DTU Compute
Department of Applied Mathematics and Computer Science

Automated Classification of Data from Websites

Master Thesis

Automated Classification of Data from Websites

Master Thesis
December, 2022

By
Patrikas Balsys

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Building 324, 2800 Kgs. Lyngby Denmark
www.compute.dtu.dk
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Approval

This thesis has been prepared over six months at the Department of Applied Mathematics and Computer Science, at the Technical University of Denmark, DTU, in partial fulfilment for the degree Master of Computer Science and Engineering, MSc Eng.

Patrikas Balsys - s202774

.....

Signature

.....

Date

Abstract

This project explores the dataset provided by The Index Project [1]. The dataset contains descriptions of design projects from around the world, and each project has categories and target goals assigned to them. Multiple machine learning classifiers are constructed, and these are tested in two scenarios. First, by taking various text from the projects' websites, the training of models with differing input texts is compared. Second, an algorithm named *Iterative Reclassification* is proposed. Multiple models are trained by using the algorithm with differing parameters, and the results from training are compared.

Acknowledgements

Patrikas Balsys, MSc Computer Science and Engineering, DTU
Author

Gaurav Choudhary, Postdoc, DTU Compute
Supervisor

Nicola Dragoni, Professor in Secure Pervasive Computing, PhD, DTU Compute
Supervisor

Contents

1 Introduction

1.1 Overview

There are many various tasks we wish to automate in our lives. Some automation is rather simple (e.g. the kettle turning off when water reaches the boiling point), while other, more complex automations can provide functionality that hasn't existed before, or provide a way to do tasks that are not feasible to do by humans manually (e.g. using spreadsheets to do thousands of calculations in an instant).

Many modern companies often collect data as part of their business model. Advertisement companies, such as Google, depend on data gathered from users, which can then be used for targeted advertising [2]. Other companies gather data collected from sensors, websites, app usage statistics and various other sources, so much so that data itself has become an important commodity [3]. The analysis, aggregation and organization of such collected data is often done automatically via the use of many techniques, machine learning among them.

1.1.1 Machine Learning

One of the ways to automate tasks that has risen to prominence in recent years is Machine Learning (ML). Though the technology isn't new [4], there has been a surge of various image creation models recently due to advancements in the field. With a model like DALL-E 2, one now has the ability to enter a text prompt describing an image, and such an image will be generated automatically [5]. However, image generation is not the only use case for ML, as the technique can be applied to almost anything that can be expressed in numbers.

As is evident by the simple interfaces these models provide, many of them can be treated as ”black boxes”, only focusing on the input and output. In other words, it is not necessary to know how such models work in order to use them, just as it is not necessary to know a programming language in order to use software. However, an understanding of the underlying principles of ML can help in constructing your own models, as well as understanding existing ones.

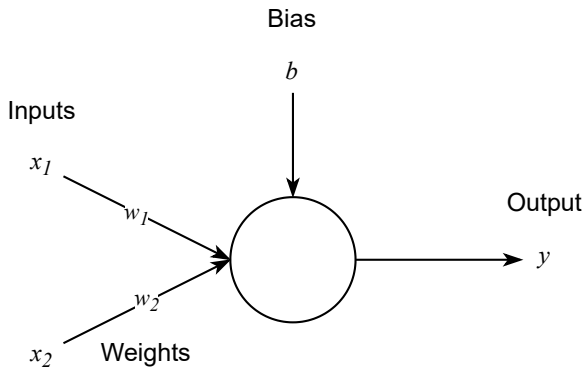


Figure 1.1: A single neuron

Neural network (NN) machine learning models are based on how the brain works; or, more specifically, how neurons work. A ”neuron” in machine learning takes in some numeric

values, multiplies each value by some weight, adds a bias, and produces an output value (Figure 1.1). Multiple neurons together can form a layer, and multiple layers together will form a deep neural network (”deep” in this case meaning ”multi-layer”).

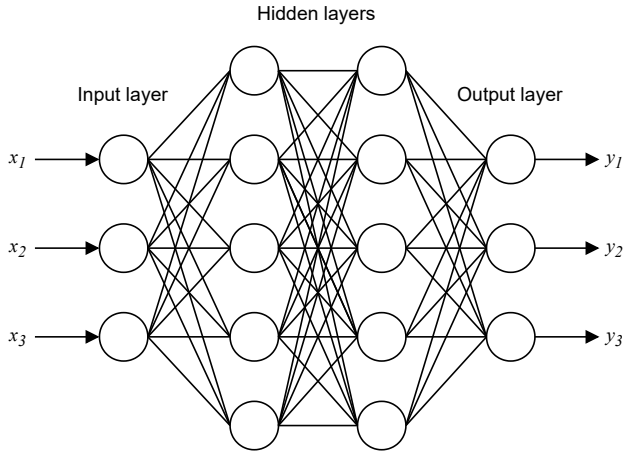


Figure 1.2: A multi-layered neural network

A network such as this can be trained on existing data. In supervised machine learning, many inputs and their expected outputs are provided, from which the training algorithm can adjust all the weights and biases in the network. Once trained, the resulting network is often referred to as a model. After training is done, the model can be used to predict new, unseen values - this is called inference.

A common use case of a model like this is classification. On a computer, almost all things can be expressed in numbers; for example, a grayscale image can be converted into numbers denoting how dark each pixel is (Figure 1.3). This can then be used as an input to the network, while the output values can signify how confident the model is that the image belongs to a specific category.

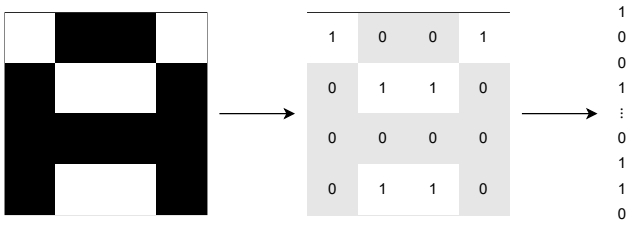


Figure 1.3: Converting an image to numbers

Neural networks are useful for tasks where simple logic is not enough. For example, finding the average of multiple numbers can be solved mathematically, and checking if someone is of legal drinking age is a simple conditional check, but checking what is depicted in an image is much more complex. ML is a technique that allows us to create a complex function without defining the exact rules within, and letting the algorithm train itself.

However, ML is not a magic bullet, as it has several drawbacks. First, the training of a network requires pre-existing data. More often than not, the quality and quantity of this

pre-existing data is the limiting factor to making a model more accurate. Secondly, ML is computationally intensive. While inference can be done on most common computer CPUs, training is intense enough that it requires one or more fast GPUs, as well as considerable time to train.

1.1.2 Natural Language Processing

A subset of machine learning that will be used in this project is called Natural Language Processing (NLP). As an image can be converted into numbers, so can text (Figure 1.4). A vocabulary is formed, and each word is converted to a number. By using text as input, we can use a model to predict what the next word will be, or what category the text belongs to. This latter use case is what this project will focus on.

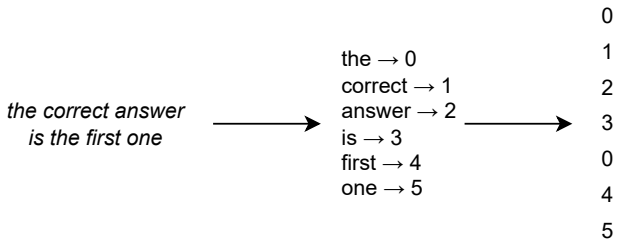


Figure 1.4: Converting text into to numbers

1.1.3 Web Scraping

If one would like to classify some text from a website, they could either copy the text manually, or utilize a tool, such as a web scraper. A web scraper is a simulated web browser with no graphical interface. Such browsers are used to interact with websites purely via code. In the case of this project, a scraper will be used to fetch any possibly relevant text from a given link. This text could then be used as input to the built model.

1.2 Problem Statement

This project is in collaboration with a company called The Index Project [1]. The company collects data on various design projects from around the world that target sustainable development goals (SDGs). In addition, the company categorises projects according to their own categories, and a few additional target goals. The current process is done by looking for web articles on news outlets, social media and other webpages, and then filling in a form on their website about each project manually, which puts the information into the company's database.

Each design project has text that describes it, and the user selects which category the project belongs to, and which goals it targets. The aim of this thesis is to create ML models that can automate this process: by using text as input, the models would predict the category and goals that a project belongs to. By utilizing these models, and with the proposition of a new algorithm, various differing conditions are tested.

1.3 Motivations and Key Contribution

The motivation of this project is to develop a system that can predict various categories and goals for a given design project. Doing this would speed up the current process of uploading information into the company's database. In the future, the model could be expanded to predict other factors; however, the current system is only focused on category and goals.

The key contribution of paper is the introduction of an algorithm, named Iterative Reclassification (IR), which is used to tackle the concept misclassified data in the existing dataset. This shall be explained more thoroughly in Section 3.2.3.

1.4 Structure

Chapter 2 shall cover some of the other research surrounding the subjects related to this project. Chapter 3 shall provide background information on the company, their dataset, techniques used, and the concept of trust in data. Chapter 4 shall cover the constructed models and the proposed algorithm, and Chapter 5 shall provide the results of the model training, as well as give any final remarks.

2 Literature Review

The project shall take use of a Python machine learning library, called Fast.ai [6]. The transfer learning techniques used for generating NLP models in the library are explained by Howard and Ruder [7]. The library utilizes some techniques on cyclical learning rates, which are based on the research of Smith and Topin [8, 9, 10]. The aforementioned techniques are used for training the model quicker and achieving a higher accuracy.

With regards to website classification, the concept has mainly been explored in two fields: search engines, and cybersecurity. Several papers cover the detection of malicious websites [11, 12, 13], with each approaching the problem in slightly different ways. Of note is the analysis of the URL itself as a parameter - various features, such as the length of the URL and the number of dots in it are used as input [11]. In addition, some more unique input examples include the info regarding the website's favicon [13], as well as the age of the domain [11]. These approaches are most likely best suited for the specific use case of detecting malicious websites, as these often have a short lifespan, and tend to be hosted on deceiving URLs. In the case of classifying website content itself, several other approaches have been made. Espinosa-Leal et al. [14] cover classifying websites into several different categories based on image snapshots. Since the approach taken in this project will be targeting classification using text, the advantage of such an approach would be the extraction of text from images - however, it was deemed that a simpler text extraction would suffice. Classification using just the text has been utilized in a few papers as well [15, 16, 17]. The research by Tsukada et al. [15] is a relatively early piece of work exploring website classification for Yahoo Japan [18], where, notably, nouns have been extracted from sentences. A similar approach to reduce text was taken by Haleem et al. [16], where ”stop words” [19] have been removed. This will not be done in this project however, as the input for transfer learning models is based on raw text. De Fausti et al. [17] offer an interesting approach, where each sentence is encoded as a grayscale image. The authors also explore an approach called False Positive Reduction to tackle the problem of low signal-to-noise ratio in web-scraped text.

The concept of data reclassification is not new either. Many papers cover datasets regarding geography: Hoyer et al. [20] cover the reclassification of volcanic activity datasets, Das et al. [21] cover the reclassification of monsoon zones in India, and Kobler et al. [22] use Kernel-based reclassification for satellite images. Other use cases include stock performance predictions [23], breast cancer risk datasets [24], and more notably for our use case, text reclassification, which was done by Rothfels and Tibshirani [25] via the use of ”seed” phrases. However, very few of the papers explore reclassification in a more iterative manner. Peng et al. [26] do use the phrase ”iterative training and reclassification cycle”, which is one of the main topics explored in this project (discussed in Section 3.2.3 and Section 4.2), but the reclassification mentioned in the paper requires constant user input.

This project also explores the concept of ”trust” in datasets. Though this term is not used in the industry as such, an adjacent topic of ”subjective” data classification yielded a few results. Kamal [27] has worked on identifying subjective text in reviews, while Mauri and Damiani [28] address the problem of model degradation. Several papers cover the detection of mislabeled data [29, 30, 31], though the datasets being worked with are not of subjective nature. This concept shall be explained more thoroughly in Section 3.2.2.

Topic	Research papers
Deep learning	[7], [8], [9], [10]
Website classification	[11], [12], [13], [14], [15], [16], [17]
Reclassification	[20], [21], [22], [23], [24], [25], [26]
Subjective data	[27], [28], [29], [30], [31]

Table 2.1: Research papers by topic

3 Background

3.1 The Company

The dataset that this project is working with belongs to a company called The Index Project [1]. Every two years, the company has an award ceremony, where various design projects from around the world are celebrated, and six are selected as winners - one from each of the five categories, and an additional one labelled ”People’s Choice”, which is selected by people outside the company, and could be from any category. The five categories include Body, Home, Work, Play & Learning, and Community.

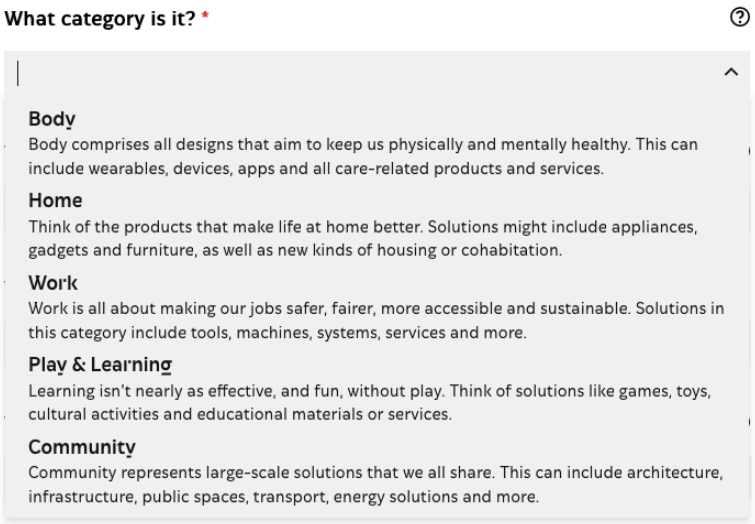


Figure 3.1: Form input on the website for selecting a category [32]

To nominate a project, anyone can visit a certain page on the website, where they have to fill in a form that describes the design project [32]. One of these fields is for selecting the category that the project belongs to. Though the categories are distinct (i.e. only one can be chosen for a given project), there is some conceptual overlap, so selecting one category can be somewhat subjective. Apart from the categories, each project also gets assigned multiple target goals. These include the Sustainable Development Goals (SDGs) [33], as well as a few goals created by The Index Project (DTIL goals) [34]. Unlike categories, multiple goals can be assigned to a single design project, up to a maximum of five.

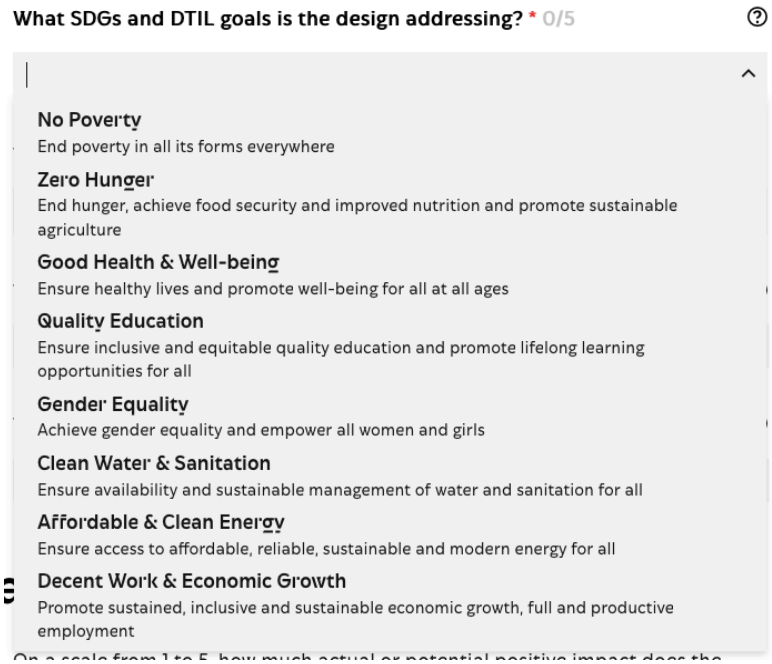


Figure 3.2: Form input on the website for selecting goals [32]

Apart from the bi-annual award ceremony, the company also hosts design awards for other companies and regions around the world. One such design competition (called *Diseño Responde*) was done in 2021 in Latin America, where the designs were targeting various issues relating to Covid-19 [35]. The text describing these projects was written in Spanish; however, much of the other information was filled in a similar manner to the regular award, meaning one category per design, and multiple targeted goals. Therefore, these projects can also be utilized.

3.2 The Dataset

The dataset used for this project includes all design projects nominated from 2014-09-09 to 2022-10-13, containing a total of 7475 projects. Each category and target goal in the dataset is numbered, categories ranging from 1 to 5, and goals ranging from 1 to 25. Before the dataset could be used, multiple alterations have been made:

1. Some of the goals belonging to the design projects were stored in a different dataset, and thus were merged into a single table.
2. Projects that are marked as ”drafts” were removed, as these can have many attributes missing.
3. Design projects allocated to award year 2023 don’t use the `long_description` attribute, and have split their description text into 3 separate parts: `dtil_description`, `how_description` and `why_description`. These descriptions were concatenated into `long_description`.
4. Websites of projects were scraped for any relevant textual data. This includes the text of the whole website, the title of the page, and the metadata.
5. Projects that belong to competitions held in Latin America were translated from Spanish to English.

6. Goals 23, 24 and 25 were removed due to their very infrequent usage, as they have been added somewhat recently. Two projects that had no target goals remaining were discarded.
7. The initial 80% of the projects were marked as the training set. The 10% after were marked as the validation set, and the remaining 10% were marked as the testing set.
8. Each project in the training set had its `trust` value calculated.
9. Many unused attributes were removed.

As such, the final version of the dataset contains the following attributes:

- `id`: unique numeric identifier for each project.
- `name`: name of the project.
- `long_description`: text describing the project.
- `page_title`: web page title scraped from the link.
- `page_body`: all text in the page scraped from the link.
- `meta_title`: meta title scraped from the link.
- `meta_description`: meta description scraped from the link.
- `category`: a number from `1` to `5` denoting the category of the project.
- `goals`: up to 5 numbers from `1` to `22` denoting the target goals of the project.
- `trust`: floating point value from `0` to `1`, denoting how trustworthy the current classification is (more information in Section 3.2.2 and Section 4.2).
- `is_valid`: `0` for projects in the training set, `1` for projects in the validation set (`2` for the testing set, though the value isn't used).

The following (Figure 3.3) is an example design project in the dataset:

name	WRISTIFY
long_description	Wristify by Embr Labs uses all-natural waves o...
page_title	Personal Cooling Device for Menopause Hot Flas...
page_body	Skip to content End of Summer Sale | Use code ...
meta_title	Personal Cooling Device for Menopause Hot Flas...
meta_description	Founded at MIT, the Embr Wave® wristband quick...
category	1
goals	3
trust	0.201939
is_valid	0

Figure 3.3: A single design project

3.2.1 Scraping

As part of the data preparation, four attributes (`page_title`, `page_body`, `meta_title` and `meta_description`) were created. These are used to store data that's been scraped from the link of each project. One of the reasons this is done is so that our model has more textual data to work with while training, but more importantly, the model has been created

with the intention that someone could just fill in the website link, and the system would scrape all of the text from the page and provide that as input to the model.

name	Oculus Rift
long_description	Virtual Reality have been talked about for yea...
page_title	Oculus VR headsets, games and equipment - Meta...
page_body	NaN
meta_title	NaN
meta_description	NaN
category	4
goals	17
trust	0.201164
is_valid	0

Figure 3.4: Example of poorly scraped data

However, the quality of the scraped data is not high; many of the projects' pages have been taken down, and as such, the textual data is empty or irrelevant (Figure 3.4). While this could degrade the accuracy of the model, it might also "build up tolerance" by noticing that those words have no relevance to the prediction result, thus creating a more consistent model in the long term. It might also be a case that the models have no concept of "too much information", so if scraped data is supplemented with a higher quality text description, the resulting accuracy will not decrease. There will be a comparison of accuracy between different types of input in Section 5.1.

3.2.2 Trust

By analyzing this dataset, a certain concept emerged, one that could be described as the "trust" of the existing classification.

There are multiple factors that could determine whether the trust in a dataset varies, or exists at all. When the company started creating this dataset, each design project only had one target goal assigned to them. This has since changed to include up to five target goals. As a consequence, many of the old projects could be assigned new, supplementary goals, and such a classification could be more accurate.

id	name	goals
1	Scanadu	3
10	Oculus Rift	17
16	WRISTIFY	3
17	Liftware	3
18	Sproutling	3
...	...	...
6961	LifeBoard - Public Rescue Board	3 12 14
6962	Wair	12
6963	NO ME CUENTES CUENTOS	4 5
6964	FiftyFifty	5
6965	Thermally stable, oral Covid-19 vaccine	3 8 10 11 9

Figure 3.5: Goals of older designs vs. newer ones

Moreover, the company has added a few extra goals since starting the dataset. Again, old projects might have fit into those goals, but that would require reclassification.

Another point is that the dataset is aggregated by different people with varying levels of expertise in the field, as the form is open to anyone in the public [32]. Some projects are filled in by the people behind them, which makes the data that is filled in more trustworthy, while other projects are found by regular people on the internet, and thus filled in by a ”third party” (Figure 3.6). People from The Index Project, who are more familiar with their system, also spend time filling in projects, and thus could be considered a bit more knowledgeable and trustworthy.

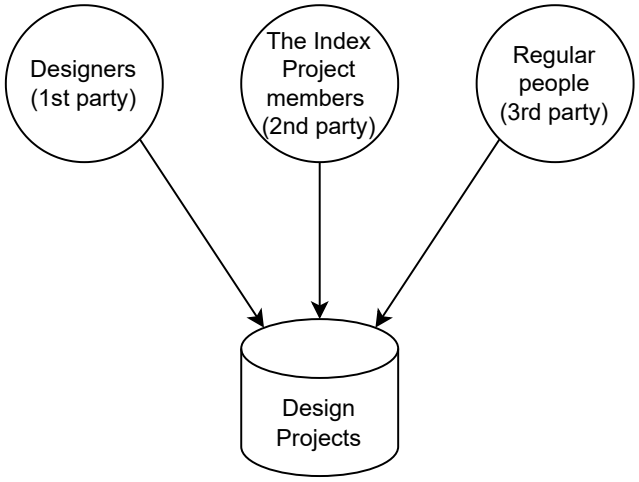


Figure 3.6: Different sources of information for the dataset

When we look more broadly at the concept of trust, there is also the idea that categorization can become stale. A project that has belonged to one category 10 years ago might belong to a different one with our current distinction and understanding of the categories. Thus, part of trust could also be assigned based on the time of creation in the dataset, assigning lower trust to projects that were inserted into the database a long time ago, and higher trust for more recent projects.

In general, the notion of trust does not apply to all, or even most datasets. A lot of existing datasets are classified by the same entity, span for a short amount of time, and do not have changing rules of the classification categories. The concept of trust should be applied to datasets with possible errors in their classification. As such, the datasets to be considered should have at least one of the following qualities:

- *Subjective categories.* This means there are not rigid rules on which classification category the datapoints could fall under (e.g. ratings of movies).
- *Potentially incorrect classification.* The causes could be many, but there has to be a belief that the classification of some of the datapoints could be wrong (e.g. the sensors of a system were inaccurate on a some days due to high temperature).
- *A change in knowledge or rules of classification.* A new piece of information, or a new form of understanding that affects the classification; perhaps a piece of input was missing, or classification categories have been added/removed. Even datasets that were classified objectively might not be very accurate in today’s standards if a certain piece of knowledge was not taken into consideration at the time.

The dataset used by The Index Project arguably falls under all of these, and thus the concept of trust and its potential usage is to be explored.

3.2.3 Reclassification

From all of this, it could be seen that each project has a varying degree of trust when it comes to their classification. This notion has been part of the inspiration behind the Iterative Reclassification algorithm, which shall be discussed in Section 4.2. In general, the idea is that each design could be assigned a specific trust value, which would then be used to reclassify some of the existing projects, leaning towards reclassification of projects that have low trust.

The reason behind reclassification is to potentially end up with a more accurate dataset than the one that was started with, as well as potentially improving the accuracy of the trained model - the prospects of that shall be explored in Section 5.2.

4 Proposed Solution

4.1 Classification Models

For classification, the models were constructed using Fastai [6]. The technique to construct the models is called transfer learning. The idea is to use a more general, pretrained model, and tune it to the specific data of your choice. The technique is often used for images (starting with a general image model that is able to predict many different objects), but can also be used for text. In the case of text, Fastai uses a model that's pretrained on the text in Wikipedia, called Wikitext 103 [36]. This model is trained to predict the next word in a piece of text, but by stripping the input and output layers, we can use it for classification as well. As an in-between step, one can also train the language model on domain-specific text; this will not be done in this project, however. Since the layers in the middle of the NN are kept, the model can still recognize general language patterns, and thus it becomes a bit quicker during training to get decent accuracy.

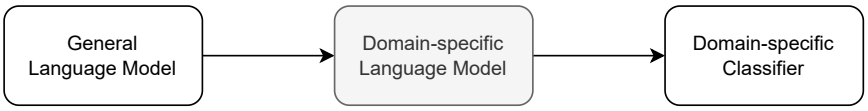


Figure 4.1: Transfer learning

Three model structures were constructed in total - one for categories, one for goals, and a multi-target one (predicting both at the same time). We will mainly be using the first two, as the last one was more of a proof-of-concept, and did not end up working very well (and for testing purposes, it is more beneficial to isolate variables, i.e. test category and goals separately). Some tests shall be conducted using the Iterative Reclassification algorithm, described in the following section. By varying the input text and algorithm parameters, the accuracy of each model shall be compared in Section 5.2.

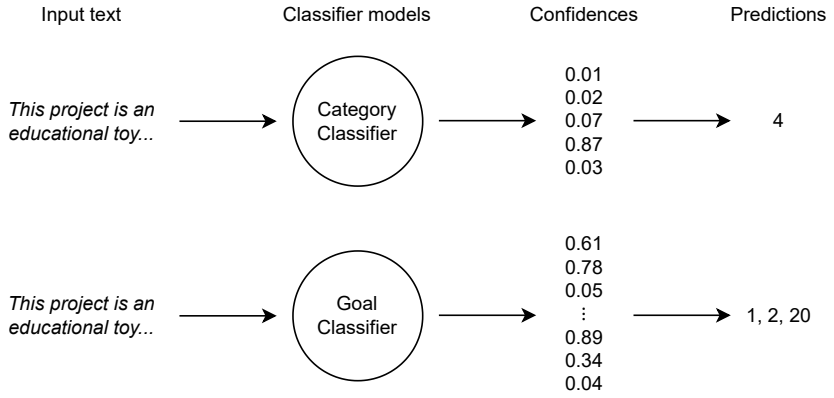


Figure 4.2: Category and goal classification models

In machine learning terms, a model that can predict multiple classification tags for a single datapoint (such as multiple goals for each design project in this dataset) is refered to as a "multi-category" model. However, as to not confuse it with the term "categories" already

being used for classifying design projects, they shall continue to be referred to as ”target goals” in this project.

4.2 Iterative Reclassification

The algorithm proposed here is meant to tackle datasets that could have misclassified data in them. As such, it will not be applicable to many datasets, but it is proposed that usage of the algorithm could benefit some users in two main ways. First, it might improve the accuracy of a ML model being built using their data (the prospects of this are explored in Section 5.2). Second, the algorithm reclassifies currently existing data, such that it might become more accurate.

One of the key concepts used in the algorithm is a numeric trust value. Each datapoint in the dataset is assigned a trust value - a floating point number from 0 to 1. The lower the value of trust, the less the algorithm trusts the existing classification of the datapoint. An example is that a design project that has been assigned a trust value of `0.2` has a higher chance of being reclassified than a project with a trust value of `0.5`.

id	name	trust
1	Scanadu	0.200000
10	Oculus Rift	0.201164
16	WRISTIFY	0.201939
17	Liftware	0.202069
18	Sproutling	0.202198
...	...	...
6185	AMI (Help child abuse)	0.999483
6186	Furniture for blind children who develop motor...	0.999612
6187	Urban connection skeleton	0.999741
6188	Waste Not Why Not Podcast	0.799871
6189	Teemill	0.800000

Figure 4.3: Dataset sample with trust

The choice of how a trust value is derived is for the user to decide. Some aspects of trust have been discussed in Section 3.2.2, and these will be taken into consideration when calculating the trust of each datapoint in this dataset. The formula that ended up being used is the following:

$$\texttt{trust} = 0.8 * \texttt{time} + 0.2 * \max(\texttt{self}, \texttt{tip})$$

- `time`: floating point value, `0` for the oldest datapoint, `1` for the most recent.
- `self`: `1` if the project was nominated by a 1st party, `0` otherwise.
- `tip`: `1` if the project was nominated by someone from The Index Project, `0` otherwise.

A separate trust value for the classification of categories and target goals could be calculated, but in this case, the formulas would end up being somewhat similar, and thus it was not deemed necessary (the one for target goals might have had a more complicated expression of time due to the changing rules, but again, the assumption is that the further in time the classification was made, the more accurate it is). The current formula is mostly based on the time each project was put into the database. On top of that, a `0.2` bonus is

added if the classification was done by either a 1st party (creators of the design) or 2nd party (people working at The Index Project), as these groups are considered to be more knowledgeable.

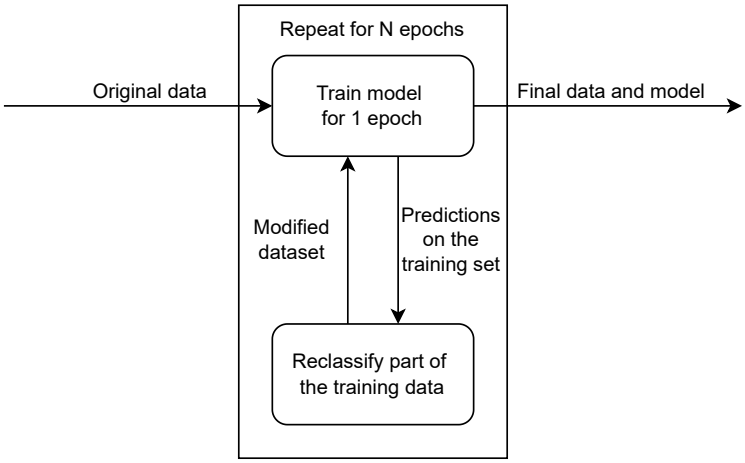


Figure 4.4: Iterative Reclassification flow

The idea behind Iterative Reclassification (IR) is rather simple (Figure 4.4). Training of models is usually done in multiple epochs. After training for a single epoch, we predict how each datapoint in the training set should be classified (in our case, which category or target goals it belongs to). An ML model, apart from providing predictions, also provides confidence values - how confident the model is that the prediction is true. Thus, we use a conjunction of confidence and trust to reclassify a part of the current dataset - the more confident the model is that the current classification is incorrect, and the lower the trust value of the existing classification of the datapoint is, the more likely it is to be reclassified.

Thus, we start by assigning a trust value for each datapoint. Then, while training, after each epoch, we use the model to predict the datapoints in the training set. We filter to only get datapoints with predictions that do not match currently existing classifications (i.e. getting predictions that the model deems incorrect). These datapoints are then sorted based on a combination of the confidence of the model, as well as their trust value. A percentage of the top datapoints are then reclassified, and the model is ready for the following epoch, which is then trained on the newly altered dataset.

The following listings contain the pseudocode for Iterative Reclassification for regular categories. Some example values have been given as comments in the pseudocode.

```
function ir(model, dataset, epochs, p, t, delay):
  for i to epochs:
    model.train(dataset)
    if i >= delay:
      dataset = reclassify(model, dataset, p, t)
  return model, dataset
```

Listing 4.1: Pseudocode for Iterative Reclassification

```
function reclassify(model, dataset, p, t):
  for row in dataset:
    preds = model.predict(row) # {"1": 0.6, "2": 0.1, ...}
    row.conf = max(preds).value # 0.6
    row.pred = max(preds).key # "1"
    row.wrong = row.pred != row.category # true

  dataset.sort(conf - t*trust)

  count = ceil(dataset.filter(training).length * p)
  for row in dataset.filter(wrong and training):
    if count == 0:
      break
    row.category = row.pred
    count = count - 1
  return dataset
```

Listing 4.2: Pseudocode to reclassify regular categories

The arguments used in the algorithm are as follows:

- `model`: A machine learning model. Can be trained on data, and can be used to predict values based on an input.
- `epochs`: Number of epochs to train the model for.
- `p`: Percentage of datapoints to reclassify after each epoch. `1` would reclassify every datapoint in the dataset after each epoch, and `0` would not reclassify anything.
- `t`: How much emphasis to put on trust, versus the model's confidence. `1` would make trust as important as the confidence value.
- `delay`: Number of epochs to skip reclassification for. Used if the model is slow to gain a certain baseline accuracy, and we wish to delay reclassification.

The `ir()` function holds the core idea of Iterative Reclassification, which is to keep training a model for a single epoch, and then reclassify the dataset used. During reclassification, we start by predicting every datapoint (`row`) in the dataset. This yields the confidences of each category. In this case, the predicted category is the one with the highest confidence, so we take that prediction and its confidence into account. We then mark if the datapoint's current category might be wrong, i.e. the prediction from the model does not match the current category.

After predictions are done, we sort the dataset in descending order based on the confidence of the model, subtracted by trust. Thus, datapoints with low trust and high model confidence will be sorted towards the beginning. `count` determines the number of datapoints the algorithm will be reclassifying - this is based on the length of the training set,

as well as `p`. Lastly, we filter out only the datapoints that are wrong and in the training set, and for the first ones, change their existing category with the one predicted by the model.

From the arguments, by adjusting the `t` value to 0, we can make the algorithm not utilize trust whatsoever; in other words, trust is not necessary for the algorithm to work, as the sorting could be done based on the confidence of the model alone. The effectiveness of using trust versus not using it shall be explored in Chapter 5.

Adjusting `p` to 0 will essentially train the model in the regular manner, without reclassifying anything.

For target goals, the algorithm has to be adjusted, since each project can have multiple goals.

```
function reclassify(model, dataset, p, t):
  thresh = 0.5
  min_goals = 1
  max_goals = 5

  for row in dataset:
    preds = model.predict(row) # {"1": 0.6, "2": 0.1, ...}
    preds_new = preds.filter(key not in row.goals)
    preds_current = preds.filter(key in row.goals)
    row.conf_add = max(preds_new).value # 0.6
    row.pred_add = max(preds_new).key # "1"
    row.conf_remove = 1-min(preds_current).value # 0.9
    row.pred_remove = min(preds_current).key # "2"
    row.wrong = max(conf_add, conf_remove) > thresh # true

  dataset.sort(max(conf_add, conf_remove) - t*trust))

  count = ceil(dataset.filter(training).length * p)
  for row in dataset.filter(wrong and training):
    if count == 0:
      break
    if row.conf_remove > row.conf_add:
      if row.goals.length == min_goals:
        row.goals.replace(row.pred_remove, row.pred_add)
      else:
        row.goals.remove(row.pred_remove)
    else:
      if row.goals.length == max_goals:
        row.goals.replace(row.pred_remove, row.pred_add)
      else:
        row.goals.add(row.pred_add)
    count = count - 1

  return dataset
```

Listing 4.3: Pseudocode to reclassify goals

Since we are dealing with multiple target goals per datapoint, some alterations had to be made. First, a few additional variables have been set:

- `thresh` - if the confidence of a goal is higher than the threshold, it is added to the possible predictions. Confidence values are usually passed through a sigmoid function, and thus result in values between 0 and 1.
- `min_goals` - minimum amount of goals that have to be selected per each datapoint.
- `max_goals` - maximum amount of goals that can be selected per each datapoint.

The key difference in this version of the algorithm is that it will be adding or removing one of the goals in each datapoint, instead of replacing the whole set of goals.

Just like before, we start by predicting target goals for each datapoint (`row`). We split the confidences of each goal into those that are new (i.e. not in the currently existing goals), and those for the existing set of goals. This time, we care about two specific goals - one of them has the maximum confidence out of the new ones, which signifies that the model thinks this prediction should be added. The other one has the lowest confidence out of those already existing in the datapoint - by subtracting it from `1`, this signifies the confidence that the model thinks this goal should be removed. To check whether the model thinks the existing classification is wrong, we check if either the confidence to add a new goal or the confidence to remove an existing goal is higher than 0.5.

The dataset is then sorted according to whichever confidence is higher, minus the trust. Just like before, we calculate the `count`, and get the datapoints that are wrongly classified and in the training set. This time, for the first ones, we check whether the confidence to remove a goal is larger than the one to add a goal. If so, and if the number of goals has reached the minimum amount, then we replace the goal with the one that had the largest confidence. If the number of goals is larger, then we just remove the one with the lowest existing confidence. If the confidence to remove a goal is lower than the one to add, we check if the number of existing goals has reached the maximum amount - if so, we replace one of the existing goals with the new one. Otherwise, the new one is added to the set.

Some of these rules might not apply for datasets where there are no minimum or maximum amount of classification categories. In our case however, each design project has to have at least 1, and at most 5 target goals.

Motivation
The reason why it is believed that the model's accuracy could improve is that the model would no longer be ”dragged down” by inaccurate classification. As an example, imagine certain misclassified datapoints in the dataset constantly stating that they belong to a certain category. Due to this error, the whole model shifts its thinking of how to classify other datapoints for that category, further misclassifying other datapoints. Of course, it is also a possibility that by using IR, the algorithm might get higher and higher confirmation bias, as it takes control of changing its own input as it deems fit. For instance, if the model is confident about classifying only two out of the five categories, it might start reclassifying other datapoints only into those two categories, which can have a snowball effect, as the whole dataset is categorized into only those two categories. Thus, it is important to put the algorithm to a test using real data. It is also important to have a separate test set, as the model can get overfitted to the validation set as well. Accuracy measurements using The Index Project's dataset shall be provided in Section 5.2.

4.3 Measuring Accuracy

One might wonder how the accuracy for models was calculated. For regular categories, it is rather straightforward - we measure the percentage of datapoints where the model's prediction is the same as the existing category. However, for target goals, where each datapoint can have multiple labels, the measurement function is different. The default metric for calculating multi-category accuracy in the Fastai library is called `accuracy_multi` [37]. It takes each datapoint, and predicts what goals it would assign. Then, it compares that to the existing goals of the datapoint. A sum of correct guesses (including true negatives) is divided by the total number of goals, and the average of those for each datapoint determines the overall accuracy of the model.

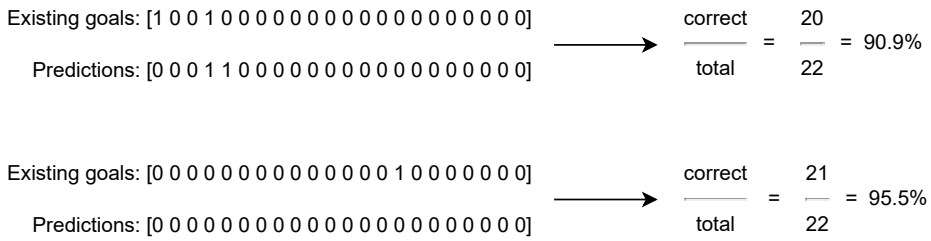


Figure 4.5: An example of using the `accuracy_multi` metric

As seen in Figure 4.5, a problem arises when we have a large amount of total goals, and a small amount of goals per datapoint. In this metric, a guess that a goal does not exist is also counted as a guess, meaning that in this scenario with 22 total goals, if a datapoint has 1 goal assigned to it, a model guessing none of the goals would still achieve 95.5% accuracy. As such, this way of measuring accuracy was unsuitable for our measurements, and a new metric was created. The metric that was chosen calculates Intersection over Union (IoU), which is a concept borrowed from ML image detection models [38]. Using this concept, we take the existing goals and predicted ones as sets, and calculate the length of their intersection divided by the length of the union (Figure 4.6). This way, the accuracy increases for every goal guessed right, and decreases for every mistake, without taking into account the true negative guesses.

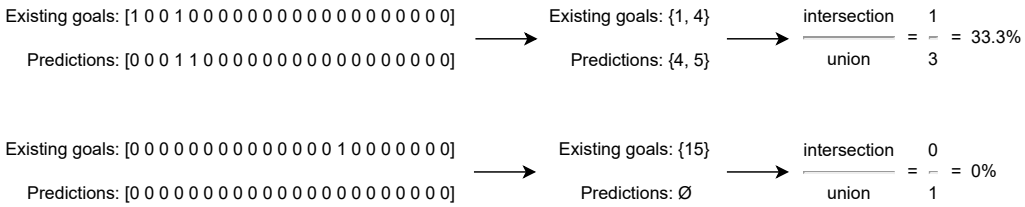


Figure 4.6: An example of using the IoU accuracy metric

4.4 Tools

The implementation was build using Jupyter Notebooks [39] and Python. The reason for this choice was due to the fact that many of the state-of-the-art ML libraries have Python implementations. As for Jupyter Notebooks, the format allows to split the code into chunks which can be run separately, but all retain the same variables and spot in the memory. It also allows to retain output information between sessions. This format is commonly used for ML tasks, as a model can be trained in one code chunk, and used in various different ways in other chunks without the need to retrain the model again when changing the code further on - this allows for faster experimentation.

The ML library of choice was Fastai v2 [6]. The library provides many high-level tools to make a model, which results in relatively few lines of code for the end user. That being said, the library has somewhat limited documentation, and many issues were encountered while using it. The library is built on top of PyTorch [40], which is a much lower-level ML library. Fastai also has a course [41] that follows the basics of using their library, as well as the basic principles of ML.

For web scraping, a library called Selenium [42] was used. The library was used to simulate a Firefox [43] browser, through which the relevant text data for each project was

downloaded.

For translations from Spanish to English, a library called Translators [44] was used. In the background, the library can use many different services to translate; the service that was chosen for this project was Google Translate [45].

For plotting, a data visualization library called Matplotlib [46] was used. All plotted charts shown in Chapter 5 were created using this library.

5 Results

In the following section, tests are made to train the models under various conditions. We start by discussing the effects of including web scraped data in the text input of category and goal models. Afterwards, we explore the effects of the proposed IR algorithm, in order to find the conditions with which the use of the algorithm is beneficial.

5.1 Web Scraping

We start with a comparison of several models trained using various columns in the dataset. We will be working using the three sets mentioned in Section 3.2: a training set, validation set, and a test set. For the following tests, the test set has its `long_description` column erased - this is used to simulate the scenario where new projects would be predicted solely based on the data gathered by scraping their website for text (this was the initial use case proposed to The Index Project). The training and validation sets are left unaltered. Also, this section does not contain usage of the Iterative Reclassification algorithm, which was proposed in previous sections; the results of using the algorithm shall be explored in Section 5.2.

Before training each model, randomness generation seeds have been set to the same value, which makes the results replicable, as well as providing a more consistent basis for comparison.

5.1.1 Category

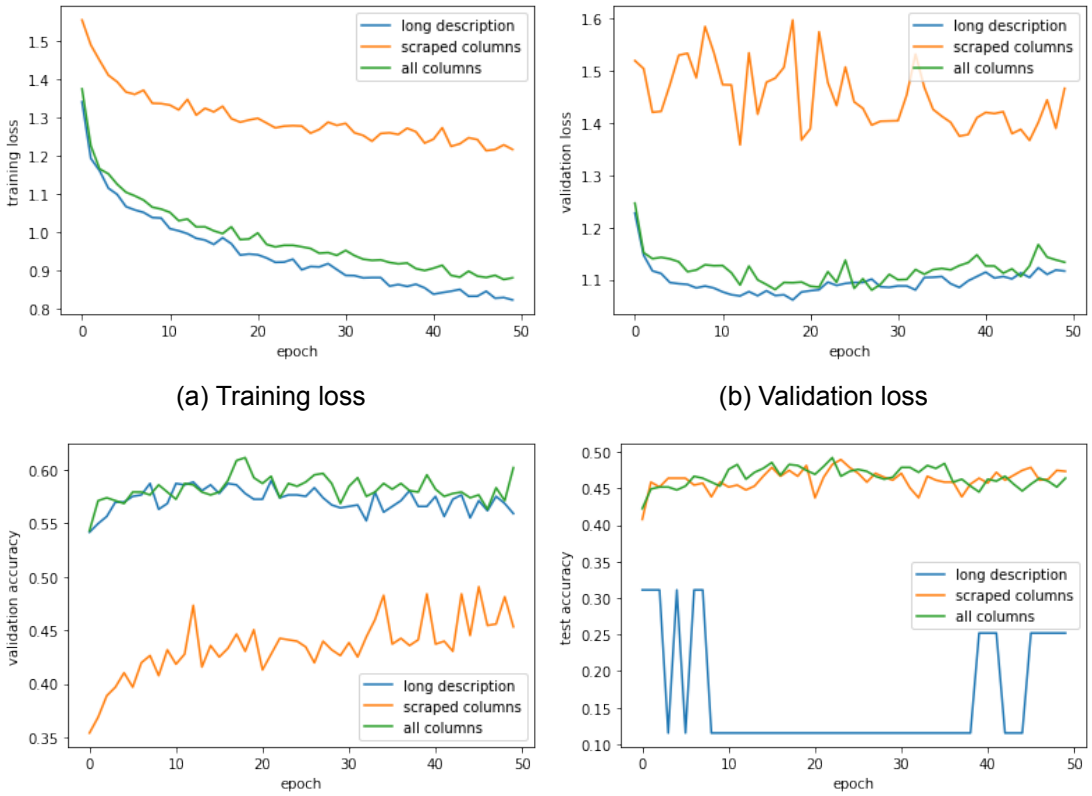
First, for a period of 50 epochs, different models have been trained to predict the category of each datapoint. To iterate, for all of the models, the test set is only using the scraped columns. As for training and validation, the three models include:

- *long description*: uses only `long_description` as the input.
- *scraped columns*: uses only the web scraped columns (`page_title`, `page_body`, `meta_title`, `meta_description`).
- *all columns*: uses all five of the aforementioned columns.

The models simulate different scenarios of input data. The first model simulates the scenario where input data is kept as it was in the starting dataset. The quality of the description text is relatively high, as it was picked manually by humans. The length varies, with a maximum of 1351 characters, and an average of 763 characters.

The second model simulates the input data being the same as the data we're trying to classify, that being only the web scraped text. The overall quality of the data is rather poor, as much of the text is irrelevant, and many datapoints have inaccessible websites, resulting in little to no text. By combining the text of all four columns, the length is extremely varying - the maximum length of a single datapoint reaches 225725 characters, while the average is 2620 characters.

The third model contains all of the text columns, and represents the scenario where we "supplement" the original input data with additional web scraped data, to provide more information about each project, and more closely resemble the text in the test dataset.



(a) Training loss

(b) Validation loss

(c) Validation accuracy

(d) Test accuracy

Figure 5.1: Comparing web-scraped column models for predicting the category

From the training loss, it can be seen that all models have the loss steadily decreasing, as is expected. The validation loss for *scraped columns* however is all over the place, seeming to indicate that using scraped columns on their own is not a consistent indicator for predicting the category. This can also be seen in Figure 5.1c, where the validation accuracy is quite low compared to the other two models. However, when it comes to the test set, the model compares admirably, with a similar reached accuracy as in the validation set (this is to be expected, as in this case, both sets use only the web scraped columns). The other two sets trade blows in validation accuracy, but notably, the *long description* model performs very poorly in the test set, seeming to suggest that adding more relevant text columns (as was done for *all columns*) was a good idea. This is a good example of domain shift, a problem where the model is trained on data that is very different from the data that is being used for inference (in this case, the test set).

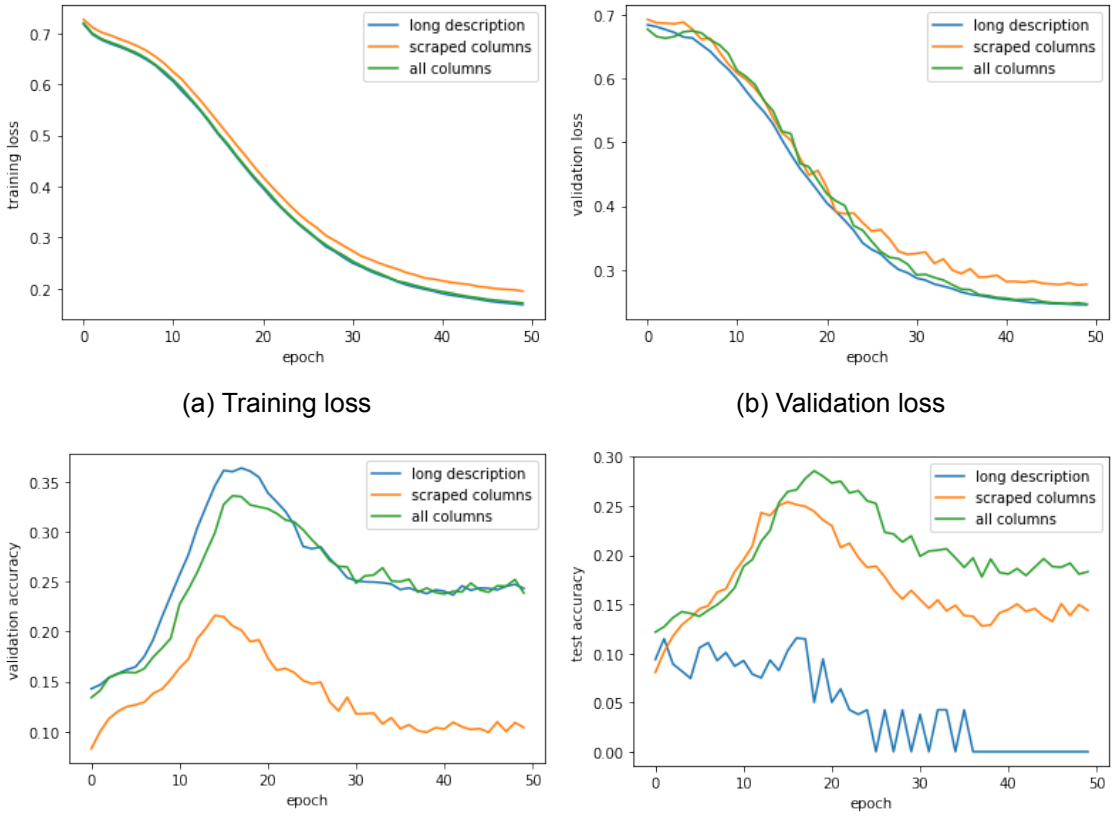
Overall, the highest test set accuracy was achieved with the *all columns* model (Table 5.1), although the difference is not too large compared to the *scraped columns* model. The training time was also a bit longer (due to longer input text).

	Max validation accuracy	Max test accuracy
long description	59.0 %	31.1 %
scraped columns	49.1 %	48.9 %
all columns	61.1 %	49.2 %

Table 5.1: Accuracy comparison for category models using web scraped text

5.1.2 Goals

For goals, the same approach was made: three models were trained for 50 epochs each. The following are the results:



(a) Training loss

(b) Validation loss

(c) Validation accuracy

(d) Test accuracy

Figure 5.2: Comparing web-scraped column models for predicting goals

Similar conclusions can be drawn from these results as was done for the previous models. Again, the validation set accuracy for using just the scraped columns is much lower than the other two models, and the test set accuracy of the *long description* model is much lower, to the point of reaching 0 in the last epochs. Notably, the difference in test accuracy of the *all columns* model and the *scraped columns* models is higher this time, suggesting that the combination of both `long_description` and the web scraped columns performs the best in this scenario, and is thus worth doing. The maximum test accuracy achieved was 28.6%, and all of the other metrics can be seen in Table 5.2.

	Max validation accuracy	Max test accuracy
long description	36.4 %	11.6 %
scraped columns	21.6 %	25.4 %
all columns	33.6 %	28.6 %

Table 5.2: Accuracy comparison for goal models using web scraped text

5.2 Iterative Reclassification

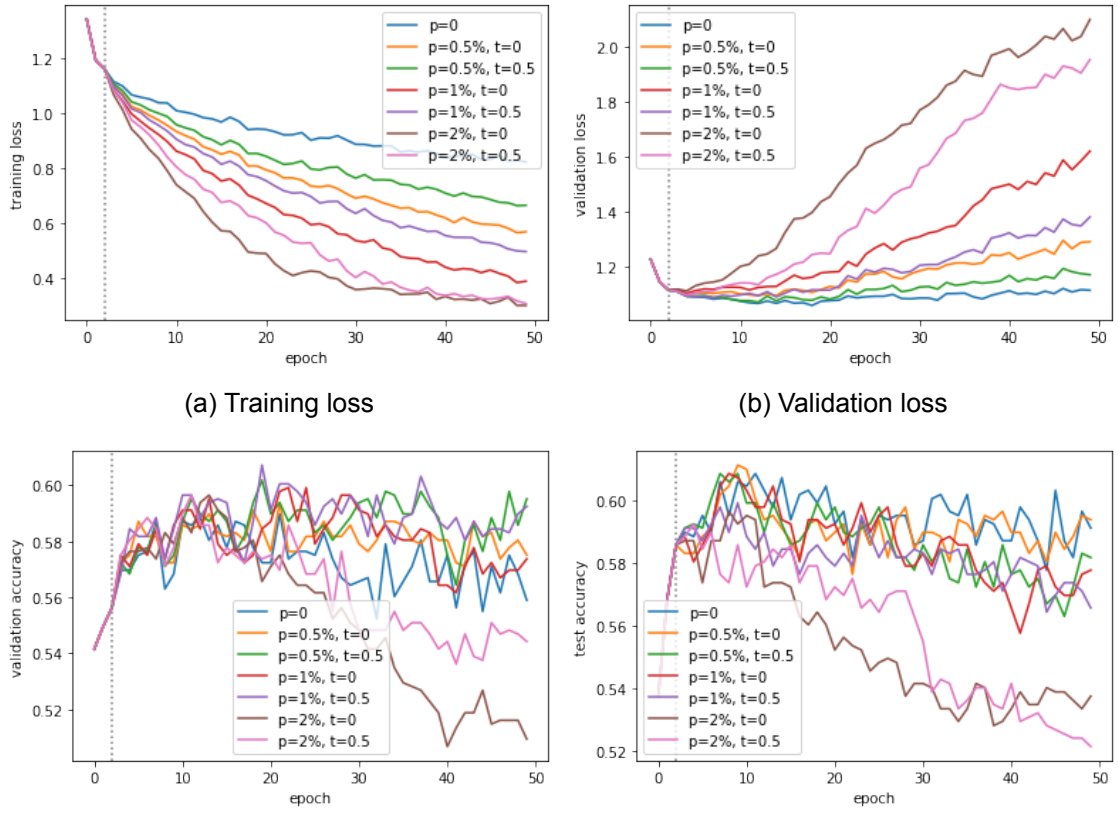
In this section, multiple models shall be constructed to test out the proposed Iterative Reclassification algorithm. Each model will be trained using the same data, with adjustments made to the various parameters in order to see how the model learning changes. As input, only the `long_description` column shall be used to produce shorter learning times. The test set predictions will be done using just the `long_description` text as well.

The parameters that shall be adjusted include:

- `p`: percentage of datapoints that are reclassified each epoch.
- `t`: weight assigned to trust.
- `delay`: number of epochs to delay reclassification for.

5.2.1 Category

To start, several models have been constructed to predict category. A delay of 2 epochs was used as a starting point (marked with a gray dotted line). As a baseline, a model with `p=0` is used - this is a model where no reclassification was made. The other models vary in their rate of reclassification `p`, as well as each model having a counterpart where trust is not utilized, relying solely on the confidence of the model.



(a) Training loss

(b) Validation loss

(c) Validation accuracy

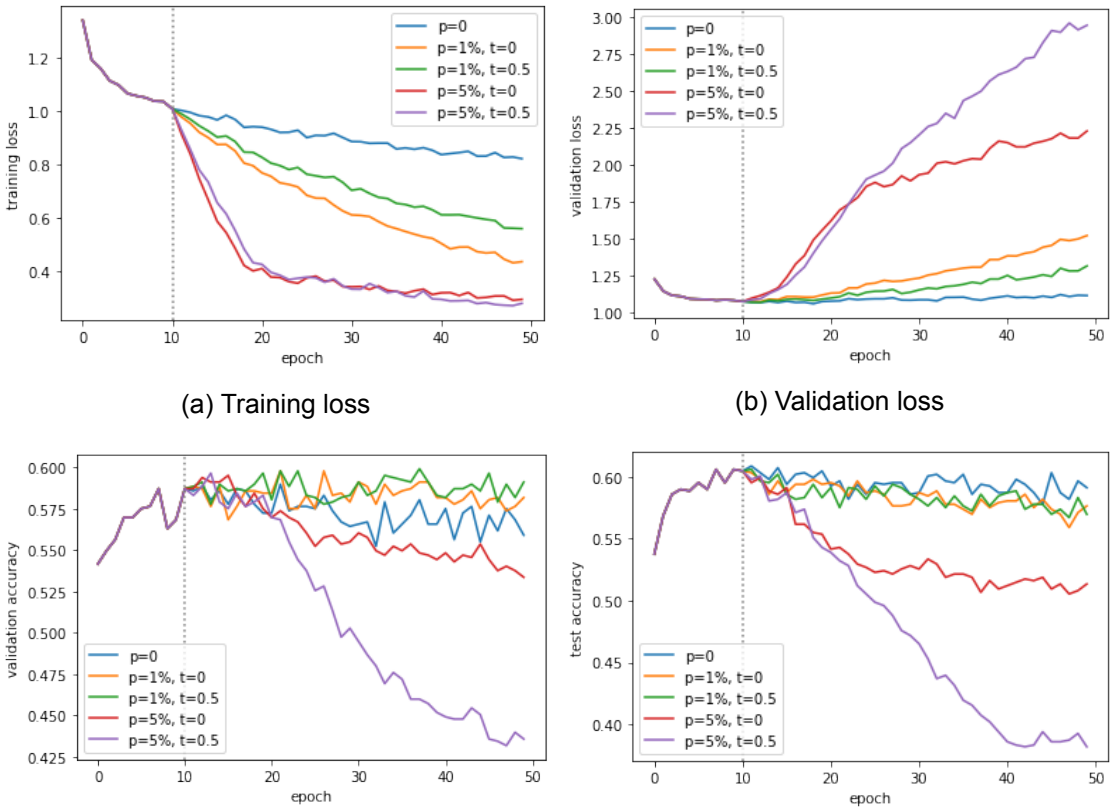
(d) Test accuracy

Figure 5.3: Comparing IR models for predicting the category (`delay=2`)

From the training loss, it can be seen that the stronger the reclassification, the faster it goes down, as well as the validation loss going up faster. As was speculated, this appears to be a case of faster overfitting - the model's opinion becomes biased based

on the training set, and it becomes worse at making generalized guesses. This makes sense, as reclassification alters the training dataset based on the model's predictions. However, when testing accuracy, there were some peaks which were slightly higher than the base model, albeit not by a lot. This could be accredited to the model changing some misclassified datapoints, though the effect is very slight.

After this, a few tests were done with an increased delay of 10. The theory was that by delaying the reclassification, the model would make better reclassification decisions.



(a) Training loss

(b) Validation loss

(c) Validation accuracy

(d) Test accuracy

Figure 5.4: Comparing IR models for predicting the category (`delay=10`)

Since reclassification was delayed, a larger `p` value was tested - this was done so that the model had the opportunity to reclassify more datapoints before the accuracy got diminished. However, a value of 5% still appeared to be too large, and the model quickly lost its accuracy.

A comparison of all the max accuracy metrics are listed in Table 5.3. Since the accuracy of the validation set is still relevant in this case, a combined accuracy metric was also measured - this is the top accuracy the model could achieve if used on a dataset containing both validation and test sets. The highest combined accuracy achieved in this case was 60%, which is a very small increase when compared to the base model's max combined accuracy of 59.7%.

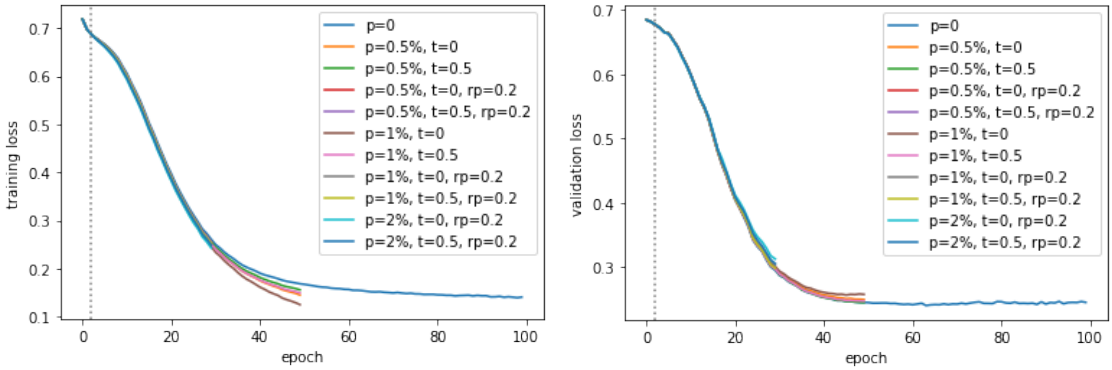
p	t	delay	max validation accuracy	max test accuracy	max combined accuracy
0			59.0 %	60.9 %	59.7 %
0.5%	0	2	59.4 %	61.1 %	59.8 %
0.5%	0.5	2	60.2 %	60.9 %	59.8 %
1 %	0	2	59.9 %	60.9 %	60.0 %
1 %	0.5	2	60.7 %	59.9 %	59.7 %
2 %	0	2	59.7 %	59.7 %	59.2 %
2 %	0.5	2	59.7 %	59.2 %	58.9 %
1 %	0	10	59.8 %	60.6 %	59.7 %
1 %	0.5	10	59.9 %	60.6 %	59.7 %
5 %	0	10	59.5 %	60.6 %	59.7 %
5 %	0.5	10	59.7 %	60.6 %	59.7 %

Table 5.3: Accuracy comparison for category models using IR

5.2.2 Goals

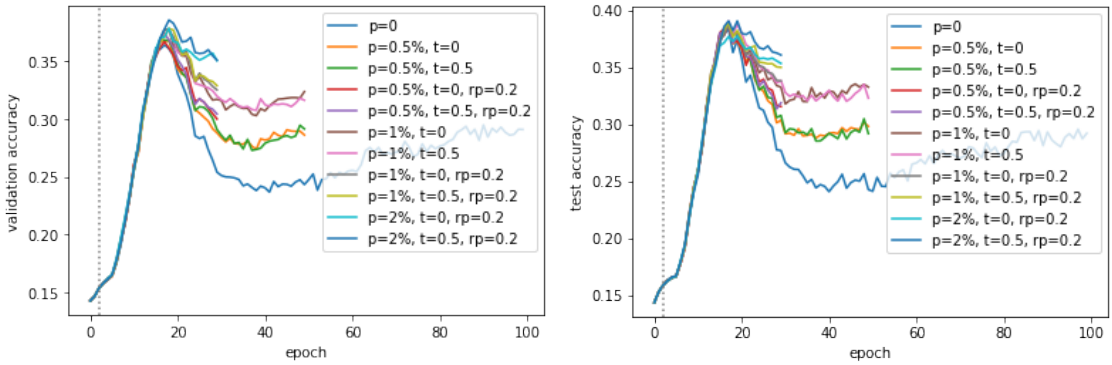
For goals, a similar approach was made, where models with varying `p`, `t` and `delay` values were trained. The base model was trained for 100 epochs, just to see the result patterns, and all the other models were trained to a smaller number of epochs. This does not affect measurements in terms of max accuracy, as that is achieved somewhat early in the training process.

However, an issue arose when training the models. Since some of the goals are used very sparingly compared to some others, the model can become less confident regarding the rarer goals. For an existing goal on a datapoint, if the confidence is low enough, it is treated as if the model wanted to remove that goal. This may not be a problem on its own, but can spiral out of control, as a model becomes less and less confident about a specific goal, until it is reclassified out of existence. If a goal that was in the starting dataset no longer exists, the training crashes. This happened a few times when training for a longer amount of epochs. For this reason, a new parameter was added - a ”reduction penalty” (`rp`). This introduces a confidence penalty for removing existing goals, making the model add new goals more often, rather than removing them.



(a) Training loss

(b) Validation loss

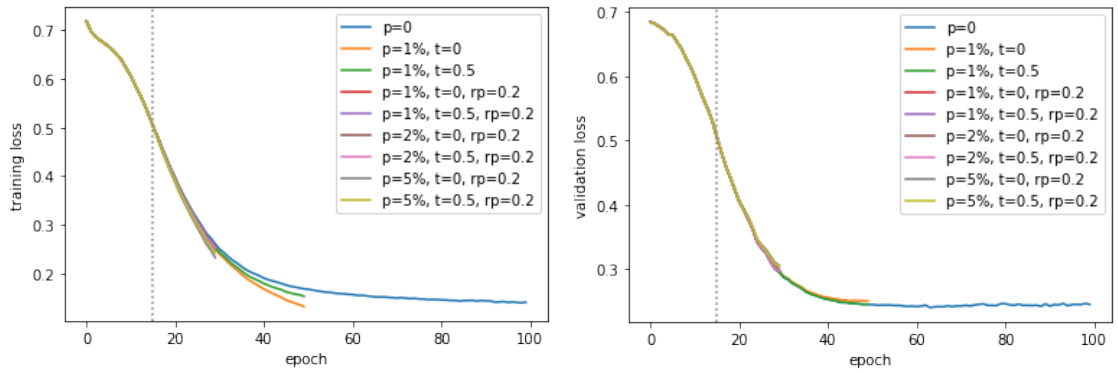


(c) Validation accuracy

(d) Test accuracy

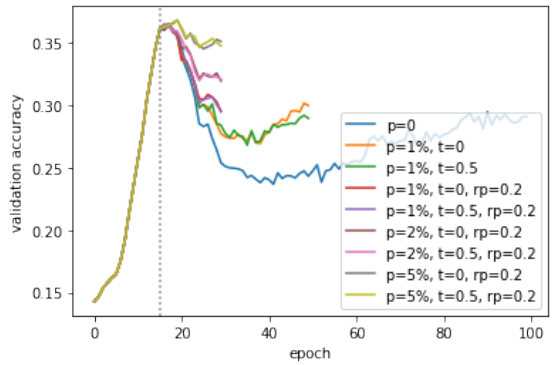
Figure 5.5: Comparing IR models for predicting goals (`delay=2`)

Compared to categories, the changes in training and validation loss are much more slight. This makes sense when considering that a single datapoint has only one of its goals reclassified each epoch, so less of the overall distribution of goals is changed per each epoch this way, even though `p` values might be equal. The accuracy of several IR models did manage to peak just a bit above the base model. Of note is that models that do not use trust (i.e. `t=0`) are very close in performance to their counterparts (i.e. `t=0.5`). This would suggest that using trust in IR does not necessarily have a large effect on accuracy.

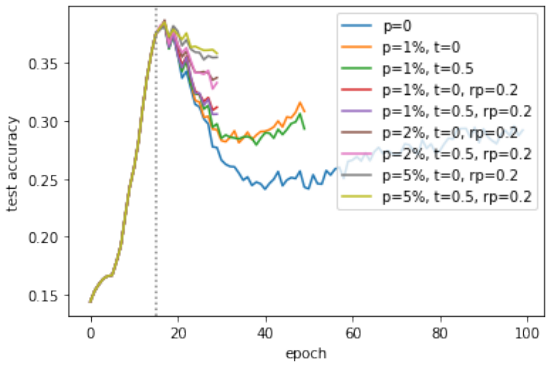


(a) Training loss

(b) Validation loss



(c) Validation accuracy



(d) Test accuracy

Figure 5.6: Comparing IR models for predicting goals (`delay=15`)

As was done with categories, several models were trained with a larger delay. The data from graphs did not differ too much visually from the models trained with a lower delay. The overall accuracies are listed in Table 5.4. The maximum combined accuracy achieved in the case of goals is 38.7%, compared to the base combined accuracy of 37.4%. This is a comparatively more sizeable increase in accuracy than was achieved with categories, and perhaps could be attributed to the fact that target goals had more changes in the lifespan of this dataset, resulting in more misclassified datapoints, thus benefitting from reclassification more.

p	t	rp	delay	max validation accuracy	max test accuracy	max combined accuracy
0				36.4 %	38.3 %	37.4 %
0.5%	0	0	2	36.8 %	38.7 %	37.7 %
0.5%	0.5	0	2	36.8 %	38.8 %	37.8 %
0.5%	0	0.2	2	36.7 %	38.8 %	37.7 %
0.5%	0.5	0.2	2	37.3 %	38.8 %	38.1 %
1 %	0	0	2	37.7 %	38.7 %	38.2 %
1 %	0.5	0	2	37.7 %	39.0 %	38.2 %
1 %	0	0.2	2	37.7 %	38.7 %	38.2 %
1 %	0.5	0.2	2	37.8 %	38.9 %	38.0 %
2 %	0	0.2	2	37.8 %	37.8 %	37.6 %
2 %	0.5	0.2	2	38.5 %	39.1 %	38.7 %
1 %	0	0	15	36.5 %	38.5 %	37.5 %
1 %	0.5	0	15	36.5 %	38.5 %	37.5 %
1 %	0	0.2	15	36.5 %	38.6 %	37.5 %
1 %	0.5	0.2	15	36.4 %	38.5 %	37.5 %
2 %	0	0.2	15	36.4 %	38.7 %	37.5 %
2 %	0.5	0.2	15	36.3 %	38.5 %	37.4 %
5 %	0	0.2	15	36.8 %	38.2 %	37.5 %
5 %	0.5	0.2	15	36.8 %	38.5 %	37.5 %

Table 5.4: Accuracy comparison for goal models using IR

5.3 Discussion

From the tests performed on web scraped columns, it seems that it is definitely worth the effort to make the input text format similar to the expected test set format; in other words, if the expected use case is to check for text scraped from websites, it benefits the model's final accuracy greatly if the training input is also in the format of web scraped text. That being said, adding more information to the training input, such as the `long_description` of each project, also seems to help, and the model's accuracy did not decrease from having ”too much information”. The only negative factor in this case was an increase in the length of time it takes to train the model.

From the tests performed on Iterative Reclassification, the changes in maximum accuracy compared to the base model are very slight in most cases. The algorithm also suffers from the danger of removing specific labels from the whole dataset. As such, reclassification makes more sense to use in very small percentages, as the number of what could be considered ”misclassified datapoints” even in this dataset appears to be very small. For most datasets, it would be recommended to train a model in the traditional way, and then to use it in conjunction with IR for only a handful of epochs, where the model with the highest accuracy would be saved.

With regards to trust, the models using it were very close to the ones not using it. This could suggest that for a large part of the datapoints, the trust value was inversely correlated to the model's own confidence, and thus using trust wasn't affecting the outcome by much. On the other hand, when we decide ourselves on what to put trust on in the dataset, we start to bias the model in the way we think it should classify. Whether that is good or not depends on the use case, but it might not necessarily be reflected in the accuracy of the model.

Bibliography

[1] *The Index Project*. Accessed: 2022-11-30. URL: https://www.theindexproject.org/.

[2] *Google: Ads and data*. Accessed: 2022-11-29. URL: https://safety.google/privacy/ads-and-data/.

[3] *Data: The Most Valuable Commodity for Businesses*. Accessed: 2022-11-29. URL: https://www.kdnuggets.com/2022/03/data-valuable-commodity-businesses.html.

[4] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

[5] *DALL-E 2*. Accessed: 2022-11-29. URL: https://openai.com/dall-e-2/.

[6] *fast.ai*. Accessed: 2022-12-03. URL: https://docs.fast.ai/.

[7] Jeremy Howard and Sebastian Ruder. *Universal Language Model Fine-tuning for Text Classification*. 2018. DOI: 10.48550/ARXIV.1801.06146. URL: https://arxiv.org/abs/1801.06146.

[8] Leslie N. Smith. *Cyclical Learning Rates for Training Neural Networks*. 2015. DOI: 10.48550/ARXIV.1506.01186. URL: https://arxiv.org/abs/1506.01186.

[9] Leslie N. Smith and Nicholay Topin. *Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates*. 2017. DOI: 10.48550/ARXIV.1708.07120. URL: https://arxiv.org/abs/1708.07120.

[10] Leslie N. Smith. *A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay*. 2018. DOI: 10.48550/ARXIV.1803.09820. URL: https://arxiv.org/abs/1803.09820.

[11] Jitendra Kumar et al. "Phishing website classification and detection using machine learning". eng. In: *2020 International Conference on Computer Communication and Informatics (iccci)* (2020), 6 pp. DOI: 10.1109/ICCCI48352.2020.9104161.

[12] Kushagra Krishna, Jaytrilok Choudhary, and Dhirendra Pratap Singh. "Malicious Webpage Classification". eng. In: *Internet of Things and Connected Technologies* (2021), pp. 389–399. DOI: 10.1007/978-3-030-76736-5_36.

[13] Prateek Gupta and Archana Singh. "Phishing Website Detection Using Machine Learning". eng. In: *Lecture Notes in Networks and Systems* 154 (2021), pp. 183–192. ISSN: 23673389, 23673370. DOI: 10.1007/978-981-15-8354-4_19.

[14] Leonardo Espinosa-Leal et al. "Website Classification from Webpage Renders". eng. In: *Proceedings of Elm2019. Proceedings in Adaptation, Learning and Optimization (palo 14)* (2021), pp. 41–50. DOI: 10.1007/978-3-030-58989-9_5.

[15] Makoto Tsukada, Takashi Washio, and Hiroshi Motoda. "Automatic web-page classification by using machine learning methods". eng. In: *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 2198 (2001), pp. 303–313. ISSN: 16113349, 03029743. DOI: 10.1007/3-540-45490-x_36.

[16] Hammad Haleem et al. "Effective probabilistic model for webpage classification". eng. In: *Lecture Notes in Electrical Engineering* 298 (2014), pp. 277–290. ISSN: 18761119, 18761100. DOI: 10.1007/978-81-322-1817-3_29.

[17] Fabrizio De Fausti, Francesco Pugliese, and Diego Zardetto. "Towards Automated Website Classification by Deep Learning". und. In: (2021).

[18] *Yahoo Japan*. Accessed: 2022-12-10. URL: https://www.yahoo.co.jp/.

[19] *What are Stop Words*. Accessed: 2022-12-10. URL: https://kavita-ganesan.com/what-are-stop-words/.
[20] Patrick A. Hoyer et al. “Machine learning-based re-classification of the geochemical stratigraphy of the Rajahmundry Traps, India”. eng. In: *Journal of Volcanology and Geothermal Research* 428 (2022), p. 107594. ISSN: 18726097, 03770273. DOI: 10.1016/j.jvolgeores.2022.107594.
[21] Saurabh Das, Chandrani Chatterjee, and Swastika Chakraborty. “A Machine Learning Approach to Re-Classification of Climate Zones Based on Multiple Rain Features over India”. eng. In: *International Geoscience and Remote Sensing Symposium (igarss)* (2019), pp. 7752–7754. ISSN: 21537003, 21536996. DOI: 10.1109/IGARSS.2019.8898422.
[22] A Kobler, S Dzeroski, and L Keramitsoglou. “Habitat mapping using machine learning-extended kernel-based reclassification of an Ikonos satellite image”. eng. In: *Ecological Modelling* 191.1 (2006), pp. 83–95. ISSN: 18727026, 03043800. DOI: 10.1016/j.ecolmodel.2005.08.002.
[23] Chulwoo Han. “Bimodal Characteristic Returns and Predictability Enhancement via Machine Learning”. eng. In: *Management Science* 68.10 (2022), pp. 7701–7741. ISSN: 15265501, 00251909. DOI: 10.1287/mnsc.2021.4189.
[24] Chang Ming et al. “Machine learning-based lifetime breast cancer risk reclassification compared with the BOADICEA model: impact on screening recommendations”. eng. In: *British Journal of Cancer* 123.5 (2020), pp. 860–867. ISSN: 15321827, 00070920. DOI: 10.1038/s41416-020-0937-0.
[25] John Rothfels and Julie Tibshirani. “Unsupervised sentiment classification of English movie reviews using automatic selection of positive and negative sentiment items”. In: *CS224N-Final Project* 43.2 (2010), pp. 52–56.
[26] Bin Bin Peng, Zheng Xing Sun, and Xiao Gan Xu. “SVM-based incremental active learning for user adaptation for online graphics recognition system”. eng. In: *Proceedings of 2002 International Conference on Machine Learning and Cybernetics* 3 (2002), pp. 1379–1386. DOI: 10.1109/ICMLC.2002.1167432.
[27] Ahmad Kamal. “Subjectivity Classification using Machine Learning Techniques for Mining Feature-Opinion Pairs from Web Opinion Sources”. und. In: (2013), p. 10.
[28] Lara Mauri and Ernesto Damiani. “Estimating Degradation of Machine Learning Data Assets”. eng. In: *Journal of Data and Information Quality* 14.2 (2022), p. 9. ISSN: 19361963, 19361955. DOI: 10.1145/3446331.
[29] Wenting Qi and Charalampos Chelmis. “Improving Algorithmic Decision-Making in the Presence of Untrustworthy Training Data”. eng. In: *Proceedings - 2021 Ieee International Conference on Big Data, Big Data 2021* (2021), pp. 1102–1108. ISSN: 26391589. DOI: 10.1109/BigData52589.2021.9671677.
[30] Xudong Kang et al. “Detection and Correction of Mislabeled Training Samples for Hyperspectral Image Classification”. eng. In: *Ieee Transactions on Geoscience and Remote Sensing* 56.10 (2018), pp. 5673–5686. ISSN: 15580644, 01962892. DOI: 10.1109/TGRS.2018.2823866.
[31] C. E. Brodley and M. A. Friedl. “Identifying Mislabeled Training Data”. und. In: (2011). DOI: 10.1613/jair.606.
[32] *The Index Project - Nominate*. Accessed: 2022-11-30. URL: https://www.theindexproject.org/award/nominate.
[33] *Sustainable Development Goals*. Accessed: 2022-11-30. URL: https://sdgs.un.org/goals.
[34] *The Index Project - Award*. Accessed: 2022-12-05. URL: https://www.theindexproject.org/award.

[35] *COVID-19: Design Responds*. Accessed: 2022-11-30. URL: https://theindexproject.org/post/covid-19-design-responds.
[36] *Fastai - Text transfer learning*. Accessed: 2022-12-04. URL: https://docs.fast.ai/tutorial.text.html.
[37] *Fastai - Metrics*. Accessed: 2022-12-18. URL: https://docs.fast.ai/metrics.html.
[38] *Intersection over Union (IoU) in Object Detection and Segmentation*. Accessed: 2022-12-18. URL: https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/.
[39] *Jupyter Notebooks*. Accessed: 2022-12-03. URL: https://jupyter.org/.
[40] *PyTorch*. Accessed: 2022-12-03. URL: https://pytorch.org/.
[41] *Practical Deep Learning for Coders*. Accessed: 2022-12-03. URL: https://course.fast.ai/.
[42] *Selenium*. Accessed: 2022-12-03. URL: https://pypi.org/project/selenium/.
[43] *Firefox*. Accessed: 2022-12-03. URL: https://www.mozilla.org/en-US/firefox/new/.
[44] *Translators*. Accessed: 2022-12-03. URL: https://pypi.org/project/translators/.
[45] *Google Translate*. Accessed: 2022-12-03. URL: https://translate.google.com/.
[46] *Matplotlib*. Accessed: 2022-12-03. URL: https://matplotlib.org/.

Technical
University of
Denmark

Richard Petersens Plads, Building 324
2800 Kgs. Lyngby
Tlf. 4525 1700

www.compute.dtu.dk