Pioneering Transparent ESG Analysis on Public Markets: Leveraging Machine Learning to Transform ESG Data

Remy Deshayes
ML Lead
Integrum ESG

Tim Murnaghan
CTO
Integrum ESG

Abstract

Sifting through millions of pages from thousands of company documents to extract and meaningfully compare Environmental, Social, and Governance data is an exceptionally challenging task for professional investors. In this short paper, in a practitioner-talk inspired fashion, we share some of the Machine Learning journey Integrum ESG undertook to solve this challenge and how a blend of human and artificial intelligence allowed us to become the only ESG risk rating provider to deliver completely transparent and always up to date ESG risk analysis and Impact data.

Correspondence: remy.deshayes@integrumesg.com

1 Introduction

The current standard for companies publishing ESG information is for them to include it in their annual sustainability reports. In the absence of mandated or agreed reporting standards these are largely in unstructured text, usually a PDF document. If reading thousands of company reports is not something humans can nor want to do, it still is a tractable Machine Learning (ML) problem when using modern technology.

As they have been guiding our ML journey, it is important to understand the principles at the core of Integrum ESG's risk ratings. For every metric, our aim is to provide a "glass-box" i.e. to provide our clients with all the underlying data motivating the risk analysis score as well as the companies relative performance compared to their peers. To translate these principles into an ML development roadmap, we have focused our work on models' explainability and robustness using innovative Human-in-the-loop and data-centric AI methodologies.

To ensure the best possible data quality for our clients, the Integrum ESG research analysts team review every relevant data point before it is sent out to our proprietary dashboard. This process had a two-fold impact on the ML work. First, we wanted to integrate the analysts' review in the modelling process. Essentially, we wanted to make sure our model could automatically refine its prediction based on the analyst-provided feedback, just like a Data Engine. The second and most crucial impact was that we needed to develop a model that was capturing most, if not all the relevant ESG data – potentially at the cost of including some irrelevant text. Think about an x-ray scanner at the airport, a failure to detect a firearm outweighs the inconveniences of performing secondary inspections. In other words, the cost of a type I error is negligible compared to a type II error. Our use case works in the same way: it is easier for financial analysts to discard irrelevant text than it is for them to sift through hundreds of pages to find relevant data which, in essence, is the initial challenge ML is meant to be solving. Type I and type II errors being concepts generally used in statistical hypothesis testing rather than ML, we will consider adjacent, but not identical, more traditional ML metrics, to express these concepts and evaluate our model: *recall* and *precision.*

The scope qualitatively described in this introduction is the basis for the work supporting the latest version of the ESG text relevancy classifier we introduce in this paper. The model currently deployed achieves a recall of 0.98 on average on public companies' annual reports, ensuring an efficient workflow for the analyst as well as solving the challenge which motivated the model development in the first place. The control of the precision is an ongoing area of research at Integrum ESG and is currently around 0.70.

Topics: ESG Data, Data Centric AI, Active learning, Data Engine, Word and Sentence embeddings, Recall and Precision, Surrogate Cost functions

2 Building an ESG Classifier

In this section we describe the bootstrapping journey undertook to develop an original ESG text classifier through selected ML research topics we hope the reader will find interesting. As previously described, the classifier takes as input some English text and highlights sections and sentences of the text relevant to Environmental, Social and Governance (ESG) metrics as defined by Integrum ESG in accordance with the frameworks set out by the Sustainable Accounting Standards Board (SASB).

This is an inherently supervised task, and, in the context of this paper, we will treat it as a binary matter - the text is or isn't relevant to an ESG metric. This way of understanding the problem unveils its highly imbalanced nature. Indeed, text is usually not about ESG and as such having a dummy model always predicting *non relevant to ESG* would probably be correct in most cases. This is not completely unheard of in ML - think about automatic credit card fraud detection for example, where only a very limited portion of the transactions are genuinely fraudulent – but certainly calls for adapted methodologies. Imbalance can commonly lead to uneven costs of False Negative and False Positive misclassifications, which, as described in Section 1 is the case for us.

The imbalance alongside our preference for recall over precision was the main challenge and driver for our technology choices. As such, the panorama made in this section will revolve around their impact on each building blocks of the model.

2.1 The Importance of Data

Data is the foundation of any ML system and even with the best intentions a model won't perform if the data is not appropriate for the use case - let it be issues with quality, quantity, distribution shifts, leaks, etc - as the colloquial saying goes *"garbage in, garbage out"*. Unfortunately, it is very common for ML practitioners to overlook the importance of data and to adopt a model-centric approach to building ML system when ten of the most-used ML tests sets still have pervasive label errors [1].

We want to take the opportunity of this practitioner paper to delve into our approach to getting data but also to switch paradigms from a model-centric approach to a data-centric approach and remind the reader that real-world data is usually messy and shouldn't be treated as fixed - improving models is not the only way to get better performance [2].

To build a binary classifier, we needed sentences in English labelled *Relevant* or *Not Relevant* to a specific ESG metric. We started with small hand curated datasets of a couple thousands data points so that we could train a first version of the model. Manually annotating data has its own set of challenges, especially when it has to do with detailed specialist knowledge. There are risks associated with human annotations - they can click the wrong button, have a difficult sentence to classify or even disagree with another reviewer. These inconsistencies introduce label noise which ultimately has an impact on the model's confidence in a prediction.

Potential human errors aside, building hand-curated datasets was not a sustainable way to collect a database worth of text covering relevancy content for over 30 ESG metrics from hundreds of reports – essentially this meant reading and annotating millions of pages which, again, is what ML is meant to be solving. This first set of small datasets allowed us to train simple models tuned for recall and which we put to work on new data to annotate thousands of sentences a day. At that stage, the very limited amount of data we had, translated in a bottleneck at training time which resulted in a simple logistic regression having the same performance as a fine-tuned Long Short Termed Memory (LSTM) net. Obviously, using labels generated by another model is the recipe to reinforce bias and errors of the initial weak models. This meant we needed a systematic way to ensure that every label generated was checked.

We broke down the work in two streams, one for each of label. First, we started working on the *Relevant* labels as our initial hand-curated datasets had very little relevant ESG content. Indeed, the human annotation process included some specialist vocabulary carefully selected but most of the annotation had been done through the traditional work of an ESG financial analyst i.e. reading a report and looking for relevant ESG content. Following our principle of reviewing every data point before it's sent out to the dashboard, everything labelled *Relevant* by the first version of the model was stored in descending order of prediction confidence for secondary inspection by an analyst. In their first instance, the models being tuned for recall, they initially returned most sections of the reports which was too much for a secondary inspection. To bring the number down, we implemented a Regular Expression filtering process which boiled down to automatically excluding from the secondary review any sentence labelled *Relevant* with no ESG keywords from a curated list. This allowed us to carry an intensive but tractable secondary review on a reduced set. The result of each review was automatically fed back to the model for regular retraining, effectively creating a Human-in-the-loop process. Ultimately this allowed to grow a controlled database of relevant ESG content and to get the average recall above 0.80 for all but five metrics with a tolerable precision just above 0.5.

Confident Learning. However, this didn't solve the fact that a growing collection of text labelled *Not Relevant* was accumulating with no safeguards. It meant two things, (1) some data points that were initially labelled as *Not Relevant* by early versions of the model could have been relevant, but also (2) that all the content that had been excluded from the secondary review was slowly creating a latent bias by teaching the model to look for keywords from the curated list, effectively creating a clever word matcher. The models started very disjointed prediction sets with either very high or very low relevancy probabilities. This eventually resulted in a weakened Human-in-the-Loop process as less and less useful to disambiguate uncertain predictions were surfacing.

To tackle the labelling checks of terabytes' worth of irrelevant text, we started reflecting on what ultimately can be the cause of the model's uncertainty on a given data point. Generally, there are two main recognized sources for a model's lack of confidence on a prediction. One is the model's inability to make sense of a data point, and the other is linked to label noise introducing uncertainty in the prediction.

To disambiguate uncertainty sources, we need to make assumptions on label noise for which we give a short theoretical explanation inspired from [3]. Let's start by introducing two useful notations: $\hat{y}$ is the observed, potentially noisy, label and $y^{\star}$ is the unobserved ground truth. We can write the probability for a certain model θ to predict i given a data point x as follows: $\hat{p}(\tilde{y} = i|x, \theta)$. This probability expresses both types of uncertainty as it depends both on the model and the data. To disambiguate them, the probability cannot depend on the data x.

The simplest possible systematic assumption we could make on label noises is that they are uniform [4] i.e. observing label i when it was in fact label j is just as frequent as swapping label l to label j for example. Which essentially mean that swapping the label *cow* to *bull* is just as frequent as swapping the label *plane* to *bull* – not applicable to a real work scenario.

Another systematic assumption is to assume a class conditional label noise [3] which, using previous notations, simply means $p(\tilde{y}|y^{*}, x) = p(\tilde{y}|y^{*})$. The label noise is not instance dependent and depends on the class which is a reasonable assumption to make in our case. Indeed, regardless of the specific text, in general something labelled *relevant to GHG emissions* has a higher probability to be swapped to a label *relevant to climate stability* than to be swapped to a label *relevant to labour practices.*

So why are we interested in this assumption? Confident Learning (CL) is built on the Class conditional assumption. We recommend reading [3] to get a full grasp of CL but, essentially CL focuses on systematically estimating label quality based on estimates of the joint distribution of observed labels and ground truth labels $p(\tilde{y}|y^*)$ i.e label noise, using out-of-sample predicted probabilities $\hat{p}(\tilde{y} = i|x, \theta)$ and the associated vector of observed labels $\tilde{y} =$. Building on an unnormalized estimate of the joint distribution matrix for $p(\tilde{y}|y^*)$, we pruned our entire database of text (both *Relevant* and Not Relevant) by probability ranking - which is robust to class imbalance [3]. Using CL allowed the discovery of interesting mislabelling patterns such as very similar fragment of texts labelled both as *Relevant* and *Not Relevant* because of a single word failing the Regular Expression matching.

CL had a very noticeable impact on the models' predictions. Following retraining on the improved data, precision went up by 15% on average with a slightly increased recall (c. 3%). As mentioned before the models had shifted to very confident probabilities for most predictions - usually above 85% or under 5% - which stopped after retraining on the new datasets and allowed the effective Human-in-the-loop process to resume. This is arguably the most interesting impact CL had.

2.2 Text Representation

To get processed by a model, text needs to be converted into numerical vectors while preserving the initial meaning of the sentence. Finding the appropriate text representation for our use case has been pivotal in our ML journey.

Just like for the datasets, choosing an appropriate text representation was heavily influenced by our preference for recall under our imbalance scenario. Early versions of the datasets being keyword driven, as described in Section 2.1, we turned our attention to retrieval technologies. Term Frequency – Inverse Document Frequency (TF-IDF), perhaps surprisingly given its relative simplicity, has proved itself to be particularly robust and well-suited to our use-case for two reasons. First, TF-IDF simply assigns a weight to a word based on a product between its frequency in a document and its rarity across the entire corpus. In our case, terms associated with the *Relevant to ESG* text fragments occur infrequently across the entire collection of company reports which make them stand-out. The second argument for using TF-IDF was linked to its inherent sparsity which makes resulting embeddings easy to manipulate and RAM-efficient in their compressed sparse row matrix form [5].

However, in conjunction with the Regular Expression filter we presented in Section 2.1 - which we replaced with Confident Learning, TF-IDF representations became heavily reliant on some of the ESG-related keywords which only reinforced their inability to capture semantic relationships. A sentence like *The creative and collaborative environment of the Lab inspired many innovative ideas among the PhD students* and *we are pleased to provide an in-depth ESG disclosure that encompasses our efforts to mitigate operational impact on the environment* would get very similar representation.

To solve these issues, we initially decided to move away from TF-IDF completely. We tried various more complex technologies such as shallow neural nets (Word2Vec, etc.) which we appreciated for their ability to consider a sliding window of context words as they iterate over the entire report but also for the possibility to fine-tune them on an ESG specialised vocabulary - which unsettled most raw off-the shelves options. Unfortunately, this didn't produce the results we were hoping for as it simply swapped 10% in recall for 10% in precision.

We were initially reluctant to continue using retrieval technology through a 2-stage model (i.e. one model, using TF-IDF, would focus on recall and generate a sparse set of candidate proposals for a second more subtle model designed to get a higher precision) as, when using messy real-world data, this tends to simply propagate errors [3]. However, as we were getting more confident about the quality of our datasets thanks to the Confident Learning work undertook, we felt that we could reintroduce some retrieval technology through a voting scheme. A voting scheme is the combination of the predictions of multiple models, usually using a derived version of majority voting. However, in imbalanced scenarios, like ours, where models might be tuned for a specific evaluation metric, raw predictions from individual models usually don't accurately represent the genuine likelihood of a label. For instance, a model tuned for recall will output higher probabilities on average than a model tuned for precision which means that a simple majority vote would simply propagate these inaccuracies. To correct the skewed probability distributions, we calibrated the output of each individual model in the voting scheme using an isotonic regression to bring their predictions in line with a more accurate depiction of the relevancy of each text fragments. Using the voting scheme, our experiments with more complex text representation techniques gave a noticeable increase in precision while holding the recall score. Coupled with the retrieval abilities of TF-IDF, we noticed that BiLSTM architectures designed for translation [6] helped in getting precision from 0.575 on average to 0.65 while getting recall above the 0.90 mark.

Models designed for translation can capture nuanced semantic relationships as they have been trained on a translation task using a parallel corpus but will fail at holding recall, which is our argument for designing a voting scheme. This typically translate in a greater ability to use the context of a sentence to make a prediction and as such to rely less heavily on keywords and avoid simple mistakes such as representing, *"The creative and collaborative environment of the Lab inspired many innovative ideas among the PhD students"* and *"we are pleased to provide an in-depth ESG disclosure that encompasses our efforts to mitigate operational impact on the environment"* in similar ways. There is no denying, that more context awareness is the next step for our models to progress further and get a boost in precision. However, we were surprised that more advanced technologies would still need the support of a 50-year-old technology to fulfil our use-case even on clean and controlled text fragments. We argue that one of the reasons for this is the size of the vectors used to represent text fragments. Indeed, newer models squeeze their text representation in small, pooled vectors. For instance, Meta's LASER represents any sentence in a 1024-element vector when TF-IDF dictionaries fitted to our database usually performed best with vectors of over 10000 elements. Admittedly, TF-IDF is not particularly robust to inputs noise and pooling contributes to efficiently reducing the dimensionality of the data while giving a level of shift invariance and capturing the most relevant features within a sentence. But is that enough to account for a whole order of magnitude difference in vector size?

Pooling involves aggregation - without too much simplification this could be summarized down to *averaging* text – which means a loss of information notably in context as the aggregation is usually done on a fixed-size window, but also a loss of positional information as pooling disregard order which is particularly harmful in our opinion when considering sentence representations. Still to this date, a prevalent method for sentence representation involves calculating the dimension-wise average of word embeddings within a given sentence [7]. As this seemed conceptually wrong to us, we investigated technologies that would account for both the position and importance of a word in a sentence and its meaning in a given context. Signal Processing derived methodologies felt like they could be an appropriate solution given their ability to decompose a signal and capture the time and space properties of its component frequencies.

Signal Processing. These methodologies are very new to the wider NLP community as it is still unclear how to transform a sentence into a signal with meaningful transitional properties between words which would the first step towards using Signal Processing (SP) tools.

Building on [8]'s observation of the almost algebraic properties of word embeddings, we represented sentence as a multi-dimensional signal where each word representation is a point in the signal. To exploit the embeddings' dynamic characteristics, we explored two variations of traditional SP tools. We alternatively tried to use wavelets as there is a great range of related Python libraries [9] as well as variations of the Dynamic Mode Decomposition (DMD) algorithm which is frequently used to decompose time-evolving systems like vibrations or fluid flows in fundamental frequencies.

Based on [7]'s seminal work, we give a brief explanation on how to adapt the DMD algorithm to a text representation context but do not elaborate further as the techniques we have explored so far have not been tractable on our local server. This is still an ongoing research area at Integrum ESG.

We denote S^N a sentence composed of N words $S^N = [w_1, ..., w_N]$, where w_i is a column vector of size m where m is the embedding size – 10,000 for TF-IDF for instance.

The sentence S^N can be written as follows:

$$\begin{pmatrix} w_1^1 & w_2^1 & \dots & w_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ w_1^m & w_2^m & \dots & w_N^m \end{pmatrix}$$

Using Koopman theory, we can express our dynamical system in a linear form [10]:

$$\left[w_1, A \cdot w_1, A^2 \cdot w_1, ..., A^{\{N-1\}} \cdot w_1\right] \quad (1)$$

where A is the Koopman matrix.

In general, A is infinite-dimensional, meaning that Equation 1 is in fact an approximation. Although there are principled ways of learning such finite approximations, most available tools are based on unstructured methods [10] that are not satisfying at this stage on very large datasets.

Over the upcoming months, we will work on bridging the gap between theory and tractable methodologies to obtain sentence level embeddings with none of the pooling defects.

2.3 Training and real-world deployment

Training a classifier in a highly imbalanced scenario comes with a set of challenges especially when deployed for real-world applications. On the one hand, the model might be unable to efficiently detect the minority class as it had very little material to learn on as well as developing a bias favouring the majority class. On the other hand, the predictions' precision might gradually decrease with time after deployment if the models are not properly monitored.

Regarding the model inability to detect the minority class, there are usually two ways to tackle this problem. One focuses on the data through resampling methods and the other is focusing on the model used. We wanted to avoid using resampling as this usually can cause data quality issues, especially with specialised vocabulary data where generating a new sample is not as straightforward as it is for a regression with numerical features for instance. Under-sampling the *Not Relevant to ESG* fragments was not something we wanted to explore either, given the considerable effort we had put in building a comprehensive collection of text fragments. Finally, resampling could have caused a covariate shift which would have weakened the model's ability to generalize to a real-world environment. Covariate shift happens when, reusing Section 2.1 notations, $p(x)$ changes but $p(y|x)$ doesn't; which in other words, means that the distribution of inputs changes between train and deployment environment, but the relationship between inputs and outputs does not change. Our use-case was the perfect grounds for it, as deployed, our models are making predictions on company reports where most of the text isn't about ESG. In this subsection, we explore some of the methods we used to account for the imbalance-related training challenges without resampling the training dataset.

The first model element we focused on was our model's evaluation metric. In an imbalanced scenario, the widely used accuracy suffers from severe limitations and can lead to favouring the majority class. For instance, using accuracy to evaluate our model would result in building a dummy model always predicting *Not Relevant to ESG*. It is therefore much more valuable to evaluate the model on the minority class, but evaluating it directly against recall would obviously have a very similar effect as evaluating it against accuracy - reversed. The F_β metric is generally used to deal with such situation as it's a generalization of the precision and recall harmonic mean with a configurable parameter $\beta > 0$ to express a preference for one or the other.

$$F_\beta = \frac{(1 + \beta^2) \cdot \text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

To find the appropriate β for our model, we designed a hyperparameter-style search involving the end-users of the models, Integrum ESG's research analysts. We used β's ranging from 2 to 10 as objective metrics for an automatic hyperparameter optimization software. This resulted in 9 sets of models fine-tuned for their respective F_β's. We then ran these models on various company reports and asked the financial analysts to review the results, as they would for live data before it is published on our dashboard. The model associated with $\beta = 8$ was the one which simplified their workflow the most.

Focal Loss. Just like we did for the evaluation metric, we wanted to explore the ability of loss functions to handle imbalance and express our preference for recall. Indeed, commonly used loss functions are disconnected from the performance metrics chosen by the ML practitioner, leading to classifiers being trained to maximize classification accuracy with the anticipation that this will be adequate to produce the desired results for the actual parameter of interest, the evaluation metric [11]. This was the case for our models as they were initially trained with a Binary Cross Entropy (BCE) loss - a traditional information theory tool which evaluate how accurate the model is when making a prediction. It would be desirable to use the F_β metric as a loss function but its non-differentiability as well as the scalability limitations of existing approaches for optimizing potentially differentiable close alternatives usually make this impossible in practice.

Computer Vision (CV) has been an interesting source of inspiration for NLP research over the past decade. Object detection is one of the most common CV tasks, it's usually highly imbalanced - think about a surveillance system in an airport looking for people on the Interpol most wanted list, most passengers aren't criminals. The traditional approach to constructing a model for this task involves building a 2-stage detector, a first model generates a sparse set of candidate proposals for a second more subtle model. This all feels very close to what we did in Section 2.2.

Recently, CV researchers trying to shift the paradigm from a 2-stage detector to a single stage detector obtained benchmark-leading results designing the Focal Loss [12]. The underlying principle is straightforward, the loss builds on the BCE and focuses training on a small subset of challenging data points while disregarding most of the easier examples from the majority class.

Below we briefly introduce how the Focal Loss differs from the BCE.

In its simplest form, the BCE is expressed as follows:

$$\text{BCE } (p_t) = -\log(p_t)$$

where p_t is the model predicted probability when $y^* = 1$ using the same notations as in Section 2.1 and $1 - p$ otherwise. The Focal Loss extends the BCE concepts by adding a modulating factor $(1 - p_t)^\gamma$ which dynamically adjusts the loss contribution based on the predicted probability, effectively giving more attention to challenging examples (low p_t) and reducing the impact of well-classified examples (high p_t):

$$\text{Focal Loss } (p_t) = -(1 - p_t)^\gamma \cdot \log(p_t)$$

Adapting the Focal Loss to our ESG classifier got us a sensible increase in recall and precision and is what yielded our best results so far. Our average recall is now 0.98 with a precision of 0.71. It is worth noting that in [12], the use of the Focal Loss allows them to shift from a 2-stage detector to a single stage detector which was not the case for us. We hope that coupling Signal Processing as described in Section 2.2 and the Focal Loss should allow us to introduce a single step ESG classifier.

3 Conclusion

In this study, we introduced, what we believe to be the first *ESG relevancy classifier* and demonstrated the feasibility of utilizing Machine Learning to extract and meaningfully compare Environmental, Social, and Governance data from company reports. The models currently deployed on our proprietary dashboard achieve a recall of 0.98 and a precision of 0.71 on average while remaining fast and computationally efficient.

We showed how building an efficient *ESG relevancy classifier* was dependent on handling two specific challenges: the imbalanced data and the uneven costs of False Negative and False Positive. To tackle these issues, we focused on three building blocks: data, text representation and learning objective optimization.

Firstly, we shifted paradigms from a model-centric approach to a data-centric approach using Confident Learning. We demonstrated that improving models is not the only way to get better performance. Secondly, we evidenced the impressive efficacy of retrieval technology in a recall-focused situation compared to more modern technologies and introduced a voting scheme combining TF-IDF retrieval abilities with BiLSTMs' context-awareness. Finally, we addressed the usual models' inability to detect the minority class in imbalanced scenarios by bridging the disconnect between evaluation metrics and objective functions.

Bibliography

[1] C. G. Northcutt, A. Athalye, and J. Mueller, "Pervasive label errors in test sets destabilize machine learning benchmarks," 2021.

[2] C. Renggli, L. Rimanic, et al., "A data quality-driven view of mlops," 2021.

[3] C. G. Northcutt, L. Jiang, and I. L. Chuang, "Confident learning: estimating uncertainty in dataset labels," 2022.

[4] J. Goldberger, and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *Int. Conf. Learn. Representations*, 2017.

[5] "Scipy documentation, scipy.sparse.csr_matrix." [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html

[6] X. Amatriain, A. Sankar, et al., "Transformer models: an introduction and catalog," 2023.

[7] S. Kayal, and G. Tsatsaronis, "EigenSent: spectral sentence embeddings using higher-order dynamic mode decomposition," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, Florence, Italy, Jul. 2019, pp. 4536–4546, doi: 10.18653/v1/P19-1445.

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances Neural Inf. Process. Syst.*, vol. 26, 2013.

[9] A. Mahajan, s. Jat, and S. Roy, "Feature selection for short text classification using wavelet packet transform," 2015, pp. 321–326, doi: 10.18653/v1/K15-1034.

[10] P. Bevanda, S. Sosnowski, and S. Hirche, "Koopman operator dynamical models: learning, analysis and control," *Annu. Reviews Control*, vol. 52, pp. 197–212, 2021, doi: https://doi.org/10.1016/j.arcontrol.2021.09.002.

[11] E. E. Eban, M. Schain, et al., "Scalable learning of non-decomposable objectives," 2017.

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.