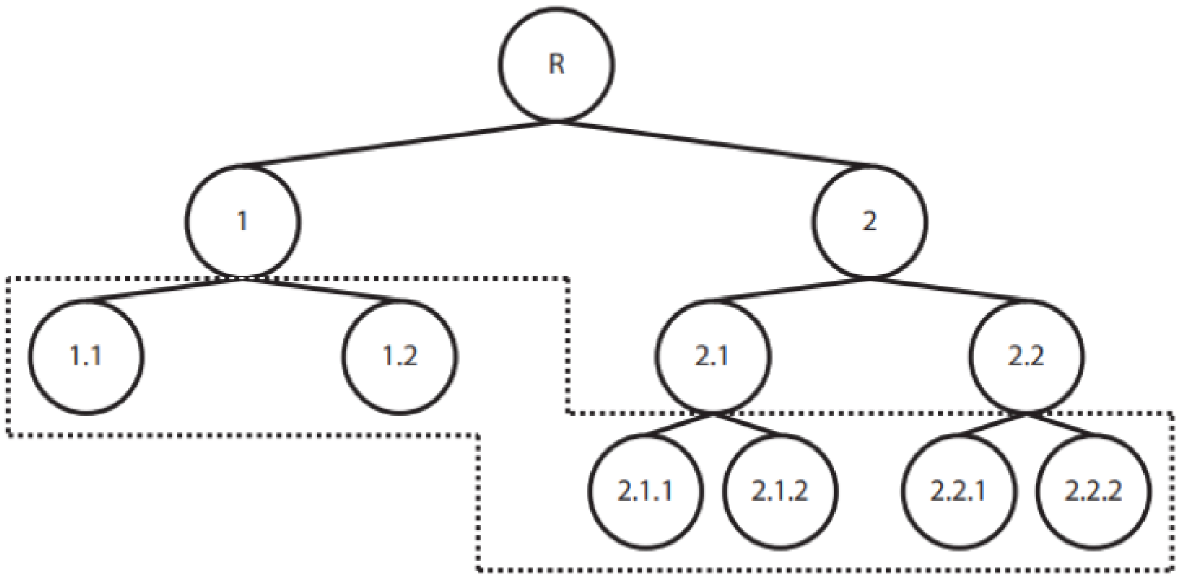


Fig 4: Using a flat multi-class classification algorithm to always predict the leaf nodes

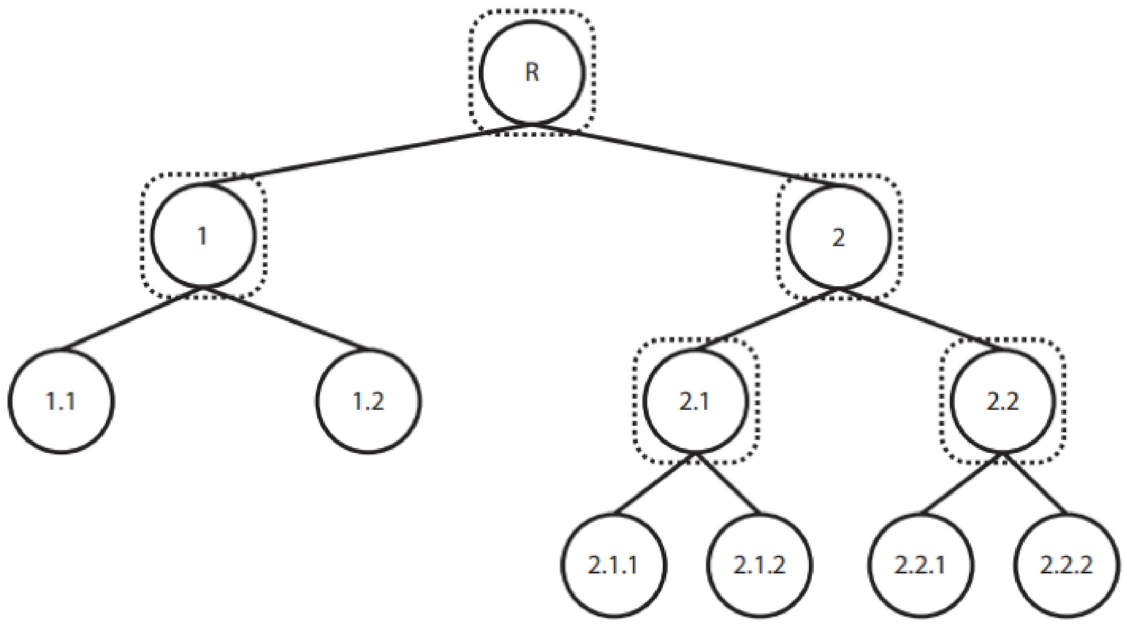


Fig 5: Local Classifier Per Parent Node Approach (circles represent classes and dashed squares with round corners representing multilabel classifiers)

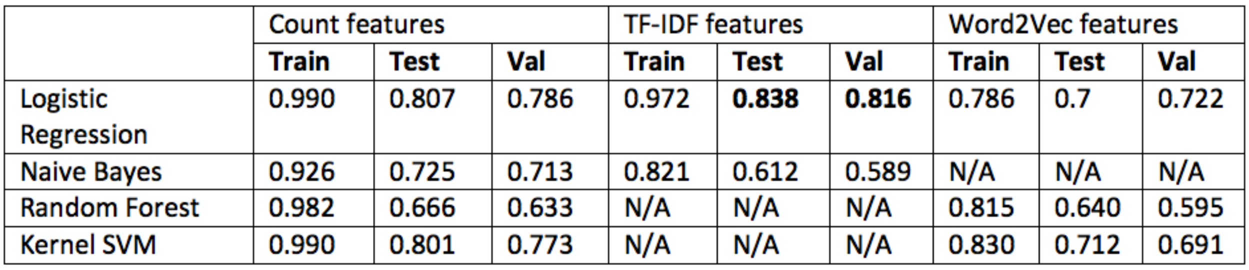
To calculate a final prediction of the pipeline, we combined each level prediction with equal weight and chose the final prediction sub-signal with maximum f1 score overall.

	Count features			TF-IDF features			Word2Vec features		
	Train	**Test**	**Val**	**Train**	**Test**	**Val**	**Train**	**Test**	**Val**
Logistic Regression	0.990	0.807	0.800	0.940	**0.821**	0.794	0.885	0.767	0.749
Naive Bayes	0.912	0.770	0.740	0.712	0.623	0.611	N/A	N/A	N/A
Random Forest	0.987	0.606	0.595	N/A	N/A	N/A	0.873	0.493	0.484
Kernel SVM	0.819	0.739	0.730	0.883	0.826	**0.810**	0.802	0.761	0.744

Table 3. Area f1 score metrics for different word embeddings and classification models

	Count features			TF-IDF features			Word2Vec features		
	Train	**Test**	**Val**	**Train**	**Test**	**Val**	**Train**	**Test**	**Val**
Logistic Regression									
Social	1.0	0.819	0.857	0.987	**0.850**	0.839	0.867	0.817	0.759
Environment	1.0	0.873	0.896	0.995	**0.918**	0.919	0.878	0.830	0.872
Governance	0.998	0.953	0.925	0.992	**0.967**	0.959	0.928	0.907	0.910
Naive Bayes									
Social	0.963	0.817	0.750	0.792	0.671	0.707	N/A	N/A	N/A
Environment	0.979	0.885	0.904	0.970	0.886	0.876	N/A	N/A	N/A
Governance	0.976	0.922	0.926	0.941	0.871	0.849	N/A	N/A	N/A
Random Forest									
Social	0.996	0.785	0.719	N/A	N/A	N/A	0.946	0.747	0.747
Environment	0.990	0.740	0.822	N/A	N/A	N/A	0.942	0.802	0.824
Governance	0.993	0.880	0.849	N/A	N/A	N/A	0.963	0.872	0.850
Kernel SVM									
Social	1.0	0.829	0.856	N/A	N/A	N/A	0.906	0.800	0.755
Environment	1.0	0.861	0.896	N/A	N/A	N/A	0.889	0.823	0.896
Governance	0.998	0.945	0.911	N/A	N/A	N/A	0.943	0.900	0.877

Table 4. Signal F1 score metrics for different word embeddings and classification models



	Count features			TF-IDF features			Word2Vec features		
	Train	**Test**	**Val**	**Train**	**Test**	**Val**	**Train**	**Test**	**Val**
Logistic Regression	0.990	0.807	0.786	0.972	**0.838**	**0.816**	0.786	0.7	0.722
Naive Bayes	0.926	0.725	0.713	0.821	0.612	0.589	N/A	N/A	N/A
Random Forest	0.982	0.666	0.633	N/A	N/A	N/A	0.815	0.640	0.595
Kernel SVM	0.990	0.801	0.773	N/A	N/A	N/A	0.830	0.712	0.691

Table 5. Joined models result by F1 score metric for different models with hierarchical approach

5.4 Productization

After analyzing the metrics for different models, we observed that the metrics differences were not significant. Thus, it was not useful to freeze the best model for our task. To get the best results, we needed to pick the best performing model for each level of the hierarchy automatically on every retraining. This solution allowed us to automate the whole process from regeneration of a new dataset to prediction with the best performing combination. In our final implementation, the best model and corresponding embedding is selected automatically during the training based on the f1 score metric. Additionally, to analyze results for every target label (area, signal, sub-signal), each model includes the capability to provide precision, recall and f1 scoring for each level.

We have built an assembly of models to predict the sub-signal for records using the article's title and body. We used methods both from traditional ML and deep learning. All models were saved as pickle files in the Cloud Storage Service, giving us the ability to load models later and to split training and prediction process if needed. To run the built pipelines in the customer's infrastructure, John Snow Labs converted the Python code used for testing to the production-ready architecture. For this implementation, models were packaged using Docker containers that wrapped Flask [6] REST API services.

The REST API allows us to send new data to the models and receive a prediction as a response. It will allow the ML models to be accessible by 3rd party business applications.

Predictions are made by passing a POST JSON request with title and body to the created Flask web server on port 5000 (by default), which is mapped as external port. The API receives this request and make a prediction, based on the latest models, which are already loaded. It returns the prediction in JSON format:

```
{
    “title”: "Disney whistleblower says company inflated revenue for several years in new SEC filing”,
    “body”: "Filed under: SALT LAKE CITY? Former Walt Disney Company accountant Sandra Kuba filed whistleblower tips with the Securities and Exchange Commission that accused Disney of overstating its revenue for several years, MarketWatch reports. Kuba, who previously worked in the revenue operations department at Disney, said employees had ?systematically overstated revenue by billions of dollars by exploiting weaknesses in the company?s accounting software,? according to MarketWatch. Kuba said she met with SEC officials to discuss the tip. Disney responded, saying the claims were ?utterly without merit.? ?This former employee, who was fired for cause, has persistently made patently false claims for over two years,? the statement said, according to Deadline. ?The claims she made to the company were thoroughly investigated and found to be utterly baseless. It is unfortunate that MarketWatch, which has been aware of the facts for months, knowingly and deliberately chose to give Ms. Kuba?s unfounded claims a platform.? Related The filings explain multiple ways employees would overstate revenue. For example, employees would record fake revenue for ?complimentary golf rounds or for free guest promotions,? according to MarketWatch. A separate filing accusations explained employees recording $500 in revenue from gift cards even though guests paid $395 for those gift guards. ?Kuba has also alleged that employees sometimes recorded revenue twice for gift cards, both when guests bought the gift card and when it was used at a resort. Sometimes, revenue was recorded even though a gift card was given to a guest for free following a customer complaint, for instance, according to the whistleblower?s allegations,? MarketWatch reports. In total, Kuba said the Walt Disney company might have overstated revenue by as much as $6 billion, or 20% higher, in just 2009, according to Business Insider. Kuba had been fired by Disney in the past a month after she reported revenue issues to management in 2013, presented them to senior executives in 2016 and then the SEC in 2017. Disney said the company fired Kuba because ?she displayed a pattern of workplace complaints against coworkers without a reasonable basis for doing so,? according to Business Insider. Start your day with the top stories you missed while you were sleeping.” }
```

ESG Model Output:

```
{
    "Governance.DataProtection.DataBreaches": 0.972,
    "Governance.DataProtection.Cybersecurity": 0.006,
    "Governance.DataProtection.DataPrivacy": 0.005,
    "Social.HumanRights.SupplyChain": 0.002
}
```

Additionally, we created different endpoints allowing the customer to receive the full information about currently uploaded models:

1. /train endpoint allows a user to retrain models with fine tuning or with the previously found best parameters
2. /metrics endpoint returns the latest model metrics for training, testing and validation subsets, allowing a parameter to be passed for bulk validation of external datasets
3. /modelClassMetrics endpoint returns detailed metrics for each sub-signal
4. /classCount returns statistics for an unstructured record dataset, with sub-signals and quantity

This implementation allows the customer's analytics team to control model performance and have easy access to the model quality metrics.
Another underrated challenge in machine learning development is the deployment of the trained models in production in a scalable way. We solved this challenge by using Docker, which allowed us to use a lot of services that work in an isolated manner and serve as a data provider to a web application.
Docker is a containerization platform which packages an application & all its dependencies. It will allow us to deploy the application on a remote server and evaluate it locally without the need to install any libraries, dependencies, or Graphical interface (IUPython, Jupyter Notebook and etc). Based on the load and performance requirements of the prediction or training processes, instances can be spun up & down on demand based on the rules that are set up.

6. Conclusion

Our algorithm with optimal model selection and joined hierarchy models achieves 84.6% F1 score metric on the text dataset and 82.3% on the validation dataset for predicting the signal with highest probability.

Joined model	**Precision**	**Recall**	**F1 Score**
train	0.987	0.992	0.989
test	0.848	0.849	**0.846**
validation	0.834	0.829	**0.823**

Table 6. Best models result with hierarchical approach

We examined the prediction errors made by our algorithm and found that the model often confuses the predicted label with the Negative label from the dataset. In general, metrics dropped significantly after adding the Negative label to the model. It may be related to the low quality of record content or lack of manual review. The process of picking up Negative labels was automated and keyword search was used to label this category.

We also found a low performing sub-signal - “Social. CommunityResponsibility. AccessAffordability”. To understand the cause of the model’s misinterpretation of the content we extracted best performing features – TF-IDF of the selected words - and applied a dimension reduction method (t-SNE [7]) to visualize the word vectors in 2-D space. Fig. 7 shows the result of this procedure. If we review the distribution of the target sub-signal we notice that it intersect with other clusters:

- "Social.HumanRights.Discrimination" (light blue cluster in the top)
- "Social.EmployeeStandards.LaborManagementCompensationBenefits Development" (lime in the center)
- "Social.HumanRights.SupplyChain " (purple in the center top).

Due to the lack of data for this sub-signal and non-specific record content, this category is not showing significant quality in classification and models have a tendency to mislabel it with other sub-signals.

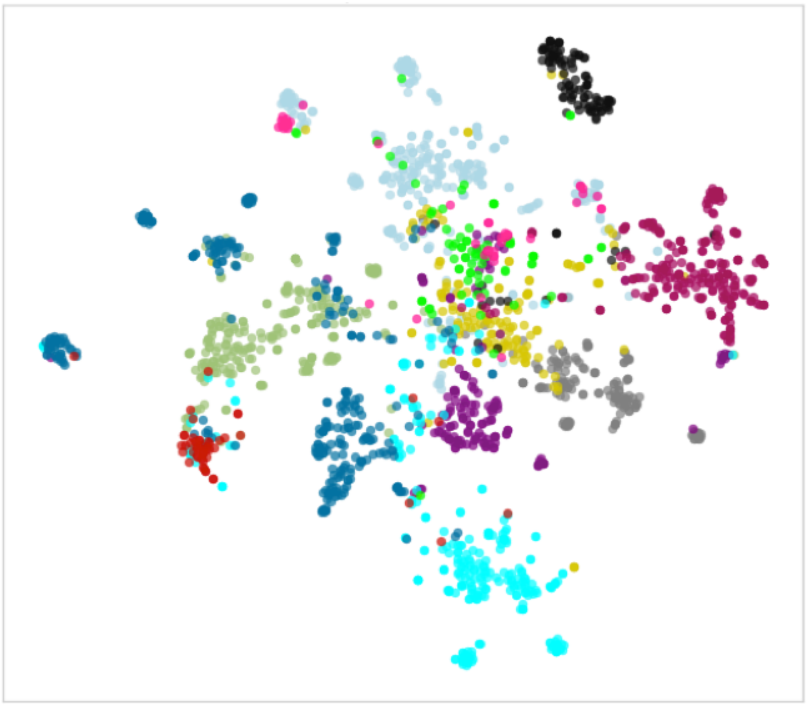


Fig 6: Visualization of word embeddings for different sub-signals

In the future, we must consider labeling more articles for this sub-signal and balance it in the dataset to improve overall metrics and model prediction power.

During the project it became clear that ESG sub-signal classification is a challenging task for even complex models. We attribute this to the subjectivity in sub-signal assignment during the labeling process.

John Snow Labs provided its customer excellent results by using advanced machine learning and NLP techniques. This was combined with a proven end-to-end delivery process – from annotations through data science to a large-scale production deployment.

7. References

[1] Q. Le, T. Mikolov. Distributed Representations of Sentences and Documents. CoRR abs/1405.4053, 2014

[2] J. Pennington, R. Socher, C. D. Manning. Glove: Global Vectors for Word Representation. EMNLP, 14:1532–1543, 2014.

[3] C.M. Bishop. Pattern Recognition and Machine Learning. Chapter 4.3.4

[4] K. Crammer, Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research, vol. 2, pp. 265-292, 2001

[5] C. Silla, A. Freitas. A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery, 22 (1-2), 31-72, 2011

[6] http://flask.palletsprojects.com/en/1.1.x/

[7] L. Maaten, G. Hinton. Visualizing Data using t-SNE. Journal of Machine Learning Research. 9: 2579–2605, 2008

[8] https://github.com/JohnSnowLabs/spark-nlp-models

About John Snow Labs

John Snow Labs Inc. is an award-winning healthcare AI & NLP company, accelerating progress in data science with state-of-the-art platforms, models and data. The company is the winner of the 2018 AI Solution Provider of the Year award, 2019 AI Platform of the Year Award, and 2019 International Data Science Technology Award, among others.

John Snow Labs is the team behind Spark NLP – the world's most widely used NLP library in the enterprise. A third of the team have a PhD or MD degree and 75% of team members have at least a Master's, coming from multiple disciplines covering data science, medicine, data engineering, pharma, security, and DataOps. A Delaware Corporation, John Snow Labs runs as a global virtual team located in 20 countries around the globe.

Other Spark NLP Case Studies

Click here to read technical summaries of real-world Spark NLP case studies:

UiPath: High accuracy fact extraction from long financial documents
Answering questions accurately based on information from financial documents, which can be a hundred or more pages long, is a challenge even for human domain experts. While traditional rule-based or expression-matching techniques work for simple fields in templated documents, it is harder to infer facts based on implied statements, on the absence of certain statements, or on the combination of other facts. Answering such questions at a very high level of accuracy requires state-of-the-art deep learning techniques applied to NLP.
Spark NLP was used to augment the UiPath smart data extraction platform in order to automatically infer fuzzy, implied, and complex facts from long financial documents. This case study covers the technical challenges, the architecture of the full solution, and lessons learned that you can directly apply to your next data extraction project.

Roche: Accurate information extraction from pathology & radiology reports
Many critical facts required by healthcare AI applications like patient risk prediction, cohort selection and clinical decision support are locked in unstructured free-text data. Recent advances in deep learning have raised the bar on achievable accuracy for tasks like biomedical named entity recognition, assertion status detection, entity resolution, de-identification and others. This case study presents the first industrial-grade implementation of these new results and its application at scale.
Roche is the world's #1 company for in-vitro diagnostics and its medicines are used to treat over 130 million people each year. It's building a clinical decision support product portfolio, starting with oncology. Roche is using Spark NLP for Healthcare to extract clinical facts from pathology and radiology reports. The case study covers the design of the deep learning pipelines used to simplify training, optimization, and inference of such domain-specific models at scale.

Deep6: Matching patients to clinical trials
Recruiting patients for clinical trials is a major challenge in drug development. Finding patients requires an in-depth understanding of their medical histories and current health statuses while the majority of patient data is unstructured and spread across physician notes, pathology, imaging, genomic, and other reports. For this reason, clinical trial recruitment is a slow and manual process. This case study describes how Deep 6 uses the Spark natural language processing (NLP) platform to apply state-of-the-art deep learning to accurately extract the relevant clinical facts from unstructured text. These facts are then used in subsequent data science pipelines in constructing patients' medical histories.

SelectData: Interpreting millions of stories with deep-learned OCR and NLP
Many businesses still depend on documents stored as images—from receipts, manifests, invoices, medical reports, and ID cards snapped with mobile phone cameras to contracts,waivers, leases, forms, and audit records digitized with scanners. Extracting high-quality data from these images comes with three challenges. First is OCR, as in dealing with crumpled receipts photographed from an angle in a dimly lit room. Second is NLP, extracting normalized values and entities from the natural language text. The third is building predictors or recommendations that suggest the best next action—and in particular can deal with missing, wrong, or conflicting information generated by the previous steps.
This case study illustrates an AI system that reads millions of pages of patient information, gathered from hundreds of sources, resulting in a great variety of image formats, templates, and quality. It explores the solution architecture and key lessons learned in going from raw images to a deployed predictive workflow based on facts extracted from the scanned documents.