

REMI_MOUZAYEK_01601868

by Remi Mouzayek

Submission date: 09-Sep-2019 12:50PM (UTC+0100)

Submission ID: 110604435

File name: REMI_MOUZAYEK_01601868.pdf (2.37M)

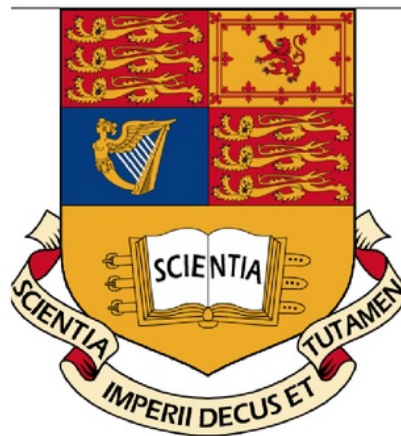
Word count: 17912

Character count: 87061

**Extraction of relation-objects from documents
using a weakly supervised setting**

by

Remi Mouzayek (CID: 01601868)



Department of Mathematics
Imperial College London
London SW7 2AZ
United Kingdom

Thesis submitted as part of the requirements for the award of the
MSc in Mathematics and Finance, Imperial College London, 2018-2019

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Signature and date:

A handwritten signature in black ink, appearing to be 'B. A.', written over a horizontal line.

02/09/2019

This report contains material confidential and proprietary to
Bloomberg L.P. and is provided under the terms of a
Non-Disclosure Agreement (Agreement #3009783) with
Imperial College.

Acknowledgements

First of all, I would like to express my gratitude to my project supervisor Dr. Thomas Cass for putting a great effort into providing guidance whenever needed.

I am very grateful to Bloomberg L.P. and especially to my manager Taryn Wilkins, and to my colleagues Fran Silavong and Carl Marcelus without whom the realization of this thesis would not have been possible. I would like also to thank Guillem Forto Cornella, Ashwin Laxmikant Bhakre, Paritosh Mital, Norvan Sahiner, Yue Xiao, Frederic Wantiez and Aditya Chaturvedi.

Last but not least, I would like to thank my family for their invaluable support.

Contents

0	Introduction	8
0.1	Motivation	8
0.2	Applications	9
0.3	Work done in this thesis	9
1	Part I: the weak supervision setting	11
1.1	How to deal with a small training data set?	11
1.2	Definition of the weak supervision	14
1.3	Overview of the framework	15
1.4	Generative model with a small annotated data set	16
1.4.1	Conditionally independent voters	17
1.4.2	Modeling accuracy	18
1.4.3	Modeling dependencies	21
1.5	Generative model without annotated data	26
1.6	Discriminative model	28
2	Part II: dividend extractions	32
2.1	Problem setup	32
2.2	Requirement specifications	35
2.2.1	Time	35
2.2.2	Performance metrics	35
2.3	The difficulties	37
2.4	Description of the data pipeline	39
2.4.1	Cleaning and patternization of the news	40
2.4.2	Extraction of the candidates	42
2.4.3	Building of the labeling matrix	43
2.4.4	The trick of the augmented candidate	45
2.4.5	Extraction of the candidates for the dates	49
2.4.6	Generative model	51

2.4.7	Calibration stage	51
2.4.8	Post-processing rules	54
2.4.9	Multiple dividends event	54
2.5	Details on the infrastructure	58
2.5.1	Localisation and handlers	58
2.5.2	Implementations of the heuristic labeling functions	58
2.5.3	Generative model	61
2.5.4	Time optimisation	61
2.6	Experimental results	62
2.6.1	Results at document level	62
2.6.2	Results at label level	64
2.6.3	Review of the performances	65
2.6.4	Assessment of the results	67
	Conclusion	69

0 Introduction

0.1 Motivation

The abilities of statistical models to learn from data has achieved great performances in lots of domains, such as computer vision or natural language processing. Historically, there are two major difficulties in obtaining a well-functioning end-to-end model: having a good quality and big enough training data set, and also performing feature engineering, which consists in extracting relevant features from the data, in order to base the predictions or classifications directly on it. The system does not learn directly from the data, but rather from the features. Feature engineering has become less fundamental after the development of recent state-of-the-art deep learning algorithms, which use learned representations of the data. One example of this is the representation of a word as a high dimensional vector which encodes its meaning.

Thus, algorithms such as the recent BERT [1] of Google AI research have reached very high performances, but are really data consuming. Building huge and very accurate training data sets for these algorithms is particularly expensive and takes a lot of time, because everything is often based on the laborious work of a large number of people, who manually annotate hundred of thousands of texts, pictures or any other training data point structures.

Weak supervision is a branch of machine learning which addresses this problem. For a given task, several sources are combined, which can be very different, such as a so-called weak classifier. It receives this name because it is generally only trained within the usual supervision setting on a small data set, or a lower-quality source, based, for example, on crowd-sourcing [27]. High-level heuristic rules implemented by subject-matter experts can also be used, or even any kind of external knowledge from a database using distant supervision [25, 26]. In a few words, weak supervision is an ensemble learning method which can combine sources of different nature with limited coverage and accuracy to obtain an end-to-end model with tremendous performances.

0.2 Applications

The weak supervision is a recent topic, since it is the completion of the work of Stanford researchers in 2017. They made Snorkel [18] available, a free open-source system which allows anyone to apply the weak supervision setting for their specific user-case.

To understand to what extent weak supervision could have an impact in a very large number of different sectors, and in the academic world, let's first consider the current benefits of machine learning within the traditional fully supervised setting: information extraction, spam detection, handwriting recognition, analyzing the health condition of an individual, stock price prediction, detection of financial frauds, virtual personal assistant, visual surveillance... All of these tasks rely on a large and high quality training data set to give enough examples to a machine learning model, which is going to learn how to generalize beyond these examples. For all these tasks, there is not a lot of free data sets available, because it is expensive, or because the data are indeed private. Hence, the weak supervision setting provides a way to obtain results much faster, cheaper, and at the same time with very high performances, comparable to the fully supervised usual setting.

For the financial industry alone, we can imagine lots of applications. For example, credit scoring [28], which is the process used by companies and institutions to assess the creditworthiness of customers in order to decide on whether to extend or deny credit, could really profit from a mix of the usual machine learning models, a couple of heuristics rules written by credit experts, and a database containing any relevant information.

0.3 Work done in this thesis

This report is separated into two parts. The first part is focused on the theory of the weak supervision setting. Most of the articles are quite recent, as they have been published over the past two years. I propose an overview of the different approaches, and describe some aspects of the underlying structure of the probabilistic model, which relies on graph theory.

Then, in the second part, I focus on my personal user-case: the extraction of dividend from news. The data pipeline implemented (in Python) is detailed, and I highlight the

main findings that helped me reach a high level of performance: the consideration of an “augmented” candidate, a calibration stage, and a Bayesian generator of heuristics. Some particularities are really specific to my user-case, but I do believe that the mindset could be reused for other tasks.

1 Part I: the weak supervision setting

1.1 How to deal with a small training data set?

Deep learning under the usual supervision learning setting has reached very high performances for a lot of different tasks. Under this setting, the machine learning model learns a map which sends an input, an email for example, to an output, which can be the binary answer of the question: “Is this email a spam?”. Then, each item of the training data set is a pair input/output. Using these pairs, the model will learn how to generalize beyond these examples.

Unfortunately, it is a known fact that any deep learning algorithm requires a large set of annotated training data. These data sets, which have to be very accurate, are not easy to find, even for the biggest companies. They need a lot of people to annotate manually hundred of thousands of training pairs. It takes time and a lot of money, especially when we expect to have a perfect training data set, without any wrong annotation.

There are a couple of methods that can be used to get around this problem [2]. First of all, if we have a good understanding of the strengths and weaknesses of our algorithm, obtained, for instance, from a clustering study of the data, we can select a couple of documents that we want to annotate because we consider that these are currently the hardest for our model. The trade-off cost/increase of performances is optimal for these items. If the task is to find out if there is a dog in a picture, one could thus choose to annotate a data set for which he knows that there are pictures of dogs and cats, and hope that the model will learn how to make the difference between these two species, because it may be one of the weakness of the model. We call this approach **active learning**.

There is also a broad set of methods, called **semi-supervised learning**, which get around the problem of the cost of annotated training data set by making use of both annotated and non-annotated data. Without going into details, there is a classic illustration that we can find in almost every educational papers presenting the basics of this setting that I reproduced below:

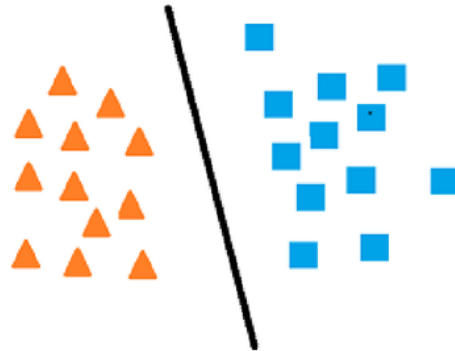


Figure 1: Boundary of a classifier in a fully supervised setting

Machine learning is all about boundaries. The example above (Figure 1) is a very simple classification problem. There are just 2 classes, represented by the squares and the triangles. We can make use of the annotated figures to find a first boundary, but it is easy to notice that, by making use of all the data, and particularly of the clustering structure, we can find a more accurate boundary for the classification.

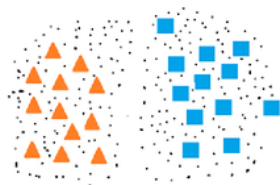


Figure 2: Assumption: clustering structure

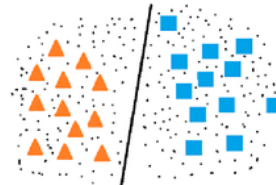


Figure 3: Boundary of an improved classifier in a semi-supervised setting

As we can already notice in this example, this approach requires a very specific structure of the data that we are working on. Like it is said in [3], one of the fundamental hypothesis is the clustering structure of the data and, as a consequence, that all the data in the same cluster belongs to the same class (see Figure 2). It makes sense because, in other words, to be able to combine the information of both annotated and non-annotated data,

the behavior that we hope is that an input data point “close” (for the relevant topology of the input) to another one is going to have an output “close” to the second one (for the relevant topology of the output).

Another option called **transfer learning** is also possible. This approach consists into pre-training a model on another task similar to ours, but for which we have a lot of annotated data, and then to retrain the model for our task using the small number of available data.

Then, while active learning still requires annotating a large number of documents, the semi-supervised learning necessitates some particular assumptions for the topology of the data, and transfer learning assumes that you already have another model on a similar task with enough training data. In addition, none of these approaches are applicable only if we have noisy labeled data, that is to say, if some of the training pairs are wrong.

The weak supervision addresses the problem of not having a large and highly accurate training data set, without requiring any of the conditions detailed above.

1.2 Definition of the weak supervision

The spearhead of the weak supervision is to be applied in cases where there is not a big training data set that could allow us to train a state-of-the-art deep learning algorithm, under the usual supervised learning setting.

Weak supervision is indeed a flexible ensemble learning method combining different models that we call labeling functions, and which can be machine learning based, or simply static, incorporating directly external knowledge. A labeling function is then a function, which takes an input, for example a picture, and returns a label such as the binary answer of the question: “Is there any dog in this picture?”.

A labeling function can then be a machine leaning algorithm with limited accuracy, because it was trained on a small data set, or trained for a slightly different task in the transfer learning mindset for instance, like stated beforehand. In the case of the external knowledge, it can come from a database, such as for the distant supervision approach. For example, if we want to extract from documents relations building/location, we can use a database with the names of lots of famous buildings associated with their geographic location, and then extract the relevant sentences from documents. This external knowledge can also be directly implemented by human users, either by annotating manually training data such as in crowd-sourcing or, in a more efficient way, by implementing heuristic rules. The labeling functions can also be applied to only certain data points and abstains for other ones. It can also be based in different components of the data point, such as in co-training. For example a classifier of rogue websites which combines a model based on the URL, and another one using the words in the home page.

Without being in the details yet, it’s important to be aware of the flexibility of the different labeling functions that can be used in this approach. Everything that you think is relevant to annotate some documents could be a labeling function. These statements raise of course a lot of questions. How does it work? What is the minimum accuracy of a potential labeling function to be beneficial of the ensemble learning model? Can the whole model be mislead if a lot of functions are wrong for a given example, or in other words how to deal with heuristic labeling functions that are somehow correlated?

1.3 Overview of the framework

The weak supervision relies on the notion of labeling function. A rough definition of these functions could be: a small entity which does a part of the job based on an external knowledge, a machine learning model, or even any kind of logic. These labeling functions can then be very different. When they are based on an external knowledge, we will refer to them as heuristic labeling functions. These are the most widely used within the weak supervision setting, because as we will see later, their implementations are fast, and do not require a large training data set.

Like said previously, the heuristic labeling functions can be of different type. They aim at incorporating an external knowledge, but this knowledge can be of different nature. An heuristic labeling function to build a model of spam detection can then:

- makes use of external knowledge, stored in existing database but which does not solve directly the problem (distant supervision) as a database of example of spams, or a list of rogue email addresses.
- be a simple rules of thumbs: "if there is not any object, this email is a spam"
- relies on crowd-sourcing: cheap annotations done by a lot of non-experts annotators, who have simply classified emails, for example.

These heuristic labeling functions can also "cheat" in the sense that they use information not available. For example, in order to identify fraudulent transactions, PayPal has recently (2018) implemented a weak supervision model [21], and some of the heuristic labeling functions are based on the report of the clients, who declare a potential fraud. This information is of course not available at the time of the transaction.

I consider this example as a suitable entry point to understand the core of the weak supervision mindset. The goal is not yet to have an end-to-end model, but rather a efficient annotator called **generative model**. Using this generative model, we can build rapidly, and with almost no cost a large training data set of annotated examples.

We then train a usual deep learning algorithm on this large data set, **called discriminative model**. Some of the annotations from the generative model may be wrong, that is why we call them weak labels, and we can also see them as noisy. However, it is not an issue because the deep learning model will be "noise-aware". For a given input, traditionally called **candidate** within the weak supervision setting, it is made possible by the fact that the generative model outputs a probabilistic label. A generative model for the spam classification mentioned earlier would not simply give a binary output, but the likelihood that a candidate (an email in this case) is a spam, or not.

Then, it is completely fine that some labeling functions "cheat" by using information not available in a real-life situation, but it is not the end of the story. The performances of the discriminative model may actually out pass the ones of the generative model. Indeed, the generative model relies on the labeling functions which in most of the cases are based on rules of thumbs implemented by humans, and using a basic reasoning. In some situation, a deep learning algorithms may find more complex dependencies that these heuristic labeling functions. It may also improve the power of generalization of the model, which will now longer depend on a small set of rules, and necessary be very correlated to the examples used by the authors of these functions.

However, as it is often the case in machine learning, there is no general solution and there are cases where the generative model is better.

In order to present clear expressions, We now focus on a binary classification problem, even if it can be generalized with the same reasoning to the multi-classes situation.

1.4 Generative model with a small annotated data set

In practice, there is usually a small labeled training data set which is available or can be annotate quite quickly, but is way too small to train a state-of-the-art machine learning model. We now make the assumption that we have such data set. Starting from this database, and a couple of labeling functions, which encode knowledge of different nature, we want to build a generative model with the highest possible performances.

In term of notations, let's consider that we have n data points $(x_i)_{i \in \{1..n\}}$ and p labeling functions $(\lambda_j)_{j \in \{1..p\}}$. We can attach each x_i with its true label y_i , and then (x_i, y_i) is a typical training example that we can find in the usual fully supervised setting. According to our initial assumption of a binary classification, we can note $\forall i \in \{1..n\} y_i \in \{-1, 1\}$.

For a given data point x_i the output of the labeling function j is $\lambda_{i,j}$. A labeling function can either opt for one of the two classes, or abstain so $\lambda_{i,j} \in \{-1, 0, 1\}$, with 0 encoding the abstention.

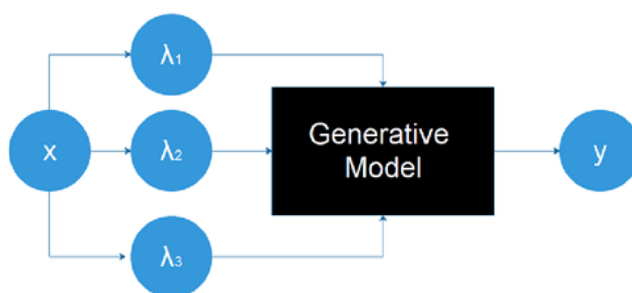


Figure 4: Sketch of the functioning of a generative model

For a new data point x_{i_0} , for which we do not know the true label, we consider, the unknown, y_{i_0} as a latent variable like proposed in the so called, by its authors, data programming approach [4]. The goal is then to rely on $(\lambda_{i_0,j})_{j \in \{1..p\}}$, computed using the known variable x_{i_0} , in order to statistically infer the value of y_{i_0} .

1.4.1 Conditionally independent voters

The most naive generative model is to consider each labeling function as an independent voter, conditionally on the results of the other labeling functions (it is clear that the strict independence is a strong assumption which does not make sense here). For a given data point, like a democratic election, the majority wins. It makes sense that this approach is very simplistic. The first reason which comes to mind is that there are probably some labeling functions which are more accurate than others. We do not want the final output of our generative model to be too much influenced by a bigger numbers of less accurate

labeling functions.

1.4.2 Modeling accuracy

Then, we can improve the previous generative model by taking into account the accuracy of the labeling functions. Like it is mentioned in [8] we can then model easily the probability distribution using a conditioning, and the assumption of conditional independence of the labeling functions:

$$\begin{aligned} p(y, \lambda_1, \dots, \lambda_p) &= p(y)p(\lambda_1, \dots, \lambda_p|y) \\ &= p(y) \prod_{i=1}^p p(\lambda_i|y) \end{aligned} \quad (1.1)$$

Then, by separating the cases, one can get:

$$p(y, \lambda_1, \dots, \lambda_p) = p(y) \prod_{i=1}^p \beta_i \alpha_i \mathbb{1}_{\lambda_i=y} + \beta_i (1 - \alpha_i) \mathbb{1}_{\lambda_i=-y} + (1 - \beta_i) \mathbb{1}_{\lambda_i=0} \quad (1.2)$$

with β_i the probability that a labeling function λ_i does not abstain, and annotates the item. α_i is the probability that a pronouncing labeling function is correct. One can notice that this approach is similar to a naive Bayes classification.

We have then a family of probability distributions which depends on both α and β , with $\alpha = (\alpha_1, \dots, \alpha_p)$ and $\beta = (\beta_1, \dots, \beta_p)$. In order to choose the most relevant parameters, we can consider each $(y_i, \lambda_{i,1}, \dots, \lambda_{i,p})$ as a sample observation and we want to maximise the likelihood to observe these samples. As a consequence, it is relevant to opt for a maximum likelihood estimation. Any efficient optimization method ([8] rightly advises a stochastic gradient descent) will then be able to find the optimal parameters $\hat{\alpha}$ and $\hat{\beta}$.

The question is now how we could use these optimal parameters in order to take into account the accuracy of the labeling functions. By reversing the conditioning, using the optimal parameters and the outputs of the labeling functions, we can deduce the most likely value for y_{i_0} .

At this stage, there is a very interesting result which is proved in [8]. I won't go to much into details, because the architecture detailed in this paper is actually different of the one I would like to present. Their generative model (which, for their user-case, also takes into account the accuracy of the different labeling functions) is used to generate a weak label y . A discriminative model is then trained to map a features representation $f(x)$ of the input x to the probabilistic weak label y , using a logistic regression, and, without knowing the true label of the training data points. For this architecture, and under assumptions that we can summarize as: the model is correct, the independence of the labeling functions is verified, and they provide sufficient information, with enough accuracy we can hope to estimate the right y , the authors proved that for a given level of error ϵ , it suffices to have only $p = O(1)$ labeling functions, and $O(\epsilon^{-2} \log \epsilon)$ non-annotated training examples in order to reach the same asymptotic performance level than in a fully supervised learning setting. It is a strong result, and it proves what our intuition could suggest. Namely, that the amount of work to apply weak supervision, and to only write $O(1)$ labeling functions, is way inferior to the traditional work which consists of annotating manually $O(\epsilon^{-2} \log \epsilon)$ training examples.

Let's now take a step back. We have just seen a generative model which weighs the labeling functions in order to favour the most accurate ones. It can easily improve the performances compared to a simple "democratic" majority vote. This model is based on the probability to observe y under the condition: λ_1 is the output of the labeling function 1, ..., λ_p is the output of the labeling function p . This consideration can push us to see this problem as a Bayesian network. A Bayesian network is a directed acyclic graph, for which each vertex is a variable and each edge corresponds to a conditional dependency. It is indeed a kind of graphical model used to model a broad set of phenomena influenced by conditional dependencies.

It is a very simple graphical model, but let's just keep in mind for now that a network looks like a good representation for a generative model.

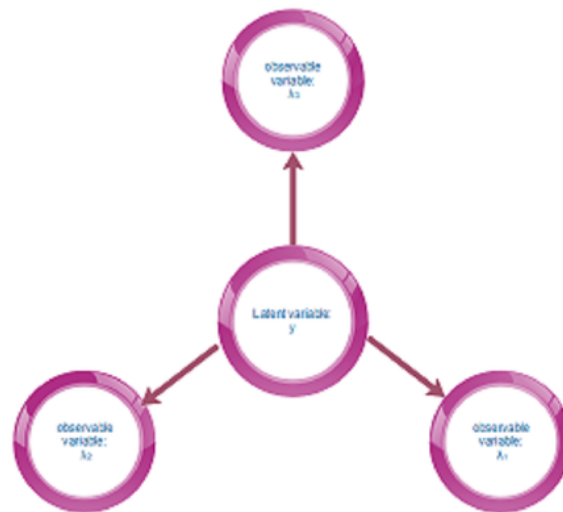


Figure 5: Bayesian Network of the generative model taking into account the accuracy of the labeling functions

The model we have studied so far is based on the assumption that all the labeling functions are conditionally independent. This assumption is very strong and, in practice, this hypothesis does not look realistic. When users specify an heuristic labeling function, they try to detect a signal which increases the likelihood that the target y takes a specific value. It is common that these labeling functions present statistical dependencies, because one signal could be linked in a way (to be defined) to other signals. Let's say that you want to build a model which try to predict if a newborn is likely to become obese in later life. One simple signal could be to write that if the weight of the newborn out passes a given threshold, he is threaten by obesity, and an other could be based on his hip circumference. Of course these two signals are correlated. In a way, we do not want our generative model to give a kind of double weight to two labeling functions which are accurate, but highly correlated (the efficiency of the signal being already provided by one of these labeling functions). The most trivial example being two users who write the same heuristic labeling function.

1.4.3 Modeling dependencies

A more efficient generative model has then to learn the dependency structures of the labeling functions. It is unthinkable to assign this task to the different subject matter experts, who encode their knowledge by writing heuristic labeling functions, because, first, there is a lot of work to do: this task would not scale with a too important number of these functions. In addition, the correlation depends on the data set we are working on, and it is not obvious that it could be generalize to other data points. Thus, if at some point, new data are added everything would have to be done over again. Moreover, some dependencies can be more subtle than others, and during the development stage, it can happen that someone adjusts an heuristic labeling function. It could change quickly its dependencies. For all of these reasons, it is needed to have a model which is able to learn by itself the dependency structures.

In order to model such dependencies, we could opt for a set of all the pairwise correlations of the labeling functions. However, this approach is very greedy. For p labeling functions, there are $\binom{p}{2}$ values to compute, and then the complexity is quadratic with respect to the number of labeling functions. Then, one of the capabilities of the generative model has to select the potential dependency relations and only compute these ones.

Similarly to what is suggested in most of the scientific papers dealing with weak supervision [2, 4, 5, 6, 8], we are going to opt for a graphical model. We then model the joint distribution of $\lambda_1, \dots, \lambda_p, y$ via a Markov random field, with associated graph $G = (V, E)$. V is the set of vertices of the graph. It is composed of all the outputs of all the labeling functions $\lambda_1, \dots, \lambda_p$ and the true label y . Two vertices are not independent conditioned on the other vertices if it exists an edge connecting them to each other. We can obviously assume that all the outputs of the labeling functions are not conditionally independent of the true label y , then $\forall i \in \{1..p\} (\lambda_i, y) \in E$. Similarly, if two labeling functions are correlated in the graph, then their respective vertex are connected by an edge.

One can notice that by this representation of the problem, as an undirected graph connecting the different weak sources, we have made the assumption that we overlook more complex correlations than the pairwise ones. This assumption is made in most of

the weak supervision related papers. In [5] the authors state that pairwise correlations are the most common, and that handling higher order dependencies is straightforward. One could also argue that, at some point, introducing harder dependency structures could push us to rethink the model as a hyper-graph instead.

I am now going to deviate, and to introduce the most useful notions of graph theory which are needed in order to understand the principles of the approach. Nevertheless, readers who wish to learn more about these graphical model can read [8] or [9], which are very clear, and that I used to write my own definitions.

The central object of the theory that I want to present is the **Markov random field**, or undirected graphical model. It is a probability distribution respecting the structure of a fixed graph. First of all an **undirected graph** $G = (V, E)$ is composed of two sets. V is the set of the vertices of the graph, and E is the set of the edges. The graph is undirected, which means that all the edges are unordered vertex pairs: $(s, t) \in E \iff (t, s) \in E$.

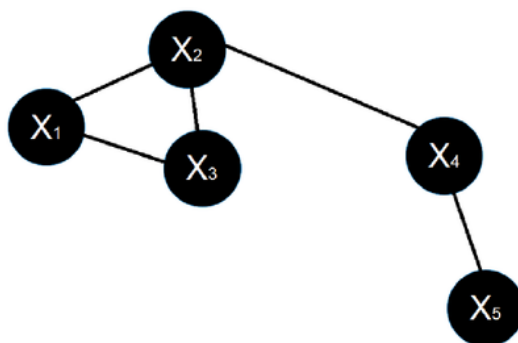


Figure 6: Example of undirected graph

We can decompose an undirected graph into meaningful entities called **cliques**. Formally, we can see these subsets of vertices as fully connected area: $C \subseteq V$ is a clique of G , if for all distincts $s, t \in C$, $(s, t) \in E$.

It can be noticed that it is often possible to find a couple of "sub-cliques" in a clique. We then introduce the notion of **clique maximal**, which are not contained by any other clique.

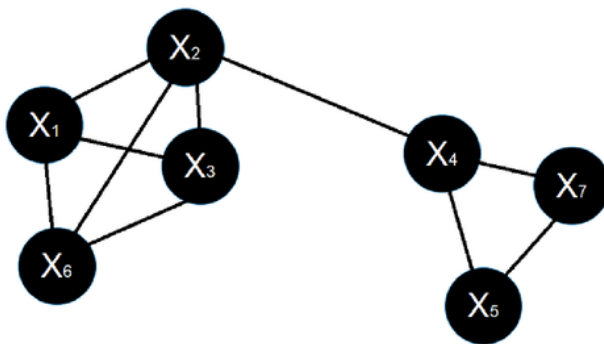


Figure 7: $\{X_1, X_2, X_3, X_6\}$, $\{X_1, X_2, X_3\}$, $\{X_4, X_5, X_7\}$ are cliques, but $\{X_1, X_2, X_3\}$ is not a clique maximal

We also define the **neighborhood** of any given vertex as: $N(s) := \{t \in V; (s, t) \in E\}$. All the vertices of the graph can be associated with a variable, observable or not. Let's call X_1, \dots, X_n the random variables associated to the n vertices of a given graph. For clarity, I have decided to do not make any differences between a vertex and the variable that it represents, then X_1 can either refers to the first vertex of the random variable associated, according to the context.

Let's now consider an undirected graph $G = (V, E)$. V represents a set of vertices $X = (X_v)_{v \in V}$. We note p the distribution followed by X , and χ the space of the different states (or configurations) which can be taken by X . We can then interpret $P(X = x)$ for $x \in \chi$ as the probability to find the vertices of the graph in the state x :

$$P(X = x) = P(\{X_1 = x_1, \dots, X_n = x_n\}) \quad (1.3)$$

We say that the distribution p is positive when $\forall x \in \mathcal{X}$, $p(X = x) > 0$. This set of random variables $X = (X_v)_{v \in V}$ is a Markov random field with respect to G , if it satisfies the **local Markov properties** [10, 11]:

- **Pairwise Markov property:** Any two variables which are not connected by any edge are conditionally independent given all other variables: $X_u \perp X_v \mid X_{V \setminus \{u, v\}}$
- **Local Markov property:** A variable is conditionally independent of all other variables given its neighbourhood: $X_v \perp X_{V \setminus v \cup N(v)} \mid X_{N(v)}$
- **Global Markov property:** Any two subsets of variables are conditionally independent given a separating subset: $X_A \perp X_B \mid X_S$ where every path from a node in A to a node in B passes through S .

The main result of the Random Markov Fields is the Hammersley-Clifford theorem.

Theorem 1.1. (Hammersley-Clifford theorem)

If $X = (X_v)_{v \in V}$ is a Markov Random Field which follow a positive distribution p , then we can write: $p(X = x) = \frac{1}{Z} \prod_{C \in Cl(G)} \phi_C(x_C)$

where Z is a normalization constant called partition, $Cl(G)$ is the sets of the cliques of the graph which can either be chosen only maximal or not, and ϕ_C is called a clique compatibility function. It maps a configuration of the clique C : $x_C = \{x_s, s \in V\}$ to a positive real.

The Hammersley-Clifford theorem provides a factorization form of a Random Markov field under the non-restrictive hypothesis of considering a positive distribution. In term of interpretability of the clique compatibility function, $\log \phi_C(x_C)$ can be viewed as the energy of the configuration x_C . It makes sense then to rewrite the joint distribution as an exponential family: $p(X = x) = \frac{1}{Z} \exp \sum_{C \in Cl(G)} \log \phi_C(x_C)$.

This expression can actually be rewrite [12]:

If $(\psi_C(x_C))_{C \in Cl(G)}$ is a set of clique-wise sufficient statistics, then we have: $p(X = x) = \frac{1}{Z} \exp \sum_{C \in Cl(G)} \theta_C \psi_C(x_C)$ for $(\theta_C)_{C \in Cl(G)}$ a set of natural parameters.

Following the reasoning of [4, 5], we now introduce three factor types.

- the propensity: $\phi_{i,j}^{Lab}(\lambda_1, \dots, \lambda_p, y) = \mathbb{1}_{\lambda_{i,j} \neq 0}$ which is the indicator of the event: the labeling function j did not abstain for the data point i .
- the accuracy: $\phi_{i,j}^{Acc}(\lambda_1, \dots, \lambda_p, y) = \mathbb{1}_{\lambda_{i,j} = y_i}$ for the event: the labeling function j gave the true output for the data point i .
- the pairwise correlation: $\phi_{i,j,k}^{Cor}(\lambda_1, \dots, \lambda_p, y) = \mathbb{1}_{\lambda_{i,j} = \lambda_{i,k}}$ which assesses the pairwise correlation of the labeling functions j and k for the data point i .

At this point, we introduce the notion of **labeling matrix**. The labeling matrix Λ which is simply the matrix for which at the coordinate of the row i and column j , there is the output of the labeling matrix j for the data point i . In addition, We note Y the vector of all the true labels y_i . We also define the concatenated vector of the three factor types: $\phi_i(\Lambda, Y)$. That is to say $\phi_i(\Lambda, Y)$ for a given i contained $\phi_{i,j}^{Lab}$ for $j \in \{1..p\}$, $\phi_{i,j}^{Acc}$ for $j \in \{1..p\}$, and $\phi_{i,j,k}^{Cor}$ for (j, k) considered as relevant pairwise correlation by the model.

According to the fact that the underlying graphical model is really linked to the exponential family models, like it is done in [4, 5]. We can assume for the joint distribution of the labeling matrix, and the true label, the following probabilistic model:

$$p_w(\Lambda, Y) = \frac{1}{Z_w} \exp \sum_{i=1}^n \omega^T \phi_i(\Lambda, Y) \quad (1.4)$$

In the case where we have the true labels Y , it is easy to compute the accuracy of the different labeling functions. We can also easily draw the sample correlation matrix of the labeling functions. The only remaining work to do is to find a way to identify the relevant pairwise correlations, because the greedy approach which consists by computing all the possibilities is unthinkable for a large number of labeling function (quadratic cost as discussed previously).

Let's denote $\Sigma_{i,j}$ the sample covariance matrix with respect to the labeling matrix and Y added as a last column (we want the whole graph and Y has to be taken into account as

a vertex). Any zero of this matrix points out labeling functions which are not correlated, or surprisingly not accurate at all (because not correlated to the true label Y). What we are actually interested in is, more precisely, the independence given the outputs of the other labeling functions. We would like to detect these independences, and then to do not compute the pairwise correlations for such pairs. Our approach is based on the result presented in the next theorem [15].

Theorem 1.2. (Zeros of the precision matrix)

Let's X_1, \dots, X_n being some random variables jointly distributed as a multivariate normal law. When these random variables are stacked in a n -dimensional random vector $X \sim \mathcal{N}(0_n, \Sigma)$ with Σ the covariance matrix, and that we can define $\Omega = \Sigma^{-1}$ the precision matrix of X , then the zeros of Ω indicates variables which are independent conditionally on all other random variables contained in X :

$$\Omega_{i,j} = 0 \iff X_i \text{ and } X_j \text{ independent conditionally on all other } X_k$$

In all rigour, we cannot apply this result directly to the case of the labeling matrix, because the labeling functions do not follow a multivariate normal law. However in practice, one can notice that a small value in the precision matrix tends to be the signature of a conditional independence for such labeling functions. Then, as described in [8], we can, in practice, use a threshold to filter out all the small values of the precision matrix. We can then keep a set of relevant pairwise correlations. Using this set, all the factors are now known and the last stage is to compute the parameters in ω . Knowing the true labels y_i this step is straightforward, one could use the maximum likelihood estimator and a gradient descent method of his choice to get the parameters $\hat{\omega}$.

1.5 Generative model without annotated data

A challenging possibility offered by the weak supervision setting is to use heuristic labeling functions to obtain a generative model without knowing any labels y_i . To do so, we first consider the true label y_i of a given data point i as a latent variable. The previous

probabilistic model is still valid, we have:

$$p_w(\Lambda, Y) = \frac{1}{Z_w} \exp \sum_{i=1}^n \omega^T \phi_i(\Lambda, Y) \quad (1.5)$$

The best approach has been developed very recently (March 2019) by Varma, Sala, He, Ratner and Re in [6]. This approach takes advantage of the sparsity of the labeling matrix. They started by considering the covariance matrix:

$$Cov[\Lambda|Y] = \Sigma = \begin{bmatrix} \Sigma_\Lambda & \Sigma_{\Lambda Y} \\ \Sigma_{\Lambda Y}^T & \Sigma_Y \end{bmatrix} \quad (1.6)$$

It is easy to draw the sample covariance matrix Σ_Λ from the labeling matrix, but like the true labels are unknown it is not possible to apply the previous approach. Once again, the precision matrix is a very efficient way to explicit the graph of the correlation structures even without ground truth.

We can define the precision matrix as:

$$P := \Sigma^{-1} = \Sigma = \begin{bmatrix} P_\Lambda & P_{\Lambda Y} \\ P_{\Lambda Y}^T & P_Y \end{bmatrix} \quad (1.7)$$

At first sight, we are now stuck because, even if we know that P is graph-structured (its zeros are the sign of conditional independence). and that we could deduce the whole graph structure from this matrix, like we do not Y we cannot compute P . But the authors introduce a trick. The first block of the covariance matrix is nothing else than the covariance of the labeling matrix, so it is known and, then, we also know its inverse Σ_Λ^{-1} . By using the block matrix inversion formula, we can actually write:

$$P_\Lambda = \Sigma_\Lambda^{-1} + (\Sigma_Y - \Sigma_{\Lambda Y}^T \Sigma_\Lambda^{-1} \Sigma_{\Lambda Y})^{-1} \Sigma_\Lambda^{-1} \Sigma_{\Lambda Y} \Sigma_{\Lambda Y}^T \Sigma_Y^{-1} \quad (1.8)$$

By noting $k = (\Sigma_Y - \Sigma_{\Lambda Y}^T \Sigma_\Lambda^{-1} \Sigma_{\Lambda Y})^{-1}$ and $C = \sqrt{k} \Sigma_\Lambda^{-1} \Sigma_{\Lambda Y}$, we can then write:

$$\Sigma_\Lambda^{-1} = P_\Lambda + CC^T \quad (1.9)$$

The goal is then to decompose Σ_Λ^{-1} in order to retrieve P_Λ . By basic algebra, we know that CC^T is a matrix of rank 1. In addition, P_Λ is a sparse matrix. To tackle this problem

of decomposition, it seems natural to make use of the low rank of CC^T , and to perform a principal component analysis. However, like the authors of [6] recalled, it has been shown that a usual principal component analysis cannot recover a low-dimensional subspace in the presence of a bounded noise, like the entry of the sparse matrix P_Λ . We have, then, to perform a variant called robust principal component analysis [17].

Once this decomposition has been done, We can then explicit the graph formed by the labeling functions λ_j and the true label y , and, by applying the same method as in the case where we have annotated data, we can, finally, compute $p_{\hat{w}}(y|\lambda_1, \dots, \lambda_p)$ for the different possible values of y .

1.6 Discriminative model

There is a choice to be made. One can choose for a given input x_i to simply output the y_i which maximises $p_{\hat{w}}(y_i|\lambda_1, \dots, \lambda_p)$ obtained by the generative model alone.

Another option is to use a second model which will be, after training, our end-to-end model. It is called discriminative model. The generative model is then only used as an annotator: for each data point x_i , it gives us the estimate of $p_{\hat{w}}(y_i = 1|\lambda_1, \dots, \lambda_p)$ and $p_{\hat{w}}(y_i = -1|\lambda_1, \dots, \lambda_p)$. The discriminative model will learn how to map the same initially unlabeled input x_i to the probabilistic label returned by the annotator.

This second model can be any deep learning algorithm suitable for our task. The difference with the classic case of a supervision learning will only occur during the training: we do not know for sure what is the true label y_i , and in some cases the generative model can completely misjudge a label: they are only "weak" labels. The true label is then only a latent variable, we can not observe it, and we have to rely on the probabilistic labels provided by the generative model which actually infers the probability distribution of y_i . It is the following discrete conditional law: $\sum_{y \in \Omega(Y)} p_{\hat{w}}(y|\lambda_1, \dots, \lambda_p) \delta_Y^y$ which becomes for our binary classifier $Y \sim p_{\hat{w}}(y = 1|\lambda_1, \dots, \lambda_p) \delta_Y^1 + p_{\hat{w}}(y = -1|\lambda_1, \dots, \lambda_p) \delta_Y^{-1}$.

In order to have an efficient training, we have to adapt the loss function of the discriminative model in relation to the fully supervised case: the model has to be noise-aware. More specifically, as in [5], we can train the model to minimize the noise-aware loss (\widehat{loss})

defined by:

$$\widehat{loss} = \sum_{i=1}^n \mathbb{E}_{y \sim \hat{Y}_i} [loss(x_i, y)] \quad (1.10)$$

where $loss(x_i, y)$ represents a typical loss in a fully supervised setting, for example the squared error, or the cross-entropy [22].

The generative model relies on the labeling functions, which in most of the cases are based on rules of thumbs implemented by humans, and using basics reasoning. In some situations, a deep learning algorithm may find more complex dependencies that these heuristics labeling functions. The goal of a discriminative model is then to generalize above the basic scope of the generative model, and to detect more complex relations. Based on these dependencies (if they exist) we can expect that the discriminative model will out pass the generative one, but it is not always the case.

According to the authors of [5], it can be proved that asymptotically (for the number of unlabelled data given to the generative model) the discriminative model, under a weak supervision, can reach the same level of performance as for a usual fully supervised setting. In the same way, we have already seen a specific situation where it was the case (page 19).

In order to illustrate the choice of using a discriminative model or not, it is relevant to have a look to the experimental results of [5].

Task	Snorkel (Gen.)				Snorkel (Disc.)				Hand Supervision		
	P	R	F1	Lift	P	R	F1	Lift	P	R	F1
Chem	78.6	21.6	33.8	+16.2	87.0	39.2	54.1	+36.5	-	-	-
EHR	77.1	72.9	74.9	+2.7	80.2	82.6	81.4	+9.2	-	-	-
CDR	52.3	30.4	38.5	+9.1	38.8	54.3	45.3	+15.9	39.9	58.1	47.3
Spouses	53.5	62.1	57.4	+42.0	48.4	61.6	54.2	+38.8	47.8	62.5	54.2

Figure 8: Performances comparison of Snorkel for different user-cases

The results come from different data sets and different tasks. The authors used Snorkel, a system which allows the rapid development of weak supervision infrastructures (and is particularly useful in the medical field) [19].

They used the following tasks:

- Chem is the extraction of potential drug interactions for some substances describe in scientific papers available online.
- EHR stands for Electronic Health Records, the system tries to detect complications after a medical interventions.
- CDR is the extraction of chemicals and the diseases associated to them from an online data set.
- Spouses is a task which consists of the extraction of mentions of spouses from documents.

There are a couple of points that we can observe on this table. First of all, the performances, under the weak supervision setting, are really close to the ones obtained when the learning is supervised (called Hand Supervision in the table). The "lift" column corresponds to the improvement of the F1 score from the case where a simple distant supervision is used (all these tasks are done using online database, so the distant supervision is particularly relevant).

As for the Discriminative model (Disc.) we can observe that on average the performances are better than in the case where only a generative model is used. However, It can be wrong like for the precision of the examples CDR and Spouses, and for the recall of Spouses.

The following figure summarizes a typical weak supervision architecture:

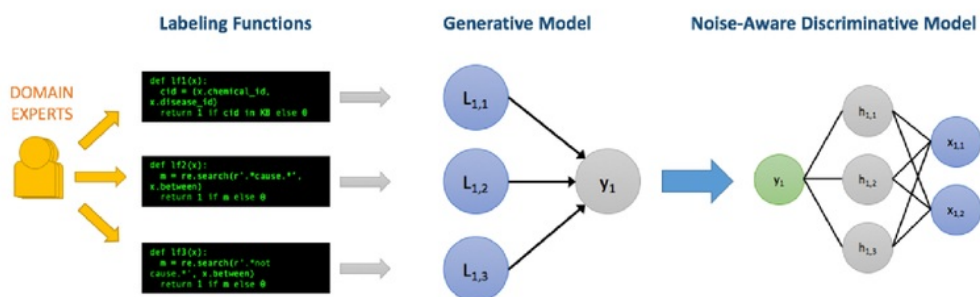


Figure 9: Illustration of a typical weak supervision infrastructure (from *Snorkel blog* [24])

2 Part II: dividend extractions

2.1 Problem setup

I work as a Data Scientist intern in the corporate action team of the Global Data division of Bloomberg in London. A corporate action is an event within a company, usually approved by the board of directors, and which has consequences for the stakeholders. As for "Data scientist", it is a name that can be used for a lot of different jobs. Mine is to automate some tasks performed by the analysts of my company.

The automation of processes by algorithms is a significant challenge for most companies, and, of course, Bloomberg does not escape this rule. My work is more precisely focused on the automation of dividend extractions from news. Several times a year, some companies publish news announcing the amount and other features (frequency, date of payment and so on ...) of the dividends which will be paid for the next period. This information is important for a lot of different financial actors, and quite naturally for Bloomberg as well, which aims at providing accurate information to as many clients as possible, in the shortest delay.

For each dividend, we aim at extracting 5 fields:

- the amount of the dividend, so the value that will be paid by the company for the next period
- its currency, which is mainly dollar for the documents I have been working on. This field is not really crucial at the moment, because its extraction is straightforward
- the type of the dividend, which can be cash, special, income, interim, partnership distribution, suspended...
- its frequency which can be quarterly, semi annually, monthly...
- its date of payment
- its date of record.

It can be noticed that, for a given news, the frequency and the type are necessary from a set of labels, so this is really a typical problem of classification. It is not the case for the other labels.

PARK ELECTROCHEMICAL CORP. DECLARES CASH DIVIDEND Melville,
New York, Wednesday, June 12, 2019.

The Board of Directors of Park Electrochemical Corp. (NYSE – PKE) declared a regular quarterly dividend of \$0.10 per share payable August 6, 2019 to shareholders of record at the close of business on July 5, 2019. Park has paid 34 consecutive years of uninterrupted regular, quarterly cash dividends, without ever skipping a dividend payment or reducing the amount of the dividend. The Company has paid approximately \$509 million in cash dividends, or \$24.85 per share, since the beginning of its 2005 fiscal year. Park Electrochemical Corp. is an Aerospace Company which develops and manufactures solution ...

Figure 10: Example of an extract of a dividend news

Just above, I put an extract of a dividend news for the company Park Electrochemical Corp. The dividend that should be extracted from this news is the following:

Dividend to be extracted	
amount:	0.10
currency;	dollar
type:	cash*
frequency:	quarterly
date of payment:	08/06/2019
date of record:	05/07/2019

**cash and regular corresponds to the same type of dividend.*

I chose this news to give an idea to the readers, but actually some news are quite different, and more difficult. We can also already notice that the layout can play a role.

To do so, I have at my disposal a data set of 500 documents which have been annotated by a couple of Bloomberg analysts. In addition, the annotations provide an important characteristic which is that the location of the different labels is annotated in the document format. I can also have thousands of non-annotated documents.

The choice of the approach is not easy. First of all, the very high standard in term of accuracy seems pretty complex. Bloomberg attaches a big importance to not publishing any wrong data. It is particularly hard to reach this level of performances by using pure machine learning methods. In addition, most of the machine learning algorithms are rightly considered as black boxes, and the predictions often rely on computations of millions of parameters. Thus, in addition to be difficult to explain, their development require time, and large annotated training data sets. An approach purely based on rules can seem tempting, especially since we can observe some kind of similar structures within the different documents of the corpus, and for a given news it is easy to find a small set of rules of thumbs, based on the different words and punctuation signs of the document, which could extract all the required information.



Figure 11: Words which appear the most in the sentences containing the amount of the dividend to be extracted

For example, on the word cloud above, we can easily notice some key words which are typical of a sentence which contains the amount of the dividend to be extracted.

Otherwise, the news look quite different from each other, so it seems really complicated

to write a set of rules that will extract the dividends with a high precision, and be valid for enough documents.

2.2 Requirement specifications

2.2.1 Time

The process has to be quick. The value of the information published by Bloomberg is really linked to the time taken for the publication. An analyst needs minutes to process a document, and the target of the algorithm is then to be in seconds. I fixed the maximum time to extract a document at 30 seconds.

2.2.2 Performance metrics

In machine learning, there are a couple of evaluation metrics which are used by the whole community of researchers and allow easy comparisons between the different models.

First, let's define the confusion matrix in the case of a binary classification in order to be the most clearest possible.

	predicted		
Truth		Spam	Not spam
Spam		True positive	False negative
Not spam		False positive	True negative

Figure 12: confusion matrix of a simple binary classifier of emails

According to the confusion matrix, we can deduce that for a given classification, there

are 4 possible scenarios: true positive, false positive, true negative and false negative. Based on these scenarios, we can define 2 fundamental quantities [14]:

- The precision is defined as $precision = \frac{true\ positives}{true\ positives + false\ positives}$. It represents the ability of the model to extract true information among the information extracted.
- The recall is another important performance metric. We have $recall = \frac{true\ positives}{true\ positives + false\ negatives}$. The recall represents then the proportion of information extracted correctly, among the information that it was possible, in theory, to extract.

There is a clear trade-off between these 2 performance metrics. For example, a model can reach a higher precision if it does not extract documents which "look" too difficult, but the recall will then be lower. That is why it is quite natural to introduce a third performance metric which combines both precision and recall.

- The F1 score is defined as $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$. This score is then the harmonic mean of the precision and the recall.

In term of performances, it is expected of me that my model reaches a very high level of precision. Bloomberg can not publish wrong data. The target, I chose to reach is then 99%. It represents a very challenging performance level, because it is rare to find any machine learning algorithms which can naturally reach this level by itself. In term of recall, I chose the target to be less demanding. I want the model able to reach 60% in a first time.

This unbalance between the precision and the recall suggests that somehow the model should make use of a calibration stage to "sell" a bit of recall, in order to "buy" a bit of precision. We can then look for a way to filter out the documents which "look" hard for the model (which assume a good understanding of its weaknesses) in order to reduce the number of false positives. However, as a consequence, the number of false negatives is going to increase.

Another key element is that the goal is to have the highest performances possible at the document level. For example, let say that you reach 99% of precision for all 6 labels.

If, for a test data set of 100 documents, the errors never occur for the same documents but are scattered you will end up with only 94 documents without any false positive. So, a good property of the model is to have a kind of correlation among the successes of the extractions of the different labels.

Let's now summarise my target:

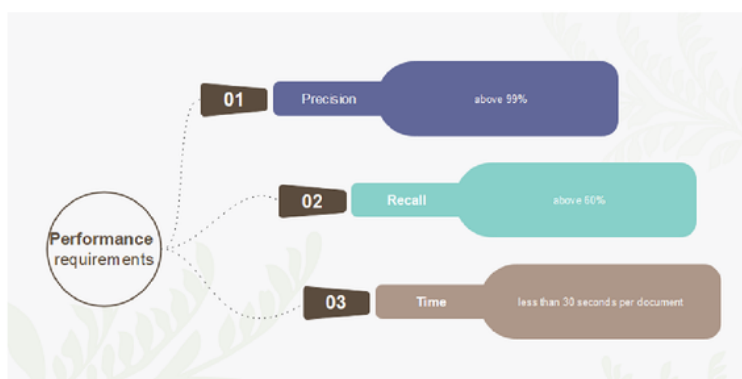


Figure 13: Summary of the performance requirements

Remark: These values have been chosen arbitrary for this report

2.3 The difficulties

First of all, some news are very special. The information can appear under different forms, at different locations. For some other news, all the labels are not present and then if, for instance a company does not specify the type of the dividend, we expect that the model extracts "missing value". Some companies, sometimes, recall the dividend for the last period, before to announce the new one. The model should then extract the right one (for the next period). It also happens that the information is actually contained in tables. Some news can also be long (a couple of pages).

One very tricky situation occurs when the model has to actually deal with a multiple dividends event. It can happen that the company announces several dividends in the same news. The ultimate goal is to be able to extract all the dividends, and to link the

information together (which amount comes with which type and so on ...). Most of the time, this multiple dividends event concerns companies which announce a "main" dividend for the next period and, in addition, that a special dividend will also be paid.

2.4 Description of the data pipeline

The data pipeline is a data engineering concept which represents the path of the data within the infrastructure since its reception, from the source, to the production of the final product. In my user-case, it is all the different stages from the reception of a news to the extraction of all the labels.

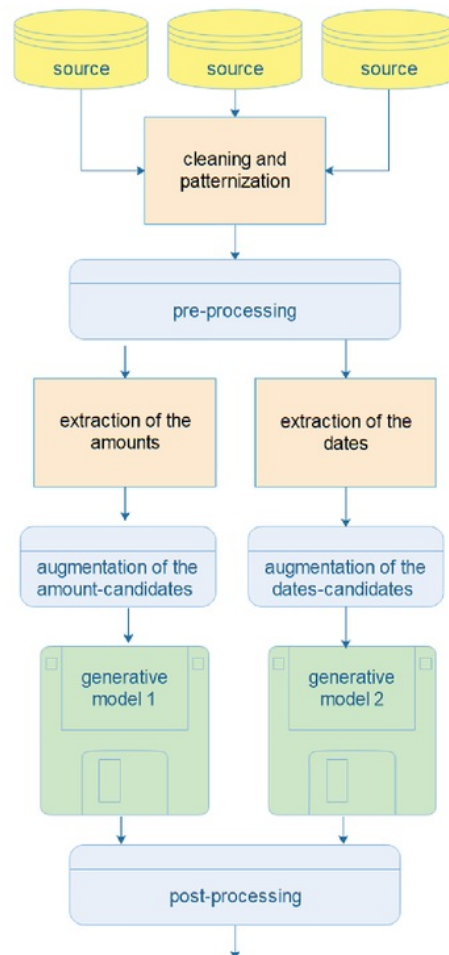


Figure 14: The different stages of my data pipeline.

I would not call my approach just as "weak supervision", because what I use is really a global approach, and for which I see more the weak supervision setting as a tool.

The previous approach of my team performs the dividend extraction based on extracting the most likely fields for each label from a dividend news. Thus, the model could extract "quarterly" as the most likely dividend frequency in the news, and "special" as the most likely dividend type, and so on. Then to group all these features into the dividend extracted by the model works as a post-processing stage. It seems that this approach works quite well for the simplest examples.

However, the main shortfall is that there is no straightforward way to tackle the multiple dividends event. In addition, this approach does not take fully advantage of the relative proximity of the labels in term of location within the documents.

What I tried to apply is a different approach. Rather than to extract all the fields, I wanted the model to be able to extract a "relation" dividend, an object which contains all the fields. It makes sense because all the different fields are kind of related, for example, because the information allowing the extraction of the different labels are often located nearby in the news. This is also probably how a human would perform the task. Let's now have a look to the data pipeline that I used.

2.4.1 Cleaning and patternization of the news

The entry point is a document under HTML format. I chose to extract, using the layout, three different parts. The title of the news, the plain text of the rest of the news, that we from now on call body, and the possible tables. The titles are used in the pre-processing stage, in a few words, if it happens that an information for a given label is already present in the title, the work is already done (almost actually, we will see later why).

We know focus on the plain text of the news which contains almost every time all the information which have to be extracted.

There is a necessary cleaning of the document based on a lot of rules that I wrote myself. I won't detail these rules in this paper, because of the poor academic relevance. However, this stage is important because, depending on the source, the news can have

a couple of problems, like missing spaces, or to be polluted by non-relevant characters coming from the layout, or the process of getting the news.

During the cleaning, the model only keeps the relevant sentences in order to cut down the size of the documents, which can reach a couple of pages. To do so, I only kept the sentences which contain the key words: dividend, share and payment. This filtration requires that the model is able to handle the punctuation signs. This is not easy. For example, looking at the dots is not enough in order to detect the different sentences of a news, because, there are also the decimal numbers, the acronyms and diverse abbreviations (such as "Inc."). The handlers have to be good enough. For this task, I implemented a lot of rules, that I wrote myself to deal with the maximum number of possible situations. I thought to use some more elaborate criteria than just these 3 key words, but this one has proven to be a really good trade-off, conserving a smaller extract of the news and always, to my knowledge, the evidence to extract all the fields needed.



Figure 15: Average size (number of characters) reduction after keeping only the relevant sentences

We can observe on the graphic just above that the size of the document is on average significantly reduced after filtration. This reduction is particularly interesting in term of time, because even if, at this stage, we do not know what is going to happen in the following stages of the data pipeline, it is quite natural to think that with smaller document, it will

have a less important computational work to perform the extraction.

2.4.2 Extraction of the candidates

The next stage is a very important one, clearly fundamental to the success of the approach. At a character level, using a lot of rules that I wrote, the model extracts from one dividend news all the amounts of money present in the news. It is quite complicated, because there are a lot of different ways to write an amount of money, and it also depends on the currency. As an example, these are several different ways to write the same amount of money: "\$0.99", "\$.99", "\$ 0.99", "99 cents", "0.99 dollar", "0.99 Dollar", "USD 0.99", "ninety nine dollar", "ninety-nine dollar". So there are a lot of ways and this is just for an amount in dollar, and assuming that the text has been perfectly cleaned because it can happen that one word is stuck just after the amount without any space.

We now call these amounts of money candidates. They are the candidates to be the amount of the dividend we want to extract. As an example, in the extract of news below, there are three candidates:

PARK ELECTROCHEMICAL CORP. DECLARES CASH DIVIDEND Melville,
New York, Wednesday, June 12, 2019.

The Board of Directors of Park Electrochemical Corp. (NYSE – PKE) declared a regular quarterly dividend of \$0.10 per share payable August 6, 2019 to shareholders of record at the close of business on July 5, 2019. Park has paid 34 consecutive years of uninterrupted regular, quarterly cash dividends, without ever skipping a dividend payment or reducing the amount of the dividend. The Company has paid approximately \$509 million in cash dividends, or \$24.85 per share, since the beginning of its 2005 fiscal year. Park Electrochemical Corp. is an Aerospace Company which develops and manufactures solution ...

Figure 16: Extraction of the amounts of money

It is crucial that this extraction is really efficient, because at this stage, if an amount is missed, the whole extraction will fail. There is no way to recover from that.

I needed to spend a lot of time working on this stage which is crucial to the success of the extraction. Finally this extractor became really efficient: it extracts, on average, 5 candidates per dividend news, but this figure varies a lot, depending obviously on the size of the document. In more than 99.9% of the cases, the true amount is among these candidates extracted. In addition, in 51% of the cases, there is only one amount of money extracted which means that this extractor is able to extract the amount of the dividend in half of the cases without having performing any machine learning, or even statistical analysis. This rate of success plays an important part in the success of the whole data pipeline.

One of the drawback is that if we are, for example, interested in expanding this model for other news from other countries, and using different currencies, there is work to be done in order to adapt this extraction (Obviously the way to write the amount of money depends on the currency).

Without going further in the data pipeline, there is now a clear question which is: how do we choose among this list of amounts the most likely to be the value of the dividend that we actually want to extract from this news?

2.4.3 Building of the labeling matrix

It is now that weak supervision comes into play. The choice of the most likely amount relies on a set of heuristic labeling functions. Why heuristic? Because, at this stage, it is only simple rules of thumbs. The rules that I wrote are based on the vocabulary used, and then try to detect signals based on key words, that I call pattern.

Here is an example of symbolic heuristic labeling function:

```
If there is "per share" after the amount then the output of the function is 1  
Otherwise the output is -1
```

Figure 17: Example of a symbolic heuristic labeling function

(Almost) All the heuristic labeling functions I wrote are based on a simple if-else schema. Eventually a condition can be tested before to enter this schema, in order to check that this labeling function is relevant (we will speak about that later on). If a signal is detected the function outputs something, otherwise it outputs something else. The possible outputs of a function are 1, 0 or -1 . 1 means that the heuristic labeling function "trusts" the candidate-amount to be the one of the dividend we want to extract, -1 if it does not, and 0 if the labeling function abstains: using just this specific signal, it is not deemed conclusive.

For the heuristic labeling function of the example, the output is either 1 or -1 (it can not abstain). The signal is: if the "evidence" of the two words "per" and "share" are detected after an amount, this amount is likely to be a dividend, and then the function outputs 1, otherwise the labeling function does not trust this amount and then returns -1 .

So, in the case of the example, the amounts 0.10 and 24.85 are followed by "per share", so the output of the function is 1. On the other hand, for 509 the output is -1 .

Another example of labeling function could be based on the key word "million". If a candidate is followed by "million", it is very unlikely to be the amount of the dividend that the model has to extract. As a consequence, such heuristic labeling function should outputs -1 . Otherwise, that is to say if there is not "million" after the candidate, it does not indicate anything in term of the likelihood that the candidate is correct or not, so the labeling function should abstain (and not output 1).

With the two examples of heuristic labeling functions, one can notice that there is a

choice of polarity to be made. When you write an heuristic labeling function, you have to choose if the signal is associated to a correct, or a wrong candidate. You also have to choose, whether or not, the function should abstains, or "votes" for the "inverse" polarity, if the signal is not detected.

Using the two previous heuristic labeling functions, the labeling matrix of the example can now be built.

Each line of this matrix is attributed to one candidate (an amount in our current user-case), and each column represents an heuristic labeling function. Thus, at the position (i, j) of the matrix is located the output of the heuristic labeling function number j to the candidate i .

candidates	Labeling function 1	Labeling function 2	
0.1	1	0	0
509	-1	-1	-1
24.89	1	0	0

Figure 18: Example of a labeling matrix for the two heuristic labeling functions defined above

2.4.4 The trick of the augmented candidate

At this stage, we now have a list of amounts of money, present in the news, that we are working on, and, in addition, a statistical method to assess the most likely one. The goal is to extract the whole object-relation dividend with all the labels.

For each news, after having detected the candidates of the amount, we can now look in the neighbourhood to try to detect evidence of other information concerning the other labels. It is particularly efficient for both the dividend type and frequency, because there are a number of key words around the amount which provide accurate and decisive information.

PARK ELECTROCHEMICAL CORP. DECLARES CASH DIVIDEND Melville, New York, Wednesday, June 12, 2019.

The Board of Directors of Park Electrochemical Corp. (NYSE – PKE) declared a regular quarterly dividend of \$0.10 per share payable August 6, 2019 to shareholders of record at the close of business on July 5, 2019. Park has paid 34 consecutive years of uninterrupted regular, quarterly cash dividends, without ever skipping a dividend payment or reducing the amount of the dividend. The Company has paid approximately \$509 million in cash dividends, or \$24.85 per share, since the beginning of its 2005 fiscal year. Park Electrochemical Corp. is an Aerospace Company which develops and manufactures solution ...

Figure 19: Detection of evidence in the vicinity of the candidates

In this example, surrounding the amount 0.10, the key words "quarterly" and "regular" can be detected, which means that if 0.10 (dollar) is the amount that we want to extract, the type is probably cash (same meaning as regular), and it is a dividend paid quarterly. 509 also looks at first sight like a cash dividend. As for 24.85, there is not any obvious key words in the vicinity. At this point it is important to specify that, when I speak of vicinity, I refer to a topology clever enough to deal with the punctuation. In the case of 24.85 the information is separated by a coma, and, we attach a big importance to punctuation signs. For the moment, let's focus on the abstract ideas.

It is now that the fundamental trick I used occur. Without this trick it is, as far as I am concerned, really not clear that this approach will work.

Idea (*Augmented candidate*):

The presence or the absence of signals for the other labels in the neighbourhood of the candidate-amount strengthens or weakens the likelihood that this amount is actually the one that should be extracted by the model.

Then, based on this idea, detecting the evidence, or signals surrounding the amounts actually works both way. By both way, what I mean is that we can use the most likely amount to detect, in its vicinity, evidence of the most likely outputs for the other labels (for example the word "cash"), but the presence, or the absence, of evidence in the neigh-

bourhood of one given amount strengthens or weakens the likelihood that this amount is actually the one that we are looking for the extraction.

First of all, it is clear that when we speak of "neighborhood" of an amount, we have to choose an efficient topology based of course on the number of words separating the different entities, and which is punctuation aware. For example in the piece of sentence: "The value of the monthly dividend paid by our company according to the board of director is \$3.4 (which represents \$7.2 annually)", we definitely wants the amount \$3.4 to be attached with "monthly", and not "annually". Let's now look again at the example.

PARK ELECTROCHEMICAL CORP. DECLARES CASH DIVIDEND Melville,
New York, Wednesday, June 12, 2019.

The Board of Directors of Park Electrochemical Corp. (NYSE – PKE) declared a regular quarterly dividend of \$0.10 per share payable August 6, 2019 to shareholders of record at the close of business on July 5, 2019. Park has paid 34 consecutive years of uninterrupted regular, quarterly cash dividends, without ever skipping a dividend payment or reducing the amount of the dividend. The Company has paid approximately \$509 million in cash dividends, or \$24.85 per share, since the beginning of its 2005 fiscal year. Park Electrochemical Corp. is an Aerospace Company which develops and manufactures solution ...

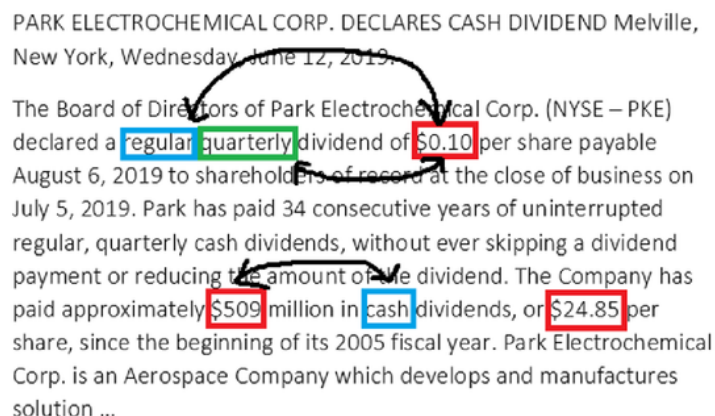


Figure 20: Detection of evidence in the vicinity of the candidates

We can easily notice that the amount 24.85 is not surrounded by any obvious signal for the frequency, or the type of the associated candidate.

All things said, the question is now how do we incorporate this idea in the model. I used what I call an "augmented" candidate. The extraction of the amount, and the extraction of the type and frequency of the dividend, have two different logics. Extracting the amount of the dividend is like choosing one of the amount of money somewhere in the document, whereas for the type, or the frequency, it is all about choosing a label at document level. In addition, for the type and the frequency, there is a finite set of possibilities, and for each one a small number of key words, which are real evidence. For

example, the frequency of the dividend can only takes the values: monthly, quarterly, semi annually, annually, irregular or missing value.

I transformed each candidate to a set of candidates which represents all the possible combinations for the fields frequency and type. It means that the candidate amount whose value is 0.1 is going to be "augmented" to a list of the following candidates:

Amount	Type	Frequency
0.1	cash	quarterly
0.1	special	quarterly
0.1	income	quarterly
0.1	interim	quarterly
0.1	suspended	quarterly
0.1	cash	monthly
0.1	special	monthly
0.1	income	monthly
0.1	interim	monthly
0.1	suspended	monthly
0.1	cash	semi-annually
0.1	special	semi-annually
0.1	income	semi-annually
0.1	interim	semi-annually
0.1	suspended	semi-annually

Figure 21: List of the "augmented" candidates for the amount 0.1

Note that as an example, I just kept some possible values for the type and the frequency of the dividend. One can notice that, from now on, we are going to have a lot more candidates (We will see later how to actually reduce this number of potential candidates, in order to increase the speed of the extraction).

What happens now is that, for each candidate, we can write heuristic labeling functions, which are going to be apply to any of these fields, and it changes a lot of things. To understand that let's return to our previous example.

PARK ELECTROCHEMICAL CORP. DECLARES CASH DIVIDEND Melville,
New York, Wednesday June 12, 2019.

The Board of Directors of Park Electrochemical Corp. (NYSE – PKE) declared a regular quarterly dividend of \$0.10 per share payable August 6, 2019 to shareholders of record at the close of business on July 5, 2019. Park has paid 34 consecutive years of uninterrupted regular, quarterly cash dividends, without ever skipping a dividend payment or reducing the amount of the dividend. The Company has paid approximately \$509 million in cash dividends, or \$24.85 per share, since the beginning of its 2005 fiscal year. Park Electrochemical Corp. is an Aerospace Company which develops and manufactures solution ...

Figure 22: The amounts and their neighbourhoods

Let's imagine a very simple function which returned 1 if "cash" is one of the word of title and if the candidate's type is "cash", otherwise the function abstains and returns 0. This example of heuristic labeling function is going to favour the true candidate: [0.1, cash, quarterly] over for example this wrong one: [24.85, cash, quarterly]. Let's recall that previously, our very simplistic generative model composed only of two example of heuristics could not make any difference between these two candidates.

The fact that we can write labeling functions for each fields increased a lot the number of possible evidence/signals that we can detect, and so also the number of functions. Then the full entity formed by the amount, the type and the frequency becomes stronger, or weaker as signals have been detected for any of the fields in the entity-relation dividend.

2.4.5 Extraction of the candidates for the dates

For the dates, it works in a slightly different way, because it can happen that the dates are further to the amounts of the dividend than the other labels. So I did a second generative

model for the two dates that we want to extract (the date of payment and the record date, as a reminder).

Initially, I proceeded the same way. First of all, an extractor detect all the possible candidates for the dates within the news. We can then, following the same principles, write some heuristics labeling functions to detect signals for each of these two labels.

PARK ELECTROCHEMICAL CORP. DECLARES CASH DIVIDEND Melville,
New York, Wednesday, June 12, 2019

The Board of Directors of Park Electrochemical Corp. (NYSE – PKE) declared a regular quarterly dividend of \$0.10 per share payable August 6, 2019 to shareholders of record at the close of business on July 5, 2019. Park has paid 34 consecutive years of uninterrupted regular, quarterly cash dividends, without ever skipping a dividend payment or reducing the amount of the dividend. The Company has paid approximately \$509 million in cash dividends, or \$24.85 per share, since the beginning of its 2005 fiscal year. Park Electrochemical Corp. is an Aerospace Company which develops and manufactures solution ...

Figure 23: Detection of signals for the dates candidates

To improve the model, I then incorporated something, in nature, very similar to the approach for the amount. It is based on a few facts: the record date and pay date are always nearby in the news, it is very unlikely to find another candidate-date between the two true labels and, in addition, the record date is always anterior to the date of payment.

The augmented candidates for the dates are then the different pairs of dates extracted in a row from the news, and the date of payment corresponds to the most recent one. Thus, in the same way that for the entity grouped around the amounts, an important number of signals for the date of payment could compensate a document which announces a record date in a special way: the different candidates becomes stronger or weaker as signals have been detected for any of the two dates of the candidate.

Actually, to be accurate these two candidates formed around the amount and the dates are not completely independent. I implemented an heuristic labeling function which

values the amount-candidates, if their amount is closed to the candidate-dates extracted. Sometimes, this heuristic labeling function will be wrong because the dates will be further, but it does not matter because the weak supervision model will learn how accurate is this rule, and also how likely it is to be true or wrong in relation with the outputs of the other labeling functions (by the dependency structures). This is one example among a lots of other possibilities to point out how powerful the weak supervision setting can be.

2.4.6 Generative model

Initially, an heuristic labeling function is just a simple rules of thumbs. The next stage is to use a generative model. Each candidate, which is, at the moment, represented as a line of the labeling matrix containing the 1, 0 and -1 of the different labeling functions is then converted into a probability. This probability represents the likelihood, according to the generative model, that this candidate is the true dividend that should be extracted by the model. Then, for both the dates-candidate and the amount-candidate, the model extracts the different fields with the maximum probability and group them into the final dividend.

At the moment, the data pipeline does not contain any discriminative model. As we will see later the results are already very good, but it is a way of improvement which is still on the table. In addition, the data pipeline only extracts a single dividend so far. The model deals with the multi-dividends event in a post processing stage.

2.4.7 Calibration stage

The goal of the extraction is to reach a very high level of precision. For this target, it is acceptable to have a lower level of recall. So it can be interesting to think about how the model can filter out news which look "difficult" then the number of false positive will be reduced, and it will increase the precision. However, the number of false negatives will also increase. As a consequence, the recall will be lower.

There are different criteria which could be used. The most naive ones are to filter out the news that we deem too long to trust the model, or the documents for which the model

extract an important number of candidates. One more interesting way, is to believe in the consistency of the model, and then, to attach two new labels to the dividend extracted: the probability given by the two generative models. We can then consider these probabilities as level of confidence of our model in the final output, and filter out the documents for which the probability is lower than a threshold. This threshold is given by the trade-off precision/recall chosen.

The most challenging entity for reaching a high level of precision is the one formed by the amount, the frequency and the type (dates are easier to extract). Let's have a look:

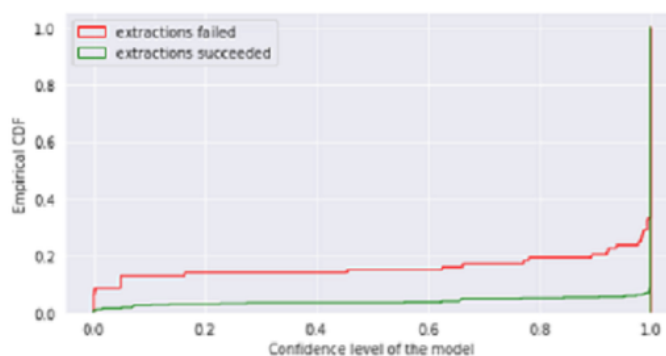


Figure 24: Boundary of a classifier in a fully supervised setting

The graph above represents the normalized cumulative number of correct and wrong candidates extracted in function of the probability level attached by the generative model. We can observe that the relative proportion of wrong candidates for the lower level of confidence is more important. For the correct candidates extracted, the level is concentrated very close to 1.

This is a really interesting result for two reasons. The first one, without actually knowing the performances of the model, you can convince yourself that the generative model has a kind of global consistency. For each candidate, it returns the probability that this candidate is the one that should be extracted. Like explained previously, the dividend extracted is then the candidate which the highest probability and this graph tell us that actually when the probabilistic label attached to its entity is lower, in term of proportion,

we can deem this extraction as risky (At this point, we should be cautious. I find that this graph is a good illustration of the phenomenon, but like it represents normalized quantity and, hopefully, there are overall more successful extractions, we can not conclude yet).

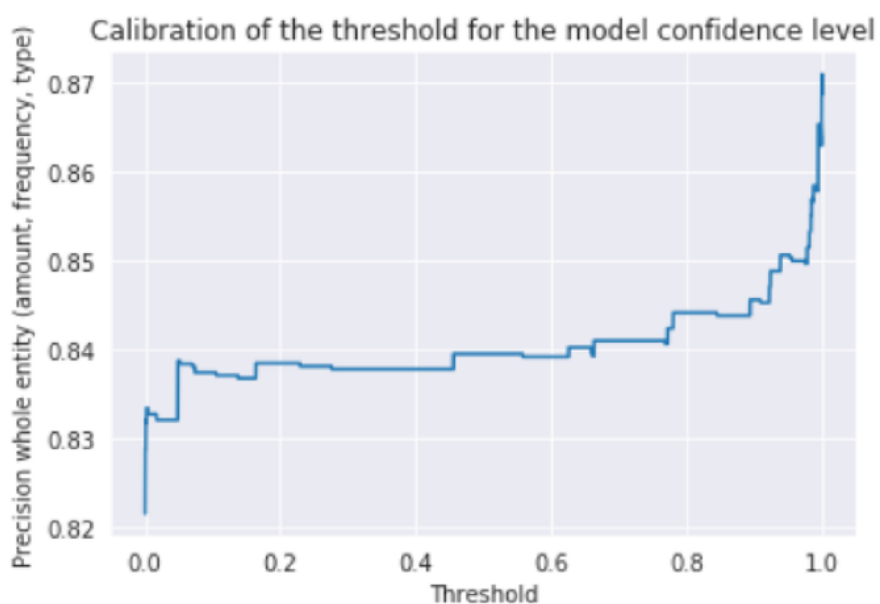


Figure 25: Boundary of a classifier in a fully supervised setting

For the entity composed of the amount, the frequency and the type of the dividend. If we fix a threshold, and then we filter out all news for which the level of confidence of a generative model, composed of 50 heuristic labeling functions, is below this threshold, then we observe a improvement of the precision (Note for clarity: if at least one of the 3 labels is wrong, it is a false positive at the entity level).

We have then showed that we can use a threshold in order to calibrate our extraction in relation with our target precision/recall.

2.4.8 Post-processing rules

I observed that a lots of errors of the model comes from an incorrect estimation of the strength of a signal. I mainly interpret these weaknesses as default of my training data set: some situations are not represented enough. For example, in the news, there are a few cases of a fixed-to-floating dividend which is also announced. For a non-expert, it looks like a dividend that we would like to extract, but it is never the case and there is always the dividend we are interested in which is announced somewhere else in the document. This situation does not happen a lot, so the weak supervision model does not learn how wrong can be the candidates close to the mention "fixed-to-floating". As a consequence, an heuristic labeling function which returns -1 when "fixed-to-floating" is in the sentence of a candidate won't work.

That is why, in order to reach very high performances, the model has to have post-processing rules which compensate the weaknesses of the model. The generative model can have a lots of labeling functions, so it has to be balanced a lot of different signals and without a very large training data set, I consider that a set of post processing rules is a very efficient way to improve the model. Otherwise, one major drawback is that this stage is not consistent with the calibration stage explained previously. These rules actually tend to favour candidates for which the model was wrong so which have a lower probability, and which are then likely to be filtered by a threshold. Based on my experience of this model, I consider that the post-processing stage is more important because there is really a small margin of "almost" free improvement by doing small changes. The calibration stage can more be seen as a study of the global consistency of the model, of the capability of the two generative models to separate the correct candidates from the wrong ones.

2.4.9 Multiple dividends event

It happens that a company announces several dividends in the same news. The main situation is when a special dividend will be paid for the next period in addition to the "main" dividend.

Based on my approach, I implemented a set of rules to deal with the multiple dividends

event in the "special" case. According to what I observed, it works pretty well. The problem is that I do not have annotation at the moment with information related to the multiple dividend event, so it is theoretical and I do not know how performing is this approach.

Let's look into this post processing stage in an example:

Declared Dividend of **\$0.49** Per Share for Quarter Ended June 30, 2019 June 30, 2019
 Dividend Income: \$7.2 million, or \$0.42 per weighted average diluted share*
 Dividends: Paid \$0.38 per share in a Regular Dividend, plus \$0.10 per share in a
 Supplemental Dividend **Total** Dividends for the quarter of **\$0.48** per share* Net
 Realized and Unrealized Portfolio Appreciation: \$3.6 million, or \$0.21 per
 weighted average diluted share Net Asset Value ("NAV") per Share: \$18.62* Pre-
 Net Investment Income: \$24.8 million for the fiscal year, or \$1.48 per weighted
 average diluted share versus \$1.02 per weighted average diluted share in the
 previous fiscal year, representing 45% growth* Dividends: Declared and Paid
 Total Dividends of \$2.27 per share versus \$0.99 in the prior fiscal year,
 representing 129% growth* Supplemental Dividend Program: During Fiscal Year
 2019, announced dividend of \$0.10 per share to be paid on a quarterly basis out of
 Undistributed Taxable Income ("UTI"), generated primarily through realized capital
 gains and Pre- Net Investment Income in excess of Regular Dividends UTI Balance
 at the end of the fiscal year ended March 31, 2019 was \$1.14 per share In
 commenting on the Company's results, Bowen Diehl, President and Chief Executive
 Officer, stated, "Fiscal year repurchased a total of 46,363 shares at an average
 price of \$16.67 per share, including commissions paid. The Company currently has
 approximately \$9.2 million available for additional repurchases under the
 program. On May 30, 2019, the Board declared a **total** dividend of **\$0.49** per share
 for the quarter ended June 30, 2019, comprised of a regular dividend of \$0.39
 per share and a supplemental dividend of \$0.10 per share. The Company's dividend
 will be payable as follows: When declaring dividends, the Board reviews estimates
 of taxable income available for distribution, which may differ from net
 investment income under generally accepted accounting principles.

Figure 26: Example of a multiple dividend event

We have an extract of a news announcing two dividends, the main one \$0.39, and a special dividend of \$0.10. The model returned for the amount 0.49. According to the extract, this amount corresponds to the "total" of the main and the special dividend. It is, then, not one of the two amounts we are interested in. However, we are already quite satisfied by this extraction, because the model extracted something quite relevant, and for example did not extract 0.48 which was the "total" dividend for the last period. In passing, a good way to use the weak supervision setting to avoid extracting old dividends is to write heuristic labeling functions which detect signs of the past like "was" or "2018".

Until so far, we dealt with each candidate-amount in the news as it is a different candidate, even if the amounts are the same. It is important to build entity like I did but also to avoid situations for which two amounts are the same within a news, but are not related (it does not correspond do the same dividend, or it is not even related to a

dividend). To tackle this multiple dividends event, I now consider all the similar amounts as co-reference of the same entity.

If at any location of an amount of money has the same value that the one extracted previously by the model (0.49) there is an evidence of multiple dividend event (for example the word "special") then the multiple dividend post processing is triggered.

Declared Dividend of \$0.49 Per Share for Quarter Ended June 30, 2019 June 30, 2019
 Dividend Income: \$7.2 million, or \$0.42 per weighted average diluted share
 Dividends Paid \$0.38 per share in a Regular Dividend, plus \$0.10 per share in a
 Supplemental Dividend Total Dividends for the quarter of \$0.48 per share Net
 Realized and Unrealized Portfolio Appreciation: \$3.6 million, or \$0.21 per
 weighted average diluted share Net Asset Value ("NAV") per Share: \$18.62 Pre-
 Net Investment Income: \$24.8 million for the fiscal year, or \$1.48 per weighted
 average diluted share versus \$1.02 per weighted average diluted share in the
 previous fiscal year, representing 45% growth Dividends: Declared and Paid
 Total Dividends of \$2.27 per share versus \$0.99 in the prior fiscal year,
 representing 129% growth Supplemental Dividend Program: During Fiscal Year
 2019, announced dividend of \$0.10 per share to be paid on a quarterly basis out of
 Undistributed Taxable Income ("UTI"), generated primarily through realized capital
 gains and Pre- Net Investment Income in excess of Regular Dividends UTI Balance
 at the end of the fiscal year ended March 31, 2019 was \$1.14 per share In
 commenting on the Company's results, Bowen Diehl, President and Chief Executive
 Officer, stated, "Fiscal year repurchased a total of 46,363 shares at an average
 price of \$16.67 per share, including commissions paid. The Company currently has
 approximately \$9.2 million available for additional repurchases under the
 program. On May 30, 2019, the board declared a total dividend of \$0.49 per share
 for the quarter ended June 30, 2019, comprised of a regular dividend of \$0.39
 per share and a supplemental dividend of \$0.10 per share. The Company's dividend
 will be payable as follows: when declaring dividends, the Board reviews estimates
 of taxable income available for distribution, which may differ from net
 investment income under generally accepted accounting principles.

Figure 27: Detection of the multiple dividend event

In this example, "supplemental" has been detected. The model then retrieves all the amounts which are in the neighborhood of \$0.49. With the multiple dividend event detection, the model can not just output \$0.49 but, nevertheless, we trust the model and we consider that the area around \$0.49 is an area of interest, and that the dividends that we want to extract are close.

Declared Dividend of \$0.49 Per Share for Quarter Ended June 30, 2019 June 30, 2019
 Dividend Income: \$7.7 million, or \$0.42 per weighted average diluted share.
 Dividends: Paid \$0.38 per share in a Regular Dividend, plus \$0.10 per share in a
 Supplemental Dividend. Total Dividends for the quarter of \$0.48 per share. Net
 Realized and Unrealized Portfolio Appreciation: \$3.6 million, or \$0.21 per
 weighted average diluted share. Net Asset Value ("NAV") per Share: \$18.62. Pre-
 Net Investment Income: \$24.8 million for the fiscal year, or \$1.46 per weighted
 average diluted share versus \$1.02 per weighted average diluted share in the
 previous fiscal year, representing 45% growth. Dividends: Declared and Paid
 Total Dividends of \$2.27 per share versus \$0.99 in the prior fiscal year,
 representing 129% growth. Supplemental Dividend Program: During Fiscal Year
 2019, announced dividend of \$0.10 per share to be paid on a quarterly basis out of
 Undistributed Taxable Income ("UTI"), generated primarily through realized capital
 gains and Pre-Net Investment Income in excess of Regular Dividends. UTI Balance
 at the end of the fiscal year ended March 31, 2019 was \$1.14 per share. In
 commenting on the Company's results, Bowen Diehl, President and Chief Executive
 Officer, stated, "Fiscal year repurchased a total of 46,363 shares at an average
 price of \$16.67 per share, including commissions paid. The Company currently has
 approximately \$9.2 million available for additional repurchases under the
 program. On May 30, 2019, the Board declared a total dividend of \$0.49 per share
 for the quarter ended June 30, 2019, comprised of a regular dividend of \$0.39
 per share and a supplemental dividend of \$0.10 per share. The Company's dividend
 will be payable as follows: When declaring dividends, the Board reviews estimates
 of taxable income available for distribution, which may differ from net
 investment income under generally accepted accounting principles.

Figure 28: Candidates of interests

The model then computes a punctuation-aware distance based on the number of words separating two entities. By punctuation aware, I mean that I give a penalty to the distance according to the punctuation signs between the two entities. It is important because, for example, if a sentence is: "the company has declared a quarterly cash dividend of \$0.2, and a special dividend that will also be paid for the next period of \$0.1", we want to associate "special" with \$0.1 and so we need to take into account the coma.

Using this distance, we can compute the amount which is the most likely to be the special dividend, the one which is the most likely to be the total dividend (if present) and deduce from that, using lot of rules, the amount for the "main" dividend as well. After this stage, the two entities can be build around the two amounts of interest.

The multiple dividend event is a quite hard problem, but based on the approach of the augmented candidates, and using a lot of post-processing rules, we can expect to find an appropriate solution.

2.5 Details on the infrastructure

2.5.1 Localisation and handlers

The localisation is the property of the annotated documents to display the location of the different labels: for one training news, I have the true labels which have to be extracted but also their location. Ideally, in a final model ready for the production, the news should be extracted by displaying the location of the different labels. My infrastructure does not make use at all of any kind of localisation information, which means that it is easier to provide training data point. The localisation is actually gained during the extractions, for the amounts and the dates, the span, that is to say the coordinate in term of number of characters of the entity is kept and stored inside the object-candidates.

In addition, to make the extraction worked I had to implement a lot of handlers in order to be able to extract sentences, compartments of punctuation and so on. There are a lot of situations for which a dot is not a punctuation sign, but, for example, is in a number or an acronym. The development of the infrastructure necessarily requires the development of these handlers and it represents an important part of the work.

2.5.2 Implementations of the heuristic labeling functions

In term of the infrastructure, I developed a way to write an heuristic labeling function using only one line of code.

The author can specify a method such as looking into the words after the candidates, the words before and choosing a size for the window. In addition, it can be asked to consider the words in the sentence of the candidates, or the words within the compartment of punctuation. The user can also write functions at document level by specifying it. Similarly, the polarity can be chosen, and adjust very easily, and a condition to check for certain type of candidates is also implemented. This convenient implementation relies on the handlers explained previously. These labeling functions can also be based on "patterns" which are built on different simple Boolean algebra relation. For instance, if you take the first labeling function of the example: if there is "million" OR "millions" in the three words after the candidate then the heuristic "vote" for "wrong candidate".

```
9 def LF_0(candidate, text):
10     return LF("words after", [{"million"}, ["millions"]], (2, 0), nbr_words = 3).label(candidate, text)
11
12 def LF_1(candidate, text):
13     return LF("words after", [{"trillion"}, ["trillions"]], (2, 0), nbr_words = 3).label(candidate, text)
14
15 def LF_2(candidate, text):
16     return LF("words after", [{"loss"}], (2, 0), nbr_words = 3).label(candidate, text)
17
18 def LF_3(candidate, text):
19     return LF("words after", [{"dividends"}], (2, 0), nbr_words = 3).label(candidate, text)
20
21 def LF_4(candidate, text):
22     return LF("words after", [{"value"}], (2, 0), nbr_words = 3).label(candidate, text)
23
24 def LF_5(candidate, text):
25     return LF("words after", [{"earnings"}], (2, 0), nbr_words = 3).label(candidate, text)
26
27 def LF_6(candidate, text):
28     return LF("words after", [{"gaap"}], (2, 0), nbr_words = 3).label(candidate, text)
29
30 def LF_7(candidate, text):
31     return LF("words after", [{"adjustments"}], (2, 0), nbr_words = 3).label(candidate, text)
32
```

Figure 29: 8 labeling functions

Note: the polarity 2 actually refers to the -1 in my thesis report

If instead of this pattern, I would have written [{"million", "per"}, ["millions"]], the test would have been, if there are "million" AND "per", OR "millions" in the three words after the candidate, do not extract this one.

At this stage, it is clear that there is a lot of parameters to choose for each labeling function. In addition to all that I have already said, more complex signals can actually be detected. Imagine that you want to teach to your generative model to do not extract the dividend related to a former period, which could be recalled in the news. You can then implement 3 heuristics which detect if "was", "were" and for example "2018" are in the sentence of the candidate. But you can also choose to strengthen the signal and only write 2 labeling functions which detect "was" OR "were" for the first one, and "2018" for the second one. What about putting the 3 key words in the same heuristic labeling function? For which Boolean relation ? There are really a lot of choices to be made.

Moreover, all the objects used are consistent and, as a consequence, the heuristic labeling functions can be written the same way for both generative models. "Candidate" can either refers to the candidate formed around the amount, or the dates.

There are two main interests of this infrastructure. Firstly, it is very quick to develop rapidly a lot of heuristic labeling functions. The second advantage is that, by using only

one line of code to specify everything, we have an overview of all the heuristics implemented and, then, to adjust and improve the model becomes easier.

When I started to develop my heuristics, I based these functions directly on the conclusions I could draw from my observations. I read a lot of news from my training data set, and tried to find good signals for the correct, but also the wrong candidates. Based on my understanding of these signals, I chose a polarity, like discussed previously and also the scale: is this signal relevant for 3 words of punctuation, the compartment of punctuation or the whole sentence? There are then a lot of choice to be made and it is not always easy to be sure. That is why I had to try different possibilities. I observed a certain form of sensitivity to small changes.

I significantly improved my generative model by implementing a generator of heuristics. This generator is based on the estimation of $p(\text{candidate is correct} | \text{word} \in \text{context})$ and $p(\text{candidate is wrong} | \text{word} \in \text{context})$. A context can either be a couple of words after, a sentence or so on ... These two quantities provided to me a lot of precious information, and gave me lists of words which differentiate particularly well the true and the wrong candidates (2 lists for the words in the sentences, 2 lists for the 3 words after...). Using these lists, I found a lots of new signals for mainly two reasons. First, because some signals have a really low coverage, they are present in less that 20% of the news and I just did not notice them. For example, if "approximately" is in a window of 3 words after the extraction of a candidate, according to my probabilistic model there is 97% of chance that this candidate is actually wrong. It is a very high level of accuracy, and a very good signal, but which only appears in 15% of the news. The second reason is because it happens that I lack of domain knowledge. For example, my generator of heuristics pointed out that "nyse" that I did not even notice, and actually stands for New York Stock Exchange, indicates with a high accuracy that the right candidate is in this sentence. "shareseries" which often refers to another payment for the next period which is actually not the one related to the dividend that we would like to extract is a good signals for pointing out that all the candidates in this sentence are not relevant.

There is another fundamental advantage of this generator. Like I said previously,

when I want to implement a signal in an heuristic labeling function, I need to choose at what scale this signal is relevant (the sentence, a few word before, the whole news ...), the generator now does the work for me. I, thus learned that "monthly" is, obviously, really characteristic of the monthly frequency if the words appear in the sentence of the candidate, but not at all for the compartment of punctuation: it is a long-range signal.

I consider that making use of this generator of heuristics really brings something to the usual weak supervision setting. Rather than incorporating domain knowledge to build heuristics labeling functions, these rules are generated in a very accurate way, without any intervention of the user: it is a probabilistic model which builds another one. If tomorrow, I would be asked to implement an extractor of dividends for the Russian market, I may be able to implement quickly a very efficient extractor without knowing a single word in Russian.

2.5.3 Generative model

For the training of the generative model, I used Snorkel Metal [18] an application available online coded by a team of researchers from Stanford (in 2018). The training provides the weights for the different labeling functions, with respect to the probabilistic model explained part I.

The advantage of Snorkel Metal over a variant put online beforehand: Snorkel [19] by the same team, is that we can just provide the labeling matrix build freely, by our-self, to the generative model. With Snorkel, you have to use the Snorkel tools to implement your labeling functions, and they do not match with my personal user-case.

2.5.4 Time optimisation

My own code represents around 7000 lines of code. One of the goal of the extraction is to be quick. This requirements is really consistent with the use of a generative model based on heuristics labeling functions, because there is significantly less work to do than for a typical machine learning approach which usually requires, in natural language processing, to use a word embedding, the most famous is word2vec, which is the conversion of words

to vectors, in order to capture their meanings [20].

In order to optimize my code, and making the extractions faster, I used a couple of pre-processing rules. It reduced the number of "augmented" candidates. Some are based on the information contained in the title, and some others on real evidence close to the candidate extracted. For example, if "quarterly" is written just before an amount there is no need to generate all the combinations of frequencies, it has to be a quarterly dividend. That is to say, I do a clear distinction between a real evidence that should be implemented as a rule (pre or post processing), and what I call a signal, which just increases the likelihood of a candidate, and has to be implemented as an heuristic labeling function.

I noticed during my experimentation that this selection of the candidates as a pre-processing stage has to be kept only for the extractions (not for the training!). To allow the generative model to learn to differentiate the correct candidates to the wrong ones, we indeed have to provide enough candidates for both cases. A very efficient pre-processing stage before the training could significantly decrease the number of wrong candidates, and the model could misjudge some signals. Any user of weak supervision should keep in mind, at all time, that the real size of the training data set is not the number of news, but the number of candidates.

2.6 Experimental results

2.6.1 Results at document level

For my experimentation, I used 198 documents never seen before by the model, or even by myself, and which contain all the fields (no missing value). The first interesting property of the model is that, by extracting a whole entity rather than just the labels, we gain a kind of consistency between the labels that the model outputs. As a consequence, I observed that the success of the extractions of the labels are correlated with each other.

As an illustration, here is the heat map of the correlation matrix:

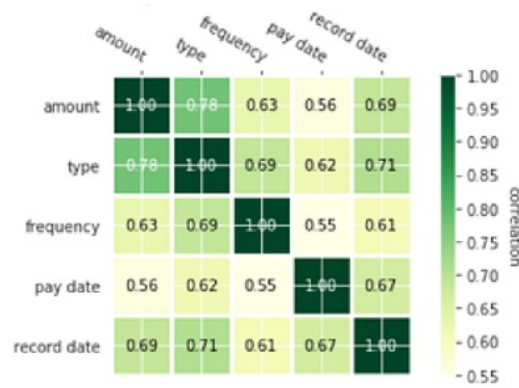


Figure 30: Heat map of the correlation matrix of the truthfulness of the labels extracted

These correlations improve the performances of the model. Thus, over the 198 documents, 197 were returned with all the fields without any missing values, and 193 were completely correct. The global success rate at document level is then: 97.5%.

Another advantage of the model I used, is that the extraction relies on the heuristic labeling functions, and by optimizing my infrastructure, I reached 0.45 seconds by document, which is significantly lower than the 30 seconds required.

To sum up, the results, at document level, for this data set are:

Performances at document level	
precision:	98.0%
recall:	99.4%
F1 score:	98.7%
global success rate:	97.5%
Average time per extraction:	0.45 seconds

2.6.2 Results at label level

The label by label performances reach the specifications required for the model.

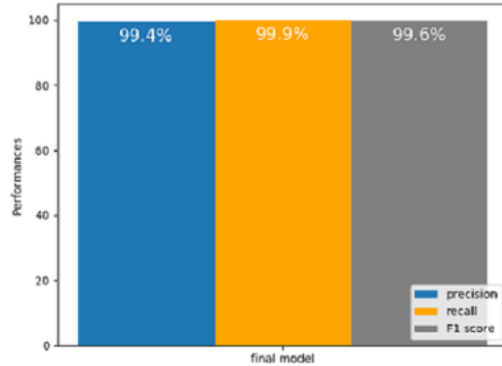


Figure 31: Final performances of the model at label level (on average)

The performances are at the same level for all the labels, and they are very high for all the performances metrics. The extractions for this data set were definitely a success.

The following table details the performances label by label:

Label by label performances			
Label	Precision	Recall	F1 score
Amount	99.5%	100%	99.7%
Type	99.0%	100%	99.5%
Frequency	100%	99.5%	99.7%
Date of payment	99.5%	100%	99.7%
Date of record	99.0%	100%	99.5%

There is still a margin of improvement for the results. The errors for this data set comes from a wrong multiple dividends event detection for one news, and a wrong logic for the partnership distribution type. As for the dates, there are usually easier to extract, but I did not have the time yet to fully take advantage of the generator of heuristics. Despite these errors, we can see that the model largely reaches the expected performances.

2.6.3 Review of the performances

One could, quite naturally, wonder what is really the role played by the weak supervision setting in the results that I got. In order to provide some answers to this legitimate question, let's assess the role played by the pure weak supervision setting. Even if the generative model works well, the rules that I wrote and which are applied at different stages of the data pipeline really improved the performances

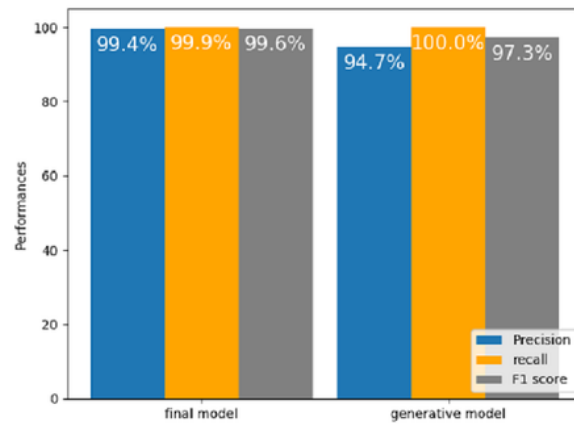


Figure 32: Comparison of the performances of the whole data pipeline, and the generative model alone (for the amount-candidate)

We have seen part I, that a generative model relies on different level of modeling. Thus, one can choose to model the accuracy and the correlations of the heuristic labeling functions, or not. the next chart illustrates the differences in term of precision.

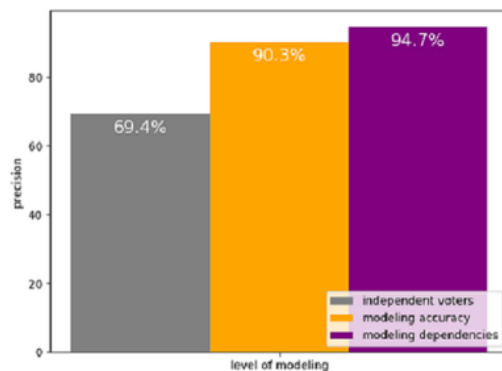


Figure 33: Comparison of the precision according to the modeling

We can see that the main progress of the generative model comes from the modeling of the accuracy of the heuristic labeling functions. When the generative model also learns the dependency structures of the heuristics, we can observe a slight, but valuable improvement.

With regard to the first part of the thesis, one can also wonder what are the performances, in the theoretic case for which we do not have any annotated data at all. What I observed is that there is not any change. The approach developed part I makes possible to retrieve the dependency structure without ground truth. The only advantage of having annotated data is to make it easier, and quicker to find good signals, in order to write heuristic labeling functions.

The two last charts show how the rules and the heuristic labeling functions are complementary for my user-case. To locate the entities which should be extracted by the model, the weak supervision works really well, because we can use signals which increase or reduce the probability of a given candidate to be the correct one. However, it happens that with a lot of signals the generative model gets somehow confused. In that situation, a model which can also relies on very accurate rules in order to detect, this time, evidence (and not signals) of a given label can be used, and correct the potential misjudgments of the generative model.

Let's now have a look to the generator of heuristics. To use it, we definitely need to use the annotated documents.

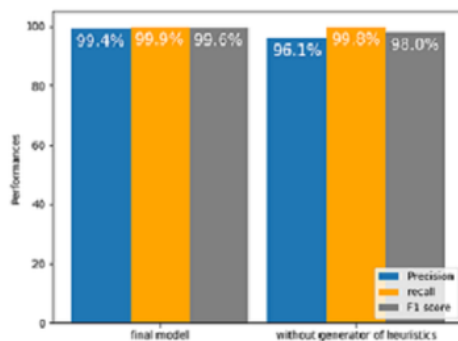


Figure 34: Assessment of the benefit of the generator of heuristics

It is clear that the generator helped me to find more complex dependencies than what I initially incorporated in the generative model. Based on these new signals, the number of heuristics labeling functions have increased from 65 to 187.

2.6.4 Assessment of the results

The model developed works then particularly well for both the performance metrics and the speed of extraction. In term of maintenance, it presents a real advantage over a pure machine learning approach. Let's imagine that this model is used in production and in a couple of months, or years, companies start using new ways of announcing the dividend within the news. To retrain a pure machine learning model, you need to choose and annotate a lot of relevant documents. In addition, you can not foretell if it will keep the same level of performances. You need then a lot of efforts, and it will be costly. It is not the case with the data pipeline I developed. What is really needed to do within the weak supervision setting? You need someone with a basic understanding of the approach, and who can implement a small number of new heuristic labeling functions. Nothing more. The business value behind this implication seems tremendous to me.

In addition, using the generator of heuristics, one can build a generative model without

any form of domain knowledge. It is then possible to build rapidly, without any cost a model to extract the news from the Russian market, for example, without knowing a single word of this language. Moreover, you can expect that your system will perform really well for this new task.

There is a last point. The machine learning algorithms are often criticized for being black boxes. It is then hard to trust them for investors, especially if they do not have a good understanding of the topic. This approach relies on a generative model which is composed of heuristic labeling functions written as an if-else statement, so the model is nothing more than a large set of very transparent boxes and, as a consequence, it is really easier to understand the principle of the approach.

Conclusion

The weak supervision setting appears to be a really good framework to perform machine learning tasks in the absence of a large training data set. Whether it is for information extraction from documents, like in my personal user-case, or any relevant assignment of the traditional fully supervised learning, the weak supervision offers a way to build rapidly a low-cost model, which can reach very high performances. Rather than annotating thousands of training examples, the users can simply write labeling functions which relies on knowledge of different nature.

There is still a gap between researchers, who work a lot on sophisticated deep learning algorithms, and the commercial applications which are often based on simpler models, and even, sometimes, on complex rules-based approaches. One of the main explanation is that the development of the state-of-the-art models requires a real investment in term of time and costs, and it is often reserved for the biggest company. I genuinely consider that the weak supervision can upturn this situation, and be an entry door to the popularization of the most sophisticated models by new actors.

With regards of the numbers of current applications, and recent improvement of models within the supervised learning thanks to the development of new models, but also the release of large data sets available online, I can not stop thinking that the weak supervision has a bright future.

In 2016, at the NIPS (worldwide famous conference in Neural Information Processing Systems), Andrew Ng proposed the following illustration in order to highlight the most crucial topics of machine learning:

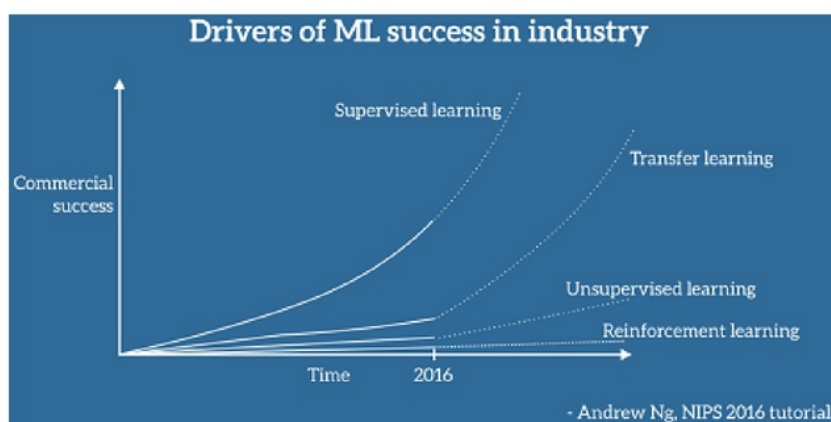


Figure 35: Drivers of ML industrial success according to Andrew Ng [23].

If the past years, and the many advances have not detracted from this forecast, since this conference, the theory of weak supervision has been developed, and now appears as a promising approach.

The question is where should be the weak supervision, today, in such graphical representation? Probably not left out ...

References

- [1] J. Devlin, M. Chang, K. Lee and K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Academic Press, 2018.
- [2] A. Ratner, P. Varma, B. Hancock, C. Ré, and other members of Hazy Lab. *Weak Supervision: A New Programming Paradigm for Machine Learning*. The Stanford AI Lab Blog, March 2019.
- [3] Z. Zhou. *A brief introduction to Weakly Supervised Learning*. National Science Review, Volume 5, Issue 1, January 2018, Pages 44–53
- [4] S. bach, B. He, A. Ratner, and C. Re. *Learning the structure of generative models without labeled data*. International Conference on Machine Learning (ICML), 2017.
- [5] A. Ratner, S. bach, H. Ehrenberg, J. fries, S. Wu and C. Re. *Snorkel: Rapid Training Data Creation with Weak Supervision*.
- [6] P. Varma, F. Sala, A. He, A. Ratner and C. Re. *Learning Dependency Structures for Weak Supervision Models*.
- [7] A. Ratner, c. Sa, S. Wu, D. Selsam, and C. Ré. *Data Programming: Creating Large Training Sets, Quickly*. Proceedings of the 29th Conference on Neural Information Processing systems (NIPS 2019). 2016
- [8] P. loh and M. Wainwright. *Structure estimation for discrete graphical models: generalized covariancse matrices and their inverses*. Annals of Statistics, 41(6):3022–3049, 2013.
- [9] V. Kuleshov and S. Ermon. *Markov random fields*. Courses notes (CS228) of Standford.
- [10] Lauritzen, Steffen. *Graphical models*. Oxford: Clarendon Press. p. 33. 1996.
- [11] *wikipedia page: Markov random field*. Wikipedia

-
- [12] E. Yang, Y. Heights, P. Ravikumar, G. Allen and Z.Liu *Graphical Models via Univariate Exponential Family Distributions*. Journal of Machine Learning Research 16 (2015) 3813-3847
- [13] M. Basaldella, E. Antollim, G. Serra and C. Tasso *Bidirectional LSTM Recurrent Neural Network for Keyphrase Extraction*. IRCDL, 2018.
- [14] A. Ng and K. Katanforoosh *Deep Learning*. Notes Stanford Course CS230, section 7 2018.
- [15] W. Wieringen *The Generalized Ridge Estimator of the Inverse Covariance Matrix*. Journal of Computational and Graphical Statistics, 2019
- [16] G. Hinton *Training products of experts by minimizing contrastive divergence*. Neural computation: 1771-1800. 2002.
- [17] E. Candes, X. Li, Y. Ma and J. Wright *Robust principal component analysis?*. Journal of the ACM, 58. 2011.
- [18] *Snorkel Metal*. <https://github.com/HazyResearch/metal>, github repository.
- [19] *Snorkel*. <https://github.com/snorkel-team/snorkel>, github repository.
- [20] D. Jurafsky and J. Martin *vector semantics* Note of course: Speech and Language Processing, chapter 6. Stanford. 2018.
- [21] O. Raviv *Handling Imperfectly Labeled Data* Conference, data Science summit (Tel Aviv). 2018.
- [22] R. Grosse *Training a Classifier* Note Toronto university course, CSC321: Intro to Neural Networks and Machine Learning. 2018
- [23] A. Ng *Illustration* NIPS 2016.
- [24] D. Iter, A. Ratner, and C. Ré *Illustration, Data Programming in TensorFlow* Snorkel blog: <https://www.snorkel.org/>.

-
- [25] R. Bunescu and R. Mooney *Learning to extract relations from the Web using minimal supervision* Meeting of the Association for Computational linguistics (ACL), 2007.
- [26] M. Mintz, S. Bills, R. Snow, and D. Jurafsky *Distant supervision for relation extraction without labeled data* Meeting of the Association for Computational linguistics (ACL), 2009.
- [27] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi *Aggregating crowdsourced binary ratings* In Proceedings of WWW. ACM, 285–294.
- [28] H. Abdou and J. Pointon *Credit scoring, statistical techniques and evaluation criteria: A review of the literature* USIR (digital collection of the University of Salford), 2011.

REMI_MOUZAYEK_01601868

GRADEMARK REPORT

FINAL GRADE

/0

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

PAGE 64

PAGE 65

PAGE 66

PAGE 67

PAGE 68

PAGE 69

PAGE 70

PAGE 71

PAGE 72

PAGE 73
