POLITECNICO DI TORINO

College of Computer Engineering, Cinema and Mechatronics

Master's Degree Thesis

Use of Artificial Intelligence Techniques to Improve the User Experience With Waste Recycling

Supervisors
Bartolomeo Montrucchio
Moreno La Quatra
Antonio Constantino Marceddu

Candidate
Thomas Avare

April 2025

Summary

In the wake of escalating global environmental challenges, sustainable waste management practices have emerged as a major concern. The need for efficient recycling strategies has been evolving. Recycling has become more and more popular and adopted amongst most of people lives. There is a need to find easier and more attractive ways to encourage people to recycle more and more accurately.

In this context, recycling practices with state-of-the-art natural language processing technologies present an opportunity to improve the way we approach waste management. For the past 10 years, the technological advancements in Deep Natural Language Processing (DNLP) have been a huge milestone in the field of Artificial Intelligence (AI). Since 2017, with the paper "Attention is All You Need" [1], DNLP has progressed at a rapid rate, leveraging for broader and more advanced applications. Every day, new technologies, new models, and papers give the opportunity to explore new original ways to solve various problems of all kinds.

This thesis embeds the development of a new recycling approach in a more intuitive way. It combines different state-of-the-art and recent models to create a speech-to-classification pipeline consisting of two different steps: a speech-to-text phase and then a classification phase. We create a new "user experience" of throwing waste and recycling. This thesis also presents the different steps of creating datasets to train and evaluate the models and the pipeline using different models, from text datasets in English and Italian to audio datasets in Italian.

Acknowledgements

I would like to thank everyone I worked with during this project: Antonio and Moreno, who guided me before, during, and after this thesis and helped to highlight my work with the writing of a scientific paper based on this work; everyone at ReLearn, who trusted me and helped to guide the work in the right direction.

I would also like to thank my parents and friends who listened to me throughout this project; even though they did not always understand, they still supported me, academically and mentally.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Waste management refers to the collection, transportation, processing, and disposal of waste materials produced by human activities. This complex system is a crucial aspect of modern life, as the quantity of wastes generated continues to rise over time, playing a vital role in mitigating the environmental impact of our consumption habits. There is typically 5 types of waste management, varying in sustainability and ecological impact:

1. **reduce** : Cut down on the initial resources needed to lessen eventual waste
2. **reuse** : Prolong a product's lifespan to get the most out of the material
3. **Recycle** : Turn waste into new products as part of an effective circular economy model
4. **Recovery** : Use waste to generate energy, extracting value from it that can't be repurposed
5. **Landfill** : A last resort for waste disposal, safely ensuring it's contained and doesn't leak into the environment

Effective waste management requires a collaborative effort. Individuals can contribute by actively participating in recycling programs, reducing waste generation through mindful consumption, and advocating for policies that promote waste reduction and resource conservation. Communities can organize educational initiatives and implement waste collection systems encouraging proper sorting and disposal.

Governments have a significant role to play by establishing regulations, promoting sustainable practices, and investing in infrastructure. The European Union exemplifies this approach by setting ambitious goals for waste management. The aim is to significantly reduce the amount of municipal waste ending up in landfills. By 2035, Member States are required to take necessary measures to ensure that no

more than 25% of the total municipal waste generated is disposed of in landfills[1]. In some countries, like Bulgaria and Romania, more than 60% of the municipal waste is disposed of in landfills, and in Europe, 40.8% of the waste is recycled. Actually, Italy is the leading country in Europe in waste recovery by recycling[2].

Achieving these goals depends on the active participation of citizens. However, navigating the complexity of waste disposal can be challenging. Confusing regulations and unclear labeling can lead to incorrect waste sorting, depending on the effectiveness of waste management programs. This highlights the need for clear communication, educational and intuitive initiatives to encourage individuals to make accurate choices about waste disposal.

Furthermore, waste management must evolve to address waste's growing volume and complexity. Technological advancements offer promising solutions. By working together, individuals, communities, and governments can create a more efficient and sustainable waste management system, minimizing our environmental footprint and ensuring a cleaner future for later generations.

Several companies are also taking action to promote waste management at different levels. ReLearn, an Italian startup that combines technology and dynamic education with a data-driven approach to help people adopt conscious behavior in terms of waste production, is one of these, as it is contributing to the EU Sustainable Development Goals (UN-SDGs) 9, 11, 12, 13[3]. NANDO, ReLearn's Artificial Intelligence (AI) system, turns a regular bin into a smart bin through an AI-powered sensor and a screen that allows user interaction. It enables the collection of accurate data on the amount of waste produced and facilitates sustainability reporting, reducing waste-related CO_2 emissions and empowering the community.

In this thesis, we present a speech processing pipeline that allows users to interact with NANDO using their voices, thus improving user experience. It follows three steps:

- The user approaching the smart bin can ask where to dispose of a waste object.
- The system processes the user's voice and identifies the object category.
- The system provides an answer to the user with the target disposal bin.

The steps above have been achieved using state-of-the-art machine learning models and specific datasets created for addressing this task.

1.2 ReLearn

This master thesis was proposed on behalf of ReLearn and contributed to making this project real. ReLearn was founded in January 2021 based on the desire of a

[1] http://data.europa.eu/eli/dir/2018/850/oj

[2] https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Waste_statistics

[3] https://www.undp.org/european-union/sustainable-development-goals

group of university colleagues to harness their academic and professional skills to drive sustainable innovation.

ReLearn's smart sensor, NANDO, is the first waste monitoring tool to constantly monitor waste production with professional reporting while spreading sustainable practices to the community engagingly.

NANDO's main focus is to bring a waste-free lifestyle into everyone's daily routine: ReLearn is the first company able to monitor the waste directly in the place in which they are produced and, at the same time, educate the users. A problem cannot be fixed if it is not monitored.

ReLearn installs the small IoT sensor on a customer's existing bins and then uses artificial intelligence to collect data on the amount and type of waste produced daily.

- MONITORING AND REPORTING: NANDO reports the quantity and quality of waste through precise data on a waste monitoring dashboard, creating a professional report according to the GRI 306 standard. Customers can also monitor their live recycling rate, recycling trends, and the filling level of the bins.
- EDUCATION: NANDO spreads sustainable practices to achieve the best recycling rate in an office space. A tablet helps to identify which bin the waste should go to depending on waste disposal regulations in the region. Moreover, it constantly communicates with users: it is able to understand and warn users if they have thrown the waste in the correct container or correct them if they have made a mistake!
- IMPROVEMENT: Thanks to this monitoring, reporting, and education process, NANDO guarantees an increase in the company's recycling rate of +58%. If the company reduces its environmental impact and carbon footprint, it will only do so because its community is acting responsibly.

Chapter 2

State of the Art

The designed speech processing pipeline consists of several stages:

- voice acquisition,
- Automatic Speech Recognition (ASR in the following),
- an optional automatic translation,
- text classification,
- and response to the user.

Each pipeline stage leverages state-of-the-art models and techniques that have shown promising results in recent studies, particularly with the Transformers architecture.

2.1 Transformers

Transformers are a type of model architecture introduced in the paper "Attention is All You Need" [1] in 2017, considered an inflection point in modern Natural Language Processing. They have since become the foundation for most (if not all) state-of-the-art Natural Language Processing/Understanding models and close fields, such as automatic speech recognition and generation. They are based originally on an encoder/decoder architecture, similar to previous architectures, and introduce the attention mechanism.

Figure 2.1 depicts a typical and simplistic encoder decoder architecture typically used in sequence-to-sequence tasks such as translation. It is interesting to notice the encoder and decoder are note inter-dependent but can be used independently depending on the context of the task as we'll explain later.

The encoder and decoder are both transformers-based. All transformers are made of the same components:

- A *tokenizer*.

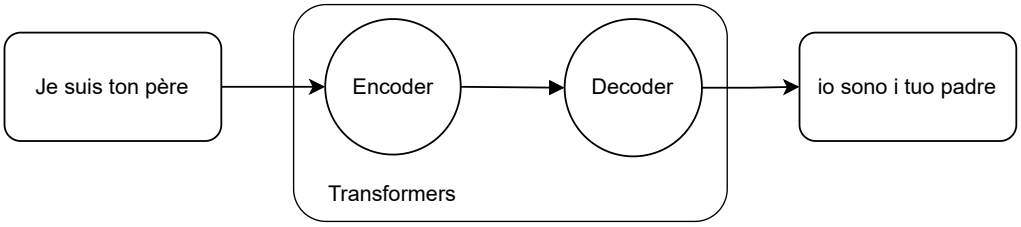


Figure 2.1: Encoder-Decoder Architecture

- An *embedding layers*.

- The *transformer layers*, composed of *feed-forward network* and *multi-head attention*.

- An *un-embedding layer*.

2.1.1 Embeddings

The *tokenizer* role is to map the discrete text input data into numerical data, such as one-hot vocabulary encoding for instance. This tokenized vector is then passed through the *embedding layer*.

The role of the *embedding layer* is to produce *embedding vectors*. An embedding vector represents a word, also designated as *token*, in a new continuous space with a certain number of dimensions d_{emb}. This vector alone does not carry much semantic information since it is not contextualised within the input sequence. To solve this problem, the transformer model performs what is called a Positional Encoding. It consists of adding information about the word regarding the input sequence, typically its position in the input sequence.

The output of the decoder is the passed through an *un-embedding layer*. The role of the un-embedding layer serves as the opposite of the embedding layer, taking a vector and transforms it into a probability vector over a set of predefined tokens (like the input vocabulary). So the role of the un-embedding is to map continuous vectors representation back to discrete tokens.

2.1.2 Core Principle of the Transformers

The *encoder* and *decoder* themselves are composed of 2 alternating layers: the *attention layer* and the *feed-forward layer*. These two layers are what really characterise the transformer architecture.

Attention Layer

Attention in Machine Learning is the process to calculate the importance of inter-dependencies between components, in our case tokens, in a sequence. This task

encompasses the principal principle of Natural Language Processing and the attention layer has revolutionised our capability to construct model with understanding capacity better than ever.

The Attention Layer corresponds to stacked *Attention Head*, also called *scaled dot-product units*. One attention head is composed of three different trainable weight matrices: the query W^q, key W^k and value W^v weights. To compute the attention, we want the query Q, key K, and value V matrices, obtained by multiplying the weight matrices with corresponding X_{query}, key X_{key} and value X_{value} vectors obtained from the input vector. These vectors are build depending on the specific usage of the transformers. We speak of self-attention when all the queries, keys, and values vectors are extracted from the same sequence. For instance, with an encoder-only model such as BERT, all the input vectors are a same input vector X due to the self-supervised nature of the training. So we would have these 3 matrices:

$$
\begin{aligned}
Q &= XW^q \\
K &= XW^k \\
V &= XW^v
\end{aligned}
$$

The (self-)attention score is obtained by performing a *Scaled Dot-Product Attention* transformation. The query and key are combined using the dot product, computing the similarity, or attention scores, between queries and keys. The whole is then scaled according to the dimension d_k to prevent large values. The whole is then transformed in probabilities by applying the softmax function, resulting in the *Attention weights* between the inputs. Each coefficient of this matrix corresponds to a similarity score between the query and keys. The whole is then multiplied by the value matrix to obtain a weighted sum.

$$
\boxed{Attention(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V}
$$

These transformations aim to extract features with "high attention" by mimicking the retrieval of a value for a query on some keys, focusing on specific passage of the input.

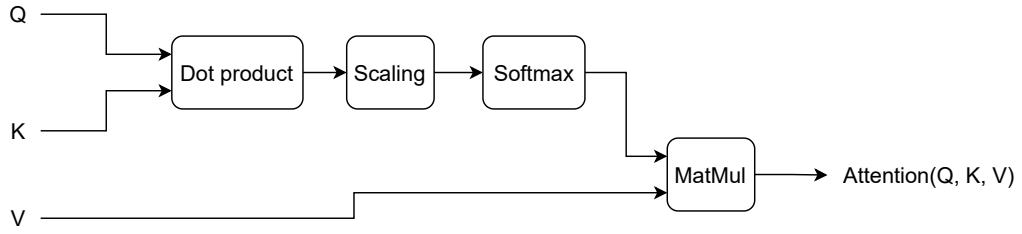


Figure 2.2: Attention Head

Figure 2.2 depicts the flow chart for the scaled dot product attention. It is important to remember that Q, K and V are matrices while the attention score is a weighted sum, so a number.

A stack of multiple attention heads is called a *multi-head attention*. Each attention head focuses on a specific portion of the input and has a different set of query, key, and value weights matrices (W^q, W^k, W^v), allowing the mapping of different kinds of relationships from the same input. The output of each attention head is then concatenated into a new vector. Each multi-head attention has a learnable projection matrix W^o, allowing the concatenated output to have the right dimension.

$$\boxed{MultiHead(Q, K, V) = [head_1, \ldots, head_h]W^o}$$

Unlike the scaled dot-product attention, the multi-head attention output is a matrix and they shouldn't be confused.

There are many advantages to using multiple-head attention. They can capture a broader range of dependencies and nuances in the input data, and it has been proven that attention head has better results for a wide range of applications compared to previous architectures. Also, because there is no recurrence in attention heads and each one is independent from one another, they can be processed in parallel during model training and inference, allowing high-speed computation. Leveraging these properties has enabled the construction of more prominent and more powerful models scaling up two tens of billions of parameters and trained on behemothic datasets ranging up to trillions of tokens !

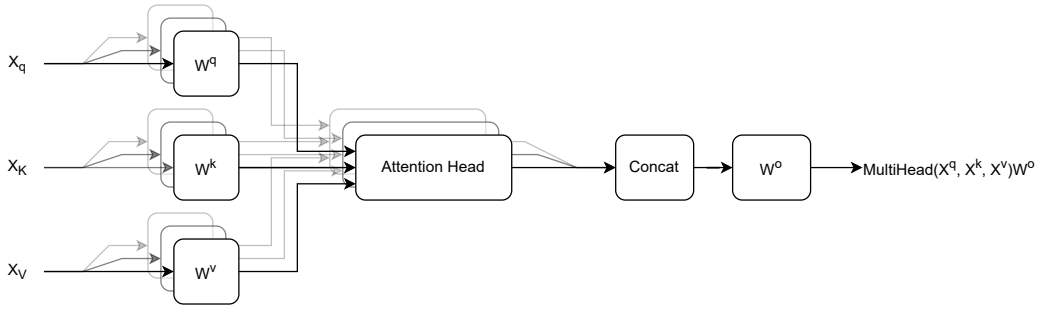


Figure 2.3: Multi-head Attention

Figure 2.3 depicts the flow chart of the multi-head attention. The parallelisation appears clearly, since most of the computation appears in the attention head, this architecture is ideal for building large models with less computation overhead as previous architectures.

Feed-Forward Layer

The feed-forward layer, more commonly established as *Convolutional Neural Network* (CNN), is one of the two classic types of neural network. This architecture characterised by an uni-directional information flow from the input to the output.

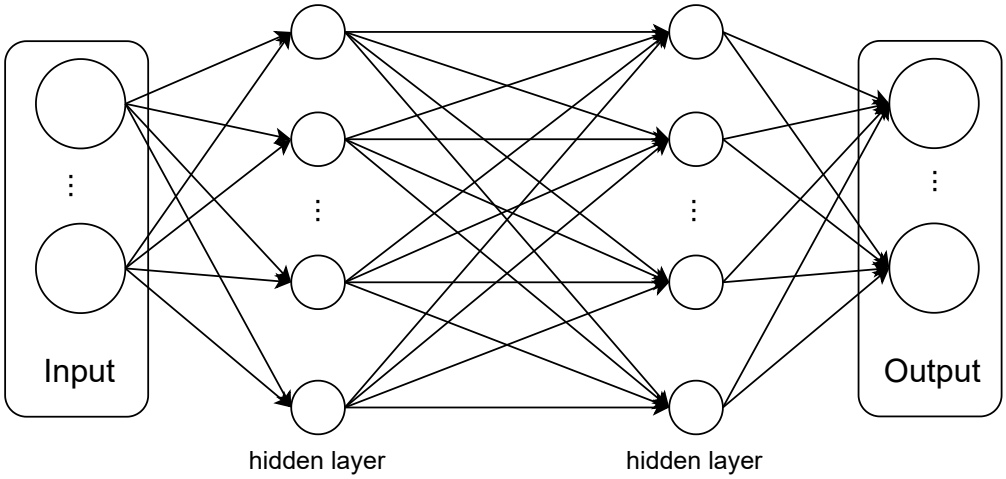


Figure 2.4: Feed-Forward Architecture

Figure 2.4 represents a simple CNN characterised by nodes and links. The input is a vector of dimension d_{emb}, typically the same dimension as the embedded input vector. The network is composed of successive vertical layers called the *hidden layers* of various dimension d_{layer}. These successive layers are fully connected, meaning that a node is connected to all the nodes in the next layer. Each node value is the weighted sum of the previous node values. This weighted sum is then passed through an *activation function*. The goal of the activation function is to introduce non-linearity in the output of the neurons, thus allowing to solve non-trivial problems. In the original Attention paper, the activation function is the *ReLU* function, lesser-known as *Rectifier Linear Unit*, which corresponds to a slope if the input is positive and zero otherwise.

$$\boxed{ReLU(x) = x^+ = \max(0, x)}$$

The output layer/vector of the feedforward network is usually passed through a softmax function to obtain probabilities. For a vector $\mathbf{z} = (z_1, \ldots, z_n)$ corresponding to the output of a feed-forward network for a classification with n classes, the i-th component softmax layer output corresponds to:

$$\boxed{\sigma_i(z) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}}$$

The whole network is trained using the backpropagation algorithm, a gradient-based algorithm used to modify the network weights for each input-output pair to minimise a *Loss function*, or *Cost function*. This function evaluates the difference between the network output and the actual output for a single input/output pair. In other words, the algorithm minimises the cost function with respect to the real values, called *ground truth*, of the training data.

Encoder/Decoder Architecture

The encoder is a stack of identical layers composed of alternating multi-head attention (Figure 2.3) and feed-forward layers (Figure 2.4). Its goal is to process the embedded input sequence and transform it into a fixed-size representation, capturing the semantic information of the input. The output vector is often called context representation or latent representation, which is a dense representation of the input.

The decoder is also a stack of identical layers composed of alternating multi-head and feed-forward layers. In a decoder architecture, such as GPT, the input is tokenized and passed through an embedding layer applying positional token embedding before feeding it to the alternating layers of the transformers. The output vectors are generated one after another by predicting the next element of the output sequence, conditioned by the previous outputs and context vectors.

The encoder and decoder work well together on many tasks, such as translation, text summarisation, or text generation. The decoder follows the encoder and takes context vectors from the encoder output to generate the output sequence. They can also be independently used for other tasks that wouldn't require both elements. For instance, BERT [2] architecture is solely based on the encoder part, such that its output is a contextual vector embedding of the input sequence (and not only a token embedding). This architecture works best for tasks that does not require the generation Natural Language, such as sentiment analysis, also called text classification or question answering.

Transformers revolutionised language processing with the self-attention mechanism allowing the models to weigh the importance of different input parts, overcoming the limitations of previous models in Natural Language Processing, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs). This parallel processing capability enables faster training of large models, with up to billions of parameters, on large datasets, with up to trillions of words. During the past few years, we have witnessed the rise of what we call *Large Language Models*, or LLMs. As suggested by its name, they are models with a very large number of parameters trained on massive amounts of data, allowing these models to enable human-like performances on a wide range of complex tasks without further training or fine-tuning, thus the term *pre-trained*. The goal is to have a unified architecture for many tasks.

2.2 Automatic Speech Recognition

Automatic Speech Recognition (ASR), also called *Speech-To-Text (STT)*, is the task of converting spoken language into text. The first research on speech recognition dates back to the 1950s and has never stopped since. This field has hugely benefited from the advances in Deep Learning and, more recently, in Natural Language Processing, leading to significant improvements in several downstream tasks [4, 5], including ASR.

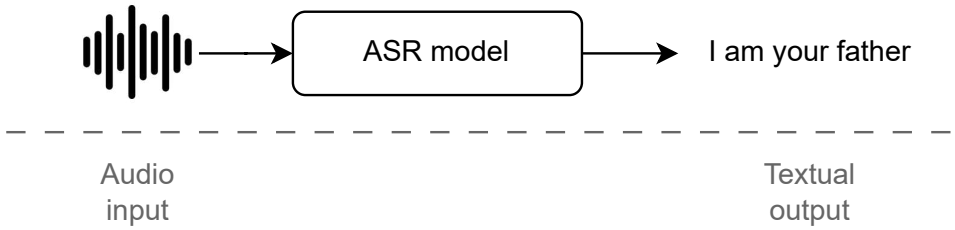


Figure 2.5: ASR pipeline

Those models usually have a encoder-decoder architecture due to the sequence-to-sequence nature of the task and are trained on large-scale datasets, using self-supervised learning [6, 7] to learn audio representations. The encoder produces previous context vectors, but on audio snippets rather than on text snippets, that can be used for downstream tasks such as speech classification, or supervised learning [8] to directly predict the text from the audio signal. In this thesis, we will not use audio representations; thus, we will not develop this aspect of ASR further.

In this part, we will elaborate on the technologies and principles used in *Whisper* [8], a multilingual model made by OpenAI with state-of-the-art performances on several ASR benchmarks. We will subsequently motivate this decision in Section 3.1.2. It is a transformer-based model composed of an encoder-decoder architecture. Since ASR and various NLP tasks share many similarities, this architecture is well-suited for these tasks. We can note that historically, both speech-to-text and language processing have often benefited from the same technologies.

Audio representation and encoding

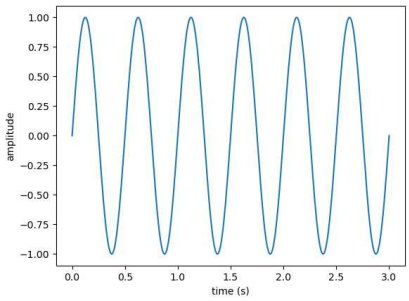
An audio signal is simply the addition of different single frequencies, in *Hertz (Hz)*, and their associated amplitude in *Decibels (dB)* over time, such as shown in Figure 2.6.

The encoder input is a pre-processed representation of the audio called *log-magnitude Mel spectrogram*. Introduced in 1937, A *Mel*, after Melody, is a perceptual scale designed to reproduce the human perception of frequencies. The frequency describes the number of vibrations of a wave per second, while the pitch measures the (human) perception between low and high frequencies. So the formula between *Hz* and *Mel* is empirical, and the most common one is given by:

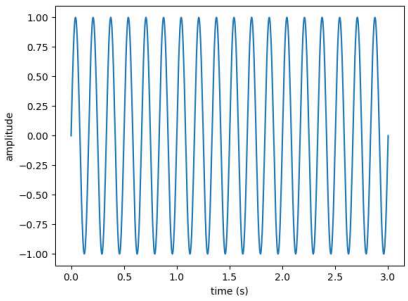
$$\boxed{m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)}$$

This unit measure is specifically valuable for Deep Learning applications because ASR aims to mimic how humans process speech.

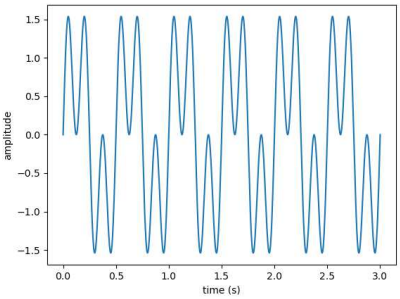
Because the signal varies over time, it is depicted as a *non-periodic* signal making it hard to analyse. To solve this issue, the signal is subdivided such that each division can be approximated as a periodic signal. Periodic signals have been widely studied for centuries within the field of signal processing, tracing back to



(a) A signal of amplitude 1 and frequency of 2 Hz



(b) A signal of amplitude 1 and frequency of 6 Hz



(c) The sum of the previous signals

Figure 2.6: Representation of a signal decomposition

to numerical analysis principles from the 17th century ! Thus, it is easy to link a signal and the Mel scale with few mathematical tools.

A *Fourier transform* is a mathematical transformation to break a signal (and more generally a function) down into its frequencies, so into a sum of sine and cosine functions. The Fourier transform has been widely studied over time, and remarkably optimised algorithms exist to efficiently compute a Fourier transform of any function, leveraging its mathematical properties. Thus, we can extract each frequency and its associated amplitude by applying a Fourier transform to the signal subdivisions. It is then easy to convert the frequencies from Herz to Mel.

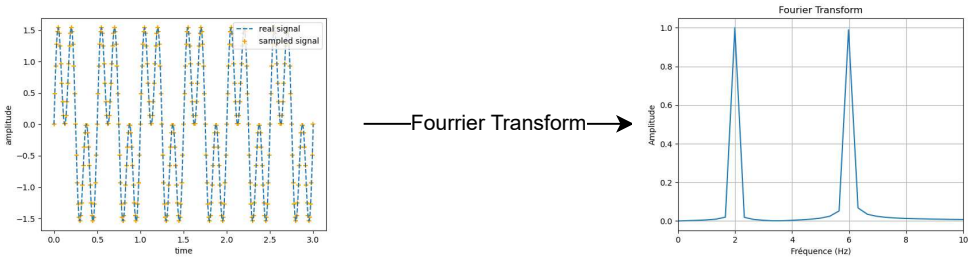


Figure 2.7: Representation of Fourier Transform

In the end, the Mel-spectrogram (Figure 2.8) is a visualisation of an audio signal over time, such that the y-axis represents the frequencies of each subdivision, and the amplitudes of each frequency are shown using a heatmap.

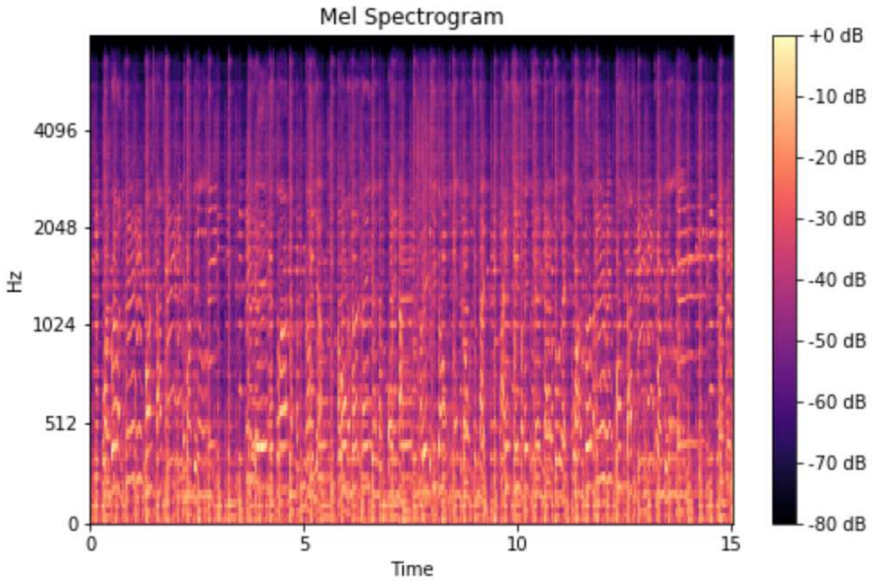


Figure 2.8: Mel-spectrogram example

The encoder consists of a feature extractor based on a Convolutional Neural Network (CNN) followed by a stack of transformer layers (multi-head attention) to extract high-level features from the input audio signal, as described in the previous Section.

The decoder is a transformer-based language model that predicts the transcription of the input audio signal given the features extracted by the encoder.

The Whisper model is available in different sizes, ranging from tiny to large, with the number of parameters increasing from 39M to 1.55B. While the large version of the model has shown state-of-the-art performance on several benchmarks, smaller versions may be more suitable for real-time applications due to their lower computational requirements and faster inference time.

2.3 Text Classification

Text classification consists in the task of assigning a label to an input text.

Similarly to the ASR task, transformer-based models have shown state-of-the-art performance on several text classification benchmarks. Text classification is a broad term that includes different tasks. For instance, we have *Natural Language Inference (NLI)*, where the goal is to establish the relationship between two inputs, or *Grammatical Correctness*, which is the task of assessing if a phrase is grammatically correct. Among the text classification tasks, we will focus on *Sentiment Analysis*, which corresponds to a classical classification task. It consists of assigning a label to a phrase according to a predefined list of classes.

To perform such a task, The encoder architecture is particularly appropriate. Indeed, after a phrase is fed into an encoder, it is transformed into a fixed-size vector, the embedding vector, capturing and carrying information about the phrase.

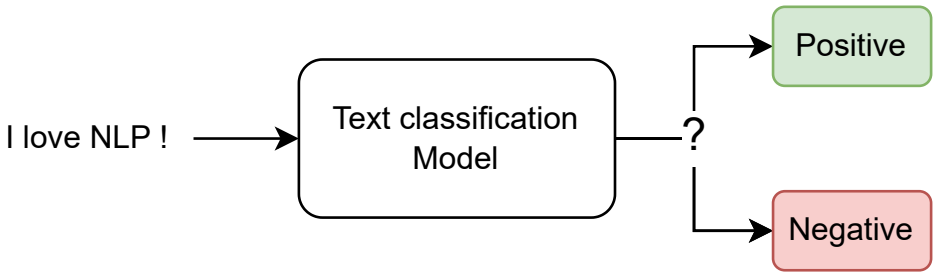


Figure 2.9: Sentiment Analysis

This embedded vector is then fed into classification layers, which is usually a Feed Forward Network. It has the same number of outputs as the number of classes. A *softmax* function is applied to transform its output in label probabilities.

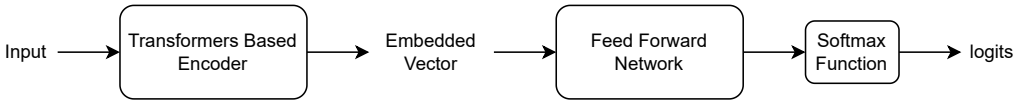


Figure 2.10: Text Classification Architecture Pipeline

For a vector $\mathbf{z} = (z_1, \ldots, z_n)$ corresponding to the output of a feed-forward network for classification with n classes, the i-th component softmax layer output corresponds to:

$$\sigma_i(z) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

BERT [2], standing for *Bidirectional Encoder Representations from Transformers*, launched by Google in 2018, is an encoder-only model designed for natural language understanding tasks and is among the first transformer-based models that have shown significant improvements in several Natural Language Processing tasks. This model has been widely used, and since we are using it, we are going to elaborate on some of its specificities.

It consists of a stack of transformer layers that analyse the input text in a bidirectional way, allowing the model to capture the context of the input text. Unlike the classical transformers model, and what was previously explained in 2.1, where the attention mechanism was uni-directional from left to right, giving only *left-context*, BERT proposes to capture context from left to right and right to left directions allowing to capture complex semantic information and solve some issues, like polysemy, so when a word changes meaning depending on the context.

It is pre-trained using self-supervised learning on large corpora and can be fine-tuned on downstream tasks even with a small amount of labeled data using supervised learning. The self-supervised learning consists of two tasks: *Masked Language Modeling (LM* and *Next Sentence Prediction (NSP)*. Masked LM consists of masking a certain percentage of the input and then predicting the masked tokens using the output of the transformer encoder. NSP is motivated by many downstream

tasks consisting of understanding sentence relationships, such as Natural Language Inference and Question Answering. The training consists of picking two phrases, A and B, and choosing if B is the sentence following A, such that 50% of the time B is the following sentence. Despite the tasks' simplicity, they have proven to be beneficial for some downstream tasks.

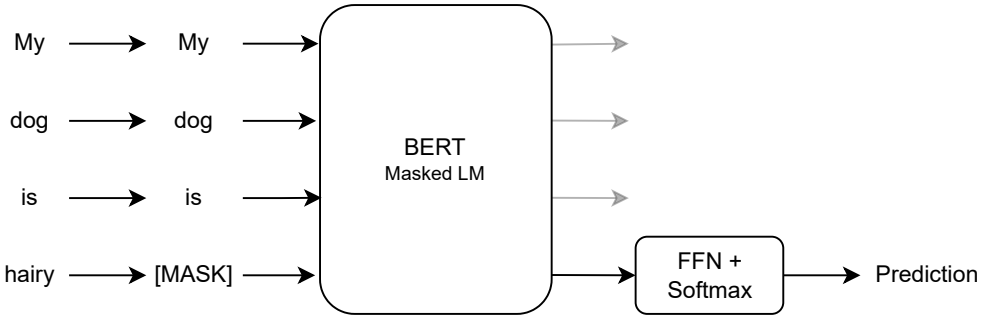


Figure 2.11: Masked Language Modeling

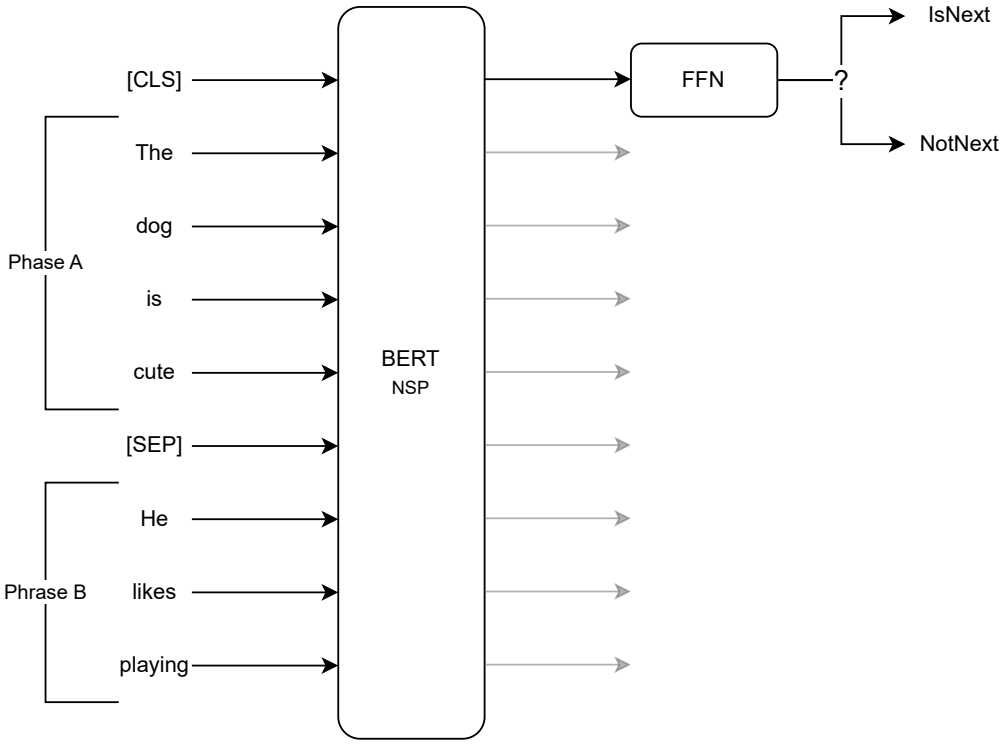


Figure 2.12: Next Sentence Prediction

RoBERTa [9] and DistilBERT [10] are some of the models that have been proposed as improvements to the original BERT model. RoBERTa is trained using slightly different pre-training objectives (e.g., dynamic masking) and more data, while DistilBERT is a distilled version of BERT that is trained using Knowledge Distillation to create a smaller model that keeps most of the performance of the

original model while being faster and more compact. The concept of Knowledge Distillation is shown in Figure 2.4.

Intent classification is a specific text classification task and a fundamental concept in Natural Language Processing that assigns a label to an input text representing the user's intention and plays a critical role in understanding the underlying purpose behind a user's text input. To perform

In waste management, the goal is to classify the user's request into one of the available recycling options. Several intent classification datasets have been proposed in the literature, each focusing on specific modalities (e.g., text, speech) and domains (e.g., voice assistants, customer service). MASSIVE [11] is a large multilingual dataset for text-based intent classification in the voice assistant domain. It covers 1M+ utterances in more than 50 languages. Speech-based language-specific versions of the dataset have been proposed in English [12] and Italian [13]. Other datasets have been proposed for specific domains, such as the ATIS dataset for flight booking [14] and the CLINC dataset for banking [15].

This work aims to propose a novel dataset specifically designed to classify user intents related to waste disposal. It covers 50 different classes of objects. To the best of our knowledge, no dataset is available for this specific task, and we believe it can be a valuable resource for the research community[1].

This dataset encompasses a comprehensive range of 50 different object categories, providing a valuable resource for researchers dedicated to advancing waste management through the power of intent classification.

By facilitating such interactions, intent classification can improve waste management practices. Users can receive clear and consistent information, leading to increased recycling rates and a more sustainable future. This novel dataset serves as a stepping stone towards achieving this goal, offering researchers the tools they need to develop intelligent systems that empower individuals to make informed waste disposal decisions.

2.4 Knowledge Distillation

Knowledge Distillation is a machine learning technique where a large model, called *teacher*, transfers its knowledge into a smaller and simpler model, called *student*. Its name comes from classical chemical distillation, consisting of extracting a component from a liquid mixture, which is a good analogy. It allows the student model to achieve similar performances while being much smaller and requiring fewer computational resources. These kinds of models allow for more accessibility for deployment and inference speed due to the much smaller size of the student model.

The proposed pipeline has substantial computational resources and inference time requirements. For this reason, we focus on DistilBERT, which allows for a

[1]Resources are available at: https://huggingface.co/collections/thomasavare/waste-classification-datasets-662f67e6ef20106ac99d87db

good trade-off between model size and performance.

The training process consists of the student model mimicking the teacher's behavior, resulting in a "distillation" of the teacher's knowledge. The student model is trained on the same dataset as the teacher but with an additional loss function encouraging to immitate the teacher behavior called *knowledge distillation loss* measuring the difference between the outputs of the teacher and student model. This loss allows the student model to learn from the data and the output of the teacher learning its behavior.

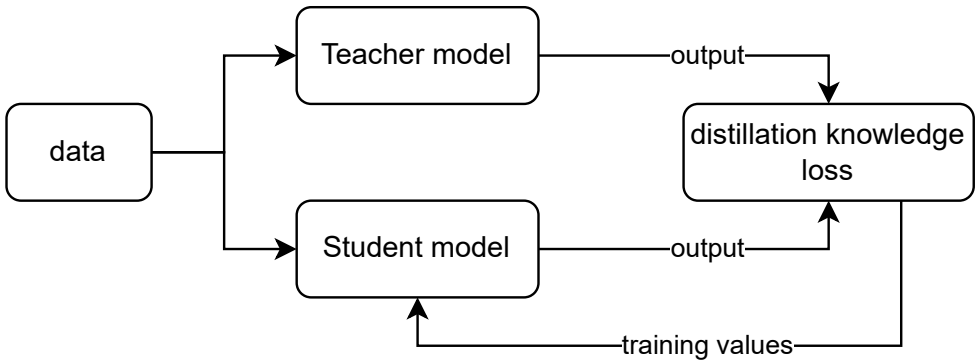


Figure 2.13: Knowledge Distillation training set up

This can also be used in the case of transfer learning to enhance the knowledge of smaller models using larger models. By successfully transferring data to a simpler model, we can develop more versatile and efficient models.

Chapter 3

Methodology

3.1 Establishing the appropriate pipeline

3.1.1 The pipeline itself

As discussed in Section 1.1, the thesis aims to establish a speech classification pipeline. Several approaches were considered, each offering its advantages.

Different pipelines were considered. The speech classification model pipeline, depicted in Figure 3.1, allows the use of fewer data and eliminates the speech translation step. Unfortunately, few models exist and do not allow for much flexibility. They often are only in English and are much bigger than needed.

A two-step pipeline, as depicted in Figure 3.2, which is composed of an ASR step followed by a classification model, seemed more appropriate. ASR models can reach human-like performances. Many ASR models are multilingual and can also perform *Automatic Speech Translation (AST)*. This would allow future products to be easily deployed in multiple countries. AST also allows the classification model to be mono-lingual. The transcripted/translated then passes through the classification model. Many classification models achieving human-like performances on the required tasks exist and are easily trained and deployable using cloud technologies. A two-step pipeline allows more versatility, but breaking down the tasks introduces more complexity and loss of information throughout the pipeline.

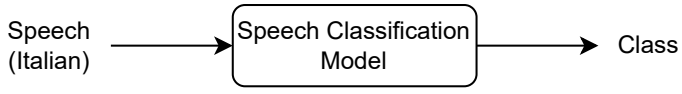


Figure 3.1: Speech Classification Model Pipeline

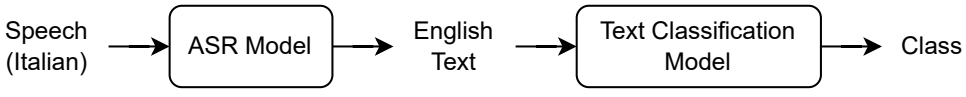


Figure 3.2: Two Step Pipeline

Both solutions offer advantages and flaws. We need to find a good trade-off between these solutions. This project is subject to real-world applications, so we can consider some specifications to lead our decision:

- The input has to be in Italian and/or English.
- The computation has to be quick for a better user experience
- The performances have to be as good as possible to avoid wrong recycling.
- The models must have a license suitable for commercial use (e.g., MIT, Apache ...).
- Cloud deployment vs embedded system.

After reviewing the state-of-the-art literature about available models, we decided that the two-step pipeline was more suited and versatile according to the previous specifications. The available models' quality, flexibility, and (multi-)language availability guided this decision. Taking leverage of pre-trained LLMs (and distilled LLMs) allows for less data because only fine-tuning for our domain-specific task is needed.

3.1.2 The speech-to-text step

After profoundly studying state-of-the-art ASR models and the available models for multi-language speech (especially in Italian) to English text, we decided that the best choice was Whisper [8]. It was introduced in December 2022 by OpenAI; it is multilingual, easily scalable, has state-of-the-art results, and is under the Apache-2.0 license.

Whisper is a multi-task speech recognition model. It performs English speech recognition, multilingual speech recognition, language identification, and speech translation. It differs from previous approaches by using a large and diverse dataset (680,000 hours of audio, about a third in foreign languages, collected from the web) and wasn't fine-tuned for any specific task. It has a straightforward architecture that is described *off-the-shelf*. An off-the-shelf architecture is a pre-designed architecture ready to be used like it is. The goal was to develop a single robust speech processing system that works reliably without needing specific fine-tuning to achieve high-quality results on particular distributions.

Whisper has similar results as previous models and sometimes outperforms them for certain tasks on various benchmarks on zero-shot evaluations. Usually, a benchmark comprises a training and test set, and we fine-tune the model on the training set before benchmarking on the test set; zero-shot evaluation consists of skipping the training part and directly evaluating the performances on the test set. For the purposes of our project, we focus only on specific translation tasks. It achieves a new state-of-the-art result at that time. It was outperformed by SeamlessM4T [16] from Facebook or AudioPaLM [17] and Google USM [18] by Google

X → en	High	Mid	Low	All
XMEF-X	34.2	20.2	5.9	14.7
XLS-R	36.1	27.7	15.1	22.1
mSLAM-CTC	37.8	29.6	18.5	24.8
Maestro	**38.2**	31.3	18.4	25.2
Zero-shot whisper	36.2	**32.6**	**25.2**	**29.1**
Google-USM	> 36.2	> 32.6	> 25.2	30.7
SeamlessM4T	N/A	N/A	N/A	34.1
AudioPaLM	N/A	N/A	N/A	**37.8**

(a) **X → english** BLEU scores for Speech Translation Performances on CoVoST2 Dataset (with language resource score).

ita → en	size	score
XLS-R	2B	34.9v
mSLAM-CTC	2B	37.3
Whisper Large v2	1.5B	30.9v
Seamlessm4T	2.3B	39.97
AudioPaLM	8B	44.3

(b) **Italian → English** BLEU scores for Speech Translation Performances on CoVoST2 Dataset.

In Tables 3.1a and 3.1b, and in the rest of this thesis, information in gray indicate that the model were released after the end of the project and are here as a reference to show the insane improvement pace transformers have started.

In Table 3.1b, we can see that Zero-shot whisper is outperformed in Italian, but it outperforms existing models (Table 3.1a) on *CoVoST2* [19] in the overall, medium, and low resource settings, corresponding to the number of resources in specific languages, but still moderately under-performs on high-resource languages compared to prior (and posterior obviously) directly supervised work.

Another advantage is that different model sizes are available and easily deployable. In fact, they trained similar architecture models in different sizes using the same dataset and hyperparameters to produce models with similar behaviors. In the end, we chose to use the base or small one in CPU-only cases; these models are very effective even tho they are smaller, but they are also much faster than the bigger models (Table 3.2). In the context of cloud-based deployment, If we have access to GPU units, we can take advantage of the parallelization of the transformers-based models and use more prominent alternatives for better results and similar inference times.

The smaller models do not have similar performances as the bigger and newer models, but smaller models with such performances and versatility do not seem to exist. It is also easily accessible, deployable, and tunable using HugginFace platform, libraries and cloud technologies. In conclusion, these arguments guided our choice towards Whisper, making it the best opportunity among the other candidates.

Whisper size	Number of parameters	BLEU Score
Tiny	39M	5.3
Base	74M	11.3
Small	244M	17.8
Medium	769M	21.6
Large	1.55B	23.5
Large v2	1.55B	23.6

Table 3.2: Number of parameters and overall BLEU score of whisper versions.

3.1.3 Classification task

The classification task consists of classifying a phrase among 50 different classes of wastes, such as aluminum cans, cigarette packs, paper plates, etc... (see Table 5.1 for all the classes). There are many classes because, depending on the geographical situation of the use, some wastes do not go in the same recycling bins. The classic recycling bins are organic waste, paper and/or plastic, sometimes glass and indifferent. All the waste classes are then mapped to a specific recycling bin according to the location rules. For instance, in Milano and Roma, the coffee paper cup goes in the paper bin, but in Torino, it goes in the indifferent bin.

The approach to choosing an appropriate Language Model for the text classification was similar to previous ones, i.e., we studied the latest state-of-the-art pre-trained models. A good thing that wasn't available in automatic speech translation, unfortunately, are benchmarks used to compare most of the recent models. The benchmarks chosen are called GLUE [20] and SuperGLUE [21]. They consist of a collection of evaluations such as single sentence tasks, similarity and paraphrase tasks, and Inference tasks. These tasks allow the testing of different tasks and give an average score for model performance in various aspects. Most of the benchmarked models, especially the top of the leaderboard, are not always public or under licenses that do not allow commercial use and are too big. In our case, we will focus on one particular task because the others are not essential to the development, even if some models are really incredible and outperform human performances. The task is SST-2, which stands for Stanford Sentiment Treebank and consists of sentences from movie reviews and human sentiment annotations. The task is to predict the sentiment of a given sentence (positive/negative) and is similar to the classification task we need to do.

Model Name	Number of Parameters	SST-2 Score
Turin UlRv6	4.6B	97.5
T5-11B	11B	97.5
BERT	340M	92.55
RoBERTa	355M	96.7
DistilRoBERTa	82M	92.5
DistilBERT	66M	92.3

Table 3.3: Various Models Results on SST-2 task

After Analysing the leader of these benchmarks, we came up with different possible solutions, including T5 [22], BERT [2], DistilBERT [10], RoBERTa [9] and DistilRoBERTa. Considering different criteria such as number of parameters, robustness, easily trainable, and inference speed, the final choices were DistilBERT and DistilRoBERTa; we ended up using DistilBERT because there was no real improvement using DistilRoBERTa and the model implementation was harder than DistilBERT with the HuggingFace tools.

Over the past years, Large Language Models (LLM) have tended to grow bigger and bigger. For instance GPT-2 [3] has 1.5 billion parameters, GPT-3 [23] has 175 billion parameters and GPT-4 [24] has an astonishing 1.7 trillion parameters (all three models were developed and pre-trained by OpenAI), Megatron-Turing NLG [25] has 530 billion parameters, BARD [26] has 137 billion parameters and PaLM 2 [27] has 340 billion parameters.

DistilBERT aimed to construct a model much smaller than BERT while preserving its performance. Distillation techniques are already known but were mainly used for task-specific models. DistilBERT innovates with distillation during the pre-training phase. At last, it was possible to reduce the size of BERT by 40% (BERT-base: 110 million parameters, BERT-large: 345 million parameters, DistilBERT: 66 million parameters) while retaining 97% of its understanding capabilities and being 60% faster.

Model	Average Score	SST-2 Score
BERT-base	79.5	92.7
DistilBERT-uncased	77.0	91.3

(a) BERT and DistilBERT on GLUE benchmark

Model	number of parameters	Inference time
BERT-base	110M	668s
DistilBERT	66M	410s

(b) BERT and DistilBERT inference time on full pass on SST-B task

More precisely, we used the "distilbert-base-uncased" model available on Hugging Face[1]. It was important to use the uncased version so it does not consider the uppercase letters.

3.2 The datasets

Another important aspect of Artificial Intelligence, and the most significant part of this work, is finding the correct datasets. The biggest challenge encountered was that the assigned task had no public dataset available for both training/testing DistilBERT (similar to the SST-2 task) and evaluating the pipeline's performance. For both of these, we had to make our own ad-hoc dataset.

[1]Available here: https://huggingface.co/distilbert/distilbert-base-uncased

3.2.1 Training and testing the classification task

As explained in Section 2.3, the classification task consists of feeding the model a phrase in English and classifying it in one of the 50 waste classes available (see Table 5.1). At one point, it was discussed about creating an "other" class as a 51st class, but it was hard to find real-life applications, except for useless cases such as proper nouns or unusual wastes (i.e., nuclear wastes).

After searching for information on the datasets already available, we found that none of the ones already released had the characteristics we were interested in, so we had to create an ad-hoc one. It was created in different steps. The first step was to obtain different phrasing of how someone would naturally address the recycling bin. To broaden the possible number of combinations and variety of the future dataset, we asked some friends how they would phrase their demands. We ended up with 50 different phrasings (not including duplicates) in English and/or French; all the phrases were then translated into English. For instance, some sentences we got were "Where do I have to throw out my plastic bag?" or "I have a Starbucks, where do I throw it?". We extracted the main part from these phrasings and eliminated the actual waste. The demand would then look like, "Where do I have to throw out [article] [object]?" or "I have [article] [object], where do I throw it?". The article token takes either "a," "an," "my," or "the value.

For each of the 50 waste classes, we generated a list of different wastes, either by hand or sometimes using language models such as ChatGPT. We tried to make the repartition of the classes as balanced as possible (see Figure 5.1), but some classes are obviously going to be more crowded because sometimes it is hard to find objects for the class, and there are too many objects for one class.

To make our dataset, we just had to combine the previously obtained datasets by replacing our [object] token with the objects that we listed and their corresponding labels.

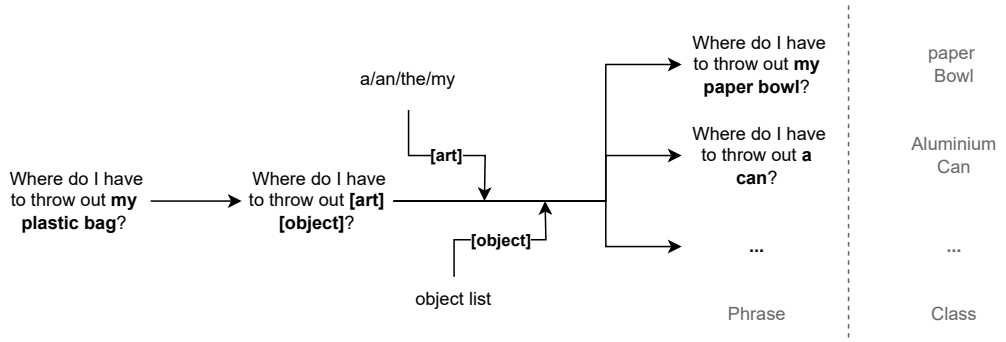


Figure 3.3: Data Creation Pipeline

We randomly selected phrases from the created dataset to make the train and test set. We chose only to use 80% of the total data created because it will be uselessly big. During the project, the datasets were modified to evaluate the optimal one. The two best-performing datasets consist of an 80%-20% (see Figure 3.4) train/test split and a 60%-20%-20% train/test/validation split (the second one

created for the purpose of the writing of a paper based on this work several months later [28]).

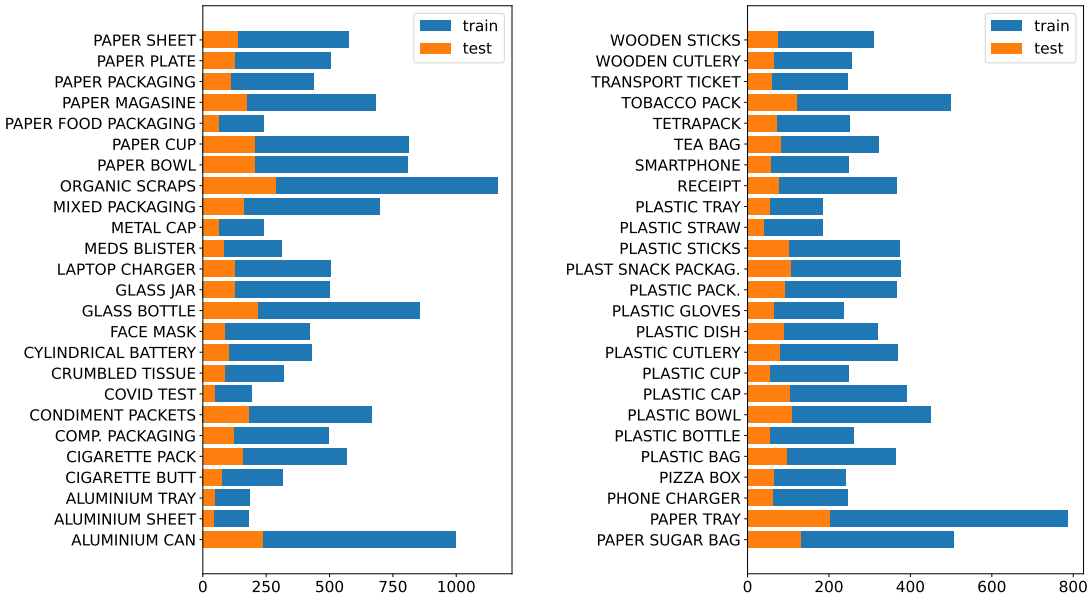


Figure 3.4: Number of samples per object classes in the first set

In the end, for the first dataset, we obtained a total of 21528 phrases in the training set and 5382 in the test set. Because we randomly selected the different phrases going into the different sets, the repartition of the classes should be similar between the original set and the train/test split (see Figure 3.4. For the second dataset, which aimed at balancing the classes, the training set comprises 16,146 sentences, while the validation and testing datasets contain 5,382 sentences each (see Figure 3.5).

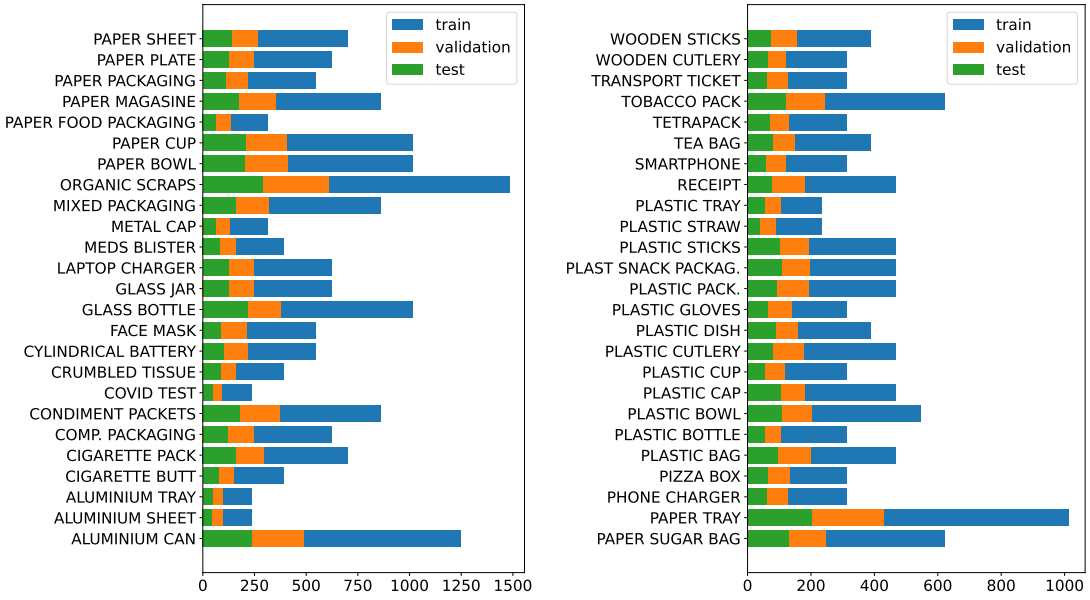


Figure 3.5: Number of samples per object classes in the second set

As a matter of fact, the datasets were modified several times due to balance issues and size issues. It was so unbalanced that the classification models was ignoring some of the classes. When a class is under-represented, the cost function optimizes it by only misclassifying it resulting in an overall better accuracy.

3.2.2 Giving more depth to the data

The rise of generative AI over the past few years has been marked by significant advancements in machine learning, natural language processing, and computer vision technologies. These advancements have led to the development of increasingly sophisticated generative models that can generate realistic and creative content. Generative AI continues to evolve rapidly and is expected to profoundly impact various industries and domains, including entertainment, healthcare, education, and beyond. As technology advances and models become even more powerful, addressing the ethical and societal implications will be essential while harnessing the creative and productive potential of generative AI. Generative AI has become so efficient and valuable that it is sometimes hard to notice if a person or a generative AI wrote it. For instance, the beginning of this Section was generated asking ChatGPT (GPT-3) to "detail the rise of generative AI over the past few years". This technology can be a real benefit and an enormous leverage for projects like this one; in our case, it can be helpful for the test dataset.

Using generative AI to modify a dataset for a text classification task can offer several advantages, which can help improve the quality and robustness of our model. Generative AI can generate additional data samples by creating new text that is similar to the existing dataset. This helps in data augmentation, which is particularly beneficial when there are limited labeled data. More data can lead to a better-performing model, as it exposes the model to a broader range of variations and examples. By generating new data samples, you can reduce the risk of overfitting. Overfitting occurs when a model becomes too specialized in learning from a limited dataset and performs poorly on unseen data. Generative AI can introduce diversity into the dataset, allowing the model to generalize better to new, unseen examples. Real-world datasets often contain noise, errors, and inconsistencies. Generative AI can help generate cleaner versions of the data or fill in missing values, making the dataset more suitable for training reliable text classification models.

The use of generative AI to modify a dataset for text classification tasks can enhance the quantity and quality of the data, improve model generalization, address class imbalances, and even strengthen privacy and security. It is a valuable tool for improving the performance and robustness of text classification models, especially in scenarios with limited or noisy labeled data.

In our case, we used ChatGPT to rephrase a thousand phrases from the 20% phrases that weren't used for the train/test splits. It gives more sense to some phrases; for instance, "Where do I throw the rest of my plastic knife?" becomes "Where should I dispose of my remaining plastic knife?". It also allows us to have more varied examples because of how the previous model was made. It allows to have a "zero-shot" dataset with different phrases and more advanced and diverse

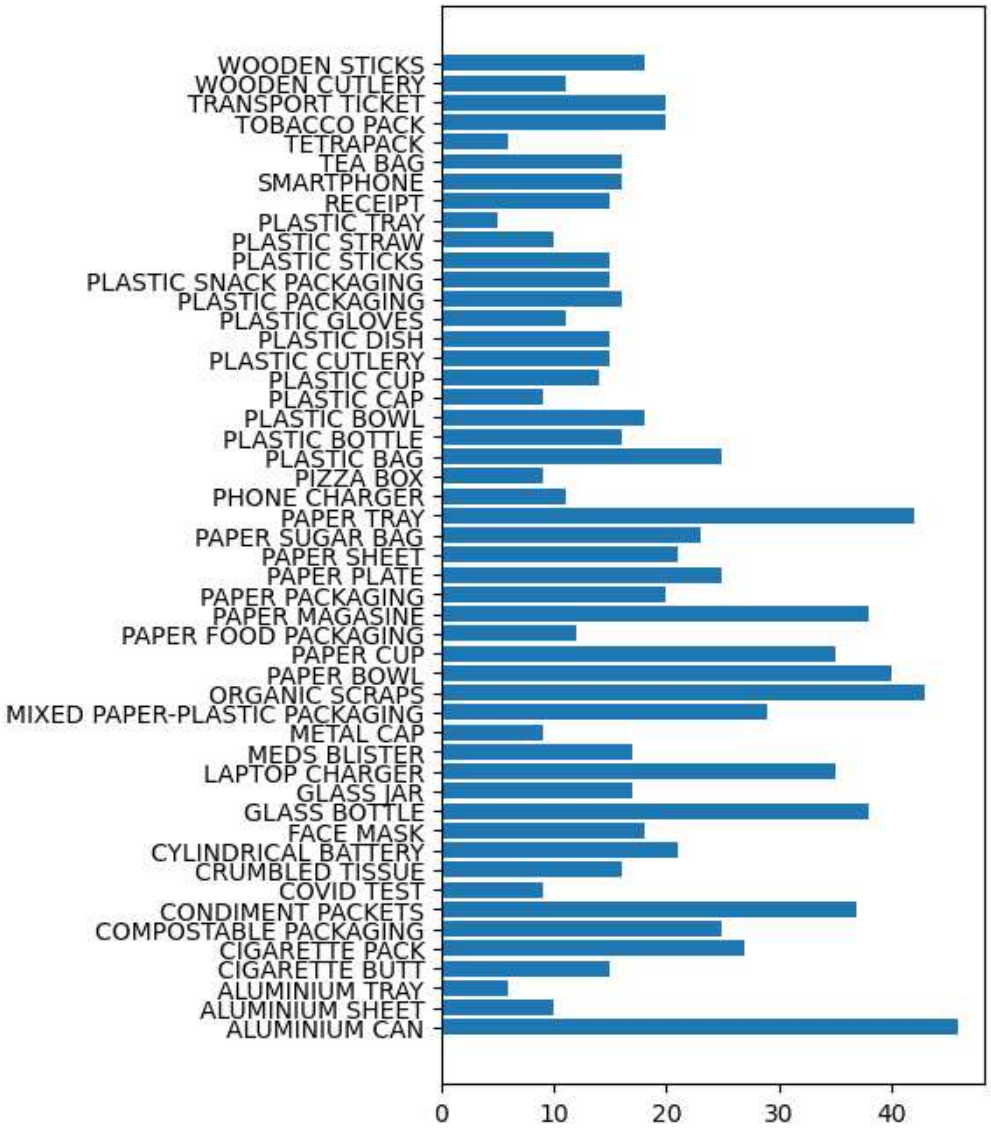


Figure 3.6: Repartitions of classes in the GPT modified set

to test the performance of the models. We did not modify all the data because it would have been a long process and not free. Another important point is that the model we are using, DistilBERT, is pre-trained and offers many advantages, such as that it does not need a lot of data and the variety of the data used has not to be enormous to make it domain-specific. We will see later that the performances achieved on the test set and the modified data are not very far apart from each other (see Figure 3.6).

Figure 3.8 is a detailed flow chart visually explaining how the original data is split to make the different datasets.

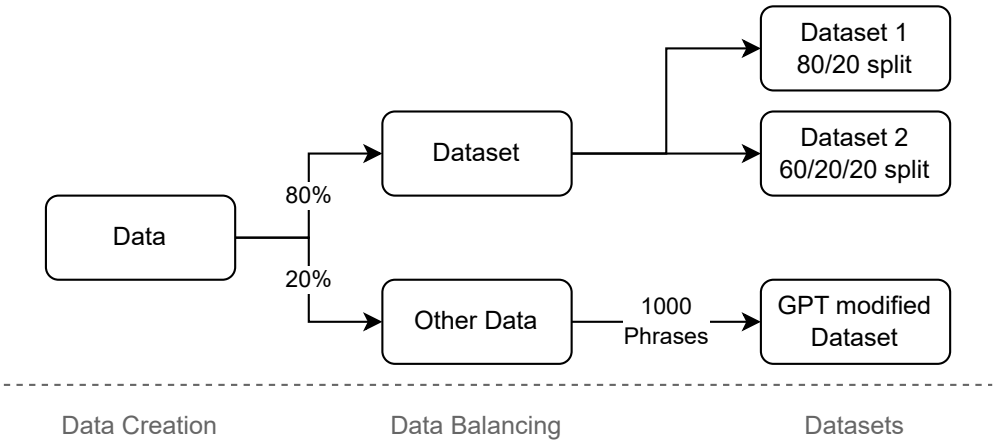


Figure 3.7: Text Dataset Creation Flow Chart

3.2.3 Evaluating the pipeline

The whole pipeline takes audio as input and infers the classification of this audio after the speech recognition task. To evaluate the pipeline, we need a dataset constituted by audios of phrases in Italian. Such a dataset, unfortunately, does not exist, so we had, once again, to make our own ad-hoc Italian audio set. To do so, we used a fraction of our English phrases test set. Using a translation language model, we translated 500 phrases from this set. At first, we used the "Helsinki-NLP/opus-mt-en-it" model [29] available on HuggingFace. It is a model made by researchers from Helsinki University; since it is available on HuggingFace, it was quickly implemented and used for translation. Unfortunately, its BLEU score (bilingual evaluation understudy) is around 35. This score is an evaluation algorithm used to measure the quality of a translation; it measures the similarity between a professional human translation and the translation score. The BLEU score can be between 0 and 100 (or 0 and 1). The best we can often achieve is around 60-70; even a human can often not achieve a score of 100. As a second translation model, we used the Deepl-API, promising better translation than any other translator on their website[2] and even better than the Google translate traductions models [30].

Model	BLEU score
Helsinki	0.41
Google	0.44
DeepL	0.45

Table 3.5: BLEU scores of different models for Italian to English translation.

Figure 3.5 shows the different BLEU scores computed for the Helsinki, Google translate, and Deepl models on the same dataset for English-to-Italian translations.

[2]Deepl website: https://www.deepl.com/en/translator

The dataset is composed of a thousand English and Italian traduction phrases extracted from the *Tatoeba project* [3]. The Tatobea project is a collection of English example sentences and their translations. Experimentally, on a tiny and "unknown" subset, DeepL seems slightly better than the rest.

After translating a subset of the test set, we used Bark [31], a transformer-based text-to-audio model created by SunoAI[4]. It creates realistic multilingual speech, including Italian. Eight speakers in Italian are available, allowing us to nuance our dataset. Using Bark, we made two datasets of 500 audio based on the two translated datasets. Each one is based on the exact English phrases and uses the same speaker to not only have the overall performances of the model but to compare the quality of the translations accurately and have an idea of how much information we have lost through the whole translating process of the subset and generating audio.

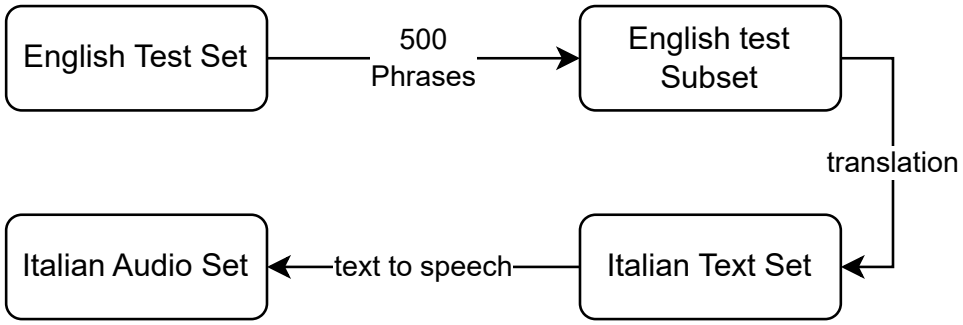


Figure 3.8: Audio Dataset Creation Flow Chart

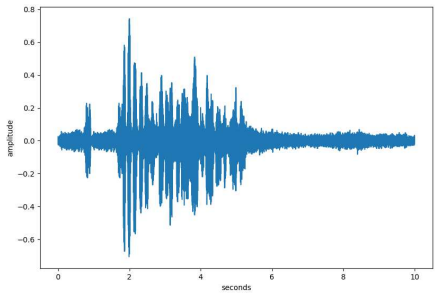
3.3 Assembling the Pipeline

Even if the pipeline is straightforward, actually integrating the pipeline is another task. Other tasks weren't in the pipeline, such as audio acquisition, downloading, and inferring on the different models. These things are crucial for the inference time and user interaction.

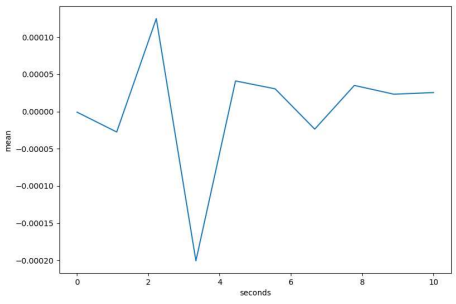
For the audio, we used the sounddevice modules that provide a binding for the portAudio library (based in C) in Python. It allows the use of different audio ports, especially inputs. Unfortunately, we noticed that sometimes Whisper, especially the smaller models, struggles with long "empty" audio inferring weird things, and cutting the audio when the user is finished speaking could shorten the computation time. So, having a fixed record time could be problematic, and to solve that problem, we integrated a stopping criteria for the recording. This stopping criterion is simply the standard deviation of the signal recorded over the previous second (or any preset time unit). We record blocks of one second by default that we stack to have the full recording using the aforementioned stopping criterion.

[3]Tatoeba project: https://tatoeba.org/fr

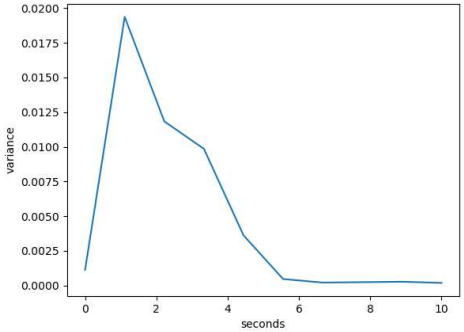
[4]Bark available here: https://github.com/suno-ai/bark



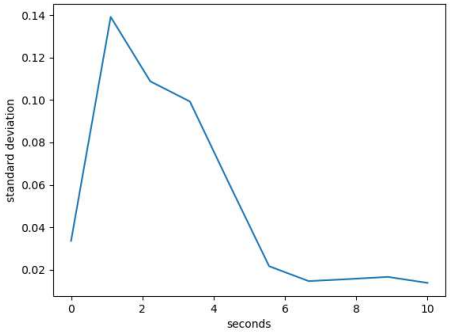
(a) The amplitude of a signal



(b) The mean of the signal



(c) The variance of the signal



(d) The standard deviation of the signal

Figure 3.9: Analysis of audio, speaking the first 5 seconds and busy street road noises in the background and then silence

The signal amplitude being positive and negative is unusable and, therefore, also the mean of this signal. Empirically, the standard deviation seemed more practical to set up and more stable as a stopping criterion. As Figure 3.9 shows, the signal's amplitude and mean can't easily be used since it can be negative. On the other hand, after empirical testing, it was more practical to choose a stopping threshold using the standard deviation rather than the variance, which offers more stability in practical use.

Another important thing for the pipeline is its initialization and deployment. Indeed, this project is subject to real-life applications, and the practical pipeline is slightly different than the theoretical one (see Figure 3.10). The models are loaded either from HuggingFace or locally, initialized, and then ready for inference to avoid downloading DistilBERT and whisper before each use. The pipeline is then looped to be used without waiting between classification requests.

We have downloaded the models from Hugging Face using the Transformers library, so when launching the pipeline, we have to download whisper and our fine-tuned DistilBERT. This leads us to a very long computation time if we have to download everything again for each classification. To avoid that, we loop the classification process after downloading everything. So our complete pipeline looks like Figure 3.10.

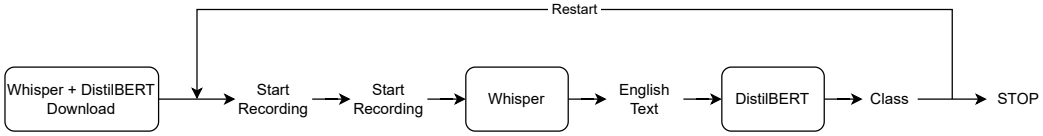


Figure 3.10: Complete pipeline

Chapter 4

Results

4.1 Results of the classification task

After different training conditions on the previously presented datasets, We have four different interesting models that we are going to analyze. For all the models, we used the Sparse Categorical Cross entropy loss from Keras, which allows us not to one hot encode our predictions and is suitable for multi-label classification. We are using this one because it is one of the most commonly used for multi-class classification and works well. This loss corresponds to the Kullback-Leibler Divergence, which measures the loss of information between two series when one is supposed to be an approximation of the other.

For the optimizer, we are using Adam [32], which is the most common optimizer nowadays; it is a stochastic gradient descent method based on adaptive estimation of first-order and second-order moments. Adam has various hyperparameters that can be set up, but, in our case, we are only going to modify the learning rate, which, as its name suggests, defines the rate of learning of the model, and also the ε [33] which is used to avoid 0 division and large ε can have an impact.

Three different versions were tested on our final testing set. We trained the various model versions using the hyperparameters specified in Table 4.1.

model	dataset	epochs	learning rate	epsilon
1st model	1	3	1×10^{-7}	1×10^{-8}
2nd model	1	3	5×10^{-7}	1×10^{-8}
3rd model	2	3	1×10^{-7}	1×10^{-8}
4th model	2	3	5×10^{-7}	1×10^{-8}

Table 4.1: Training parameters and hyperpameters of the different models

As explained before, the first two models were trained on a dataset without a validation and the 2 second with a validation set. The validation test is used during training to fine-tune hyperparameters. All of the models were trained on three epochs; taking into account the size of the dataset, the size of the model, and its performance, training for more epochs could cause overfitting.

If we denote:

- *True positive* (TP): Truth is positive and prediction is positive.
- *True negative* (TN): Truth is negative and prediction is negative.
- *False positive* (FP): Truth is negative and prediction is positive.
- *False negative* (FN): Truth is positive and prediction is negative

		Predicted Values	
		Positive	Negative
Actual Values	Positive	True Pos (TP)	False Neg (FN)
	Negative	False Pos (FP)	True Neg (TN)

Table 4.2: Confusion Matrix

The metrics that we are using are the following:

- The *accuracy* is the ratio of good predictions amongst all the predictions.
- The *precision* (or true positive rate) is the probability of a positive result, so $\frac{\text{true positives}}{\text{true positives+false positives}}$ and a low precision involves a lot of false positives.
- The *recall* (or true positive rate) is the ratio positive predictions among the actual positive values, so $\frac{\text{true positives}}{\text{true positives+false negatives}}$, a low recall involves a lot of false negatives.
- The *f1 score* represents the arithmetic mean of the precision and recall. It carries both information.

metrics	**Formulas**
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
f1 score	$\frac{1}{\frac{1}{prec}+\frac{1}{recall}} = \frac{2TP}{2TP+FP+FN}$

Table 4.3: Metrics used and their formulas.

It is essential to choose coherent metrics that carry the information we need to analyze and that can be interpreted as it varies. For instance, the specificity (or true negative rate) is a very classic metric that can be useful to analyze the population

of false positives, the specificity being the probability of a negative outcome, so $\frac{FP}{FP+TN}$. Unfortunately, in our case, since the number of objects per class compared to the number of objects is very low and there are not a lot of false negatives, the specificity does not vary enough to be exploited and meaningful.

In multi-label classification, the metrics are computed class per class, and the result is the average (the classes are unweighted). Because of the number of positive cases for each case relative to the large number of cases, both the precision and recall are good indicators and carry different information.

model	**metrics**
1st model	- accuracy: 0.9729 - precision: 0.9809 - recall: 0.9709 - f1 score: 0.9714
2nd model	- accuracy: 0.9739 - precision: 0.9805 - recall: 0.9732 - f1 score: 0.9725
3rd model	- accuracy: 0.9710 - precision: 0.9677 - recall: 0.9771 - f1 score: 0.9666
4th model	- accuracy: 0.9703 - precision: 0.9773 - recall: 0.9695 - f1 score: 0.9673

Table 4.4: Metrics on test set for different versions of the models.

In Figure 4.4, we can see that adding the validation set did not really change the overall result. There was also a slight improvement when changing the learning rate between models 1 and 2.

The results of the models seem balanced overall with slightly different behavior. The precision is higher than the recall for model 1, meaning that the model is more specific than sensitive, so we produce slightly more false negatives than false positives. We also computed the results per class for model 2 (see Table 4.5 only showing the classes without perfect classification) to see what is precisely the classification behavior.

The classification is not perfect in only eight classes, actually, being 5: compostable packaging, 18: paper bowl, 19: paper cup, 21: paper magazine, 30: plastic bowl, 34: plastic cutlery, 36: plastic gloves, 42: receipt. In all cases, except for plastic bottles, it is either the recall or the precision that is not 1, so there are only false positives or false negatives.

The results of these classes are also interesting. For each class, the results are not random but seem biased. Among one class when false negatives are produced, it is always only in one class, and sometimes, there is another one. Still, we can observe

class	mean	precision	recall	f1 score
cigarette butt	0.9998	1.0000	0.9894	0.9947
compostable packaging	0.9872	1.0000	0.4052	0.5767
paper bowl	0.9976	0.9449	1.0000	0.9717
paper cup	0.9974	1.0000	0.9278	0.9626
paper magazine	0.9870	0.7154	1.0000	0.8341
plastic bottle	0.9918	0.6556	0.8194	0.7284
plastic cap	0.9974	0.8833	1.0000	0.9381
plastic cutlery	0.9946	1.0000	0.6506	0.7883
plastic gloves	0.9978	0.8261	1.0000	0.9048
receipt	0.9974	1.0000	0.8667	0.9286

Table 4.5: Classes without perfect classification on test set with model 2.

the same phenomenon with false positives in fewer instances. False negatives and positives are inter-influential between classes, so it is a good sign that we also have only one/two classes among the false positives, showing a bias. A solution for this problem is to analyze the AUC score and the ROC curve and work with thresholds just like in binary classification. The ROC curve, standing for Receiver Operating Characteristic, is the curve of true positive rate, the proportion of correctly classified positive cases, depending on the false positive rate, the proportion of negative cases incorrectly classified as positive. A good model will have a high true positive rate and a low false positive rate, maximizing the number if true positives while maximizing the number of false positives. The AUC, standing for Area Under the Curve, is a metric used to summarize the ROC curve by calculating the area under the ROC curve. The AUC indicator is between 0 and 1, and an AUC close to 1 indicates that a model has a good ability to differentiate classes.

Given the number of classes, it is normal to have this kind of behavior during the classification process. The more classes we have, the more complicated it is to be specific in the majority of classes without having to degrade the sensitivity. This kind of error can be corrected through more training by modifying the dataset and the optimizer parameters. Given the size of the dataset and the number of classes, this might be difficult.

We can observe that, on the set modified using ChatGPT, the metrics of the 2nd model are slightly higher than the metrics of the final model (Table 4.6), showing that even if the final model has better results on the actual test set, the third model may have slightly better adaptation capabilities but, in overall, the performances of both models are extremely close and negligible. An important thing to consider here is that there's no significant difference between the results of the test set and the ChatGPT "improved" set. The models do not seem to lose their capabilities to capture the semantics of a phrase even after training over a "monotone" dataset, and this is where we see the advantages of pre-trained models. Over only three epochs, the slight change in the learning rate provides small changes in the training behavior and their behavior towards new data. If we do a performance comparison, we have a lot of errors in the same classes showing similarities in the learning, but the 3rd model seems more stable overall.

model	metrics
1st model	- accuracy: 0.964 - precision: 0.9773 - recall: 0.9486 - f1 score: 0.9542
2nd model	- accuracy: 0.96 - precision: 0.9683 - recall: 0.9482 - f1 score: 0.9457
3rd model	- accuracy: 0.96 - precision: 0.9483 - recall: 0.9684 - f1 score: 0.9457
4th model	- accuracy: 0.9620 - precision: 0.9716 - recall: 0.9538 - f1 score: 0.9546

Table 4.6: Metrics on ChatGPT modified set for different versions of the model

These results (see Figures 4.4 and 4.6) also highlight some behaviors of pre-trained large language models, such that they do not always need, in specific conditions like here, validation sets to fine-tune hyperparameters to achieve similar performances as training with validation set.

We can also note that, with or without a validation set, the models have similar learning patterns and, thus, behaviors.

4.2 Performances of the pipeline

Before showing and analyzing the results, it is essential to understand how the information has changed and degraded. We have translated a subset of our dataset and generated audio to create a new dataset. To only create the new dataset, we have degraded our original dataset. The beginning of the whole pipeline uses whisper for automatic speech translation. The BLEU score for the speech translation from Italian to English is very low (between 5.3 and 23.6 depending on the model size, see Figure 3.2). So, the input of the classification model might be really different from the original dataset, and sometimes it does not even make sense anymore. For instance: "Where do I throw a tetra pack" becomes "Where? On the whiteboard." using whisper base. Unfortunately, this method of measuring the results is not the most accurate, and future results are certainly lower than actual results due to the loss of information in the process of creating the dataset. The full pipeline is still working, and we can test it with the different models that are available.

We are using the same metrics as before to analyze the results of the prediction of the full pipeline on our audio datasets.

class	# elements	mean	precision	recall	f1 score
overall	1000	0.96	0.9684	0.9482	0.9457
aluminium tray	6 (0.6%)	0.9950	1.0000	0.1667	0.2857
compostable packaging	25 (2.5%)	0.9820	1.0000	0.2800	0.4375
condiment packets	37 (3.7%)	0.9990	1.0000	0.9730	0.9863
paper bowl	40 (4.0%)	0.9980	0.9524	1.0000	0.9756
paper cup	35 (3.5%)	0.9980	1.0000	0.9429	0.9706
paper magazine	38 (3.8%)	0.9810	0.6667	1.0000	0.8000
plastic bottle	16 (1.6%)	0.9940	0.7778	0.8750	0.8235
plastic cap	9 (0.9%)	0.9980	0.8182	1.0000	0.9000
plastic cutlery	15 (1.5%)	0.9950	1.0000	0.6667	0.8000
plastic gloves	11 (1.1%)	0.9960	0.7692	0.9091	0.8333
plastic snack packaging	15 (1.5%)	0.9960	0.7895	1.0000	0.8824
plastic sticks	15 (1.5%)	0.9940	0.9091	0.6667	0.7692
receipt	15 (1.5%)	0.9940	0.7368	0.9333	0.8235

(a) Classes with not perfect classification on GPT modified set with model 2

class	# elements	mean	precision	recall	f1 score
overall	1000	0.964	0.9773	0.9486	0.9542
aluminium tray	6 (0.6%)	0.9970	1.0000	0.5000	0.6667
compostable packaging	25 (2.5%)	0.9820	1.0000	0.2800	0.4375
organic scraps	43 (4.3%)	0.9980	0.9556	1.0000	0.9773
paper bowl	40 (4.0%)	0.9980	0.9524	1.0000	0.9756
paper cup	35 (3.5%)	0.9970	0.9211	1.0000	0.9589
paper magazine	38 (3.8%)	0.9810	0.6667	1.0000	0.8000
plastic bottle	16 (1.6%)	0.9940	0.7778	0.8750	0.8235
plastic cap	9 (0.9%)	0.9970	1.0000	0.6667	0.8000
plastic cutlery	15 (1.5%)	0.9950	1.0000	0.6667	0.8000
plastic gloves	11 (1.1%)	0.9960	0.7692	0.9091	0.8333
plastic sticks	15 (1.5%)	0.9990	1.0000	0.9333	0.9655
receipt	15 (1.5%)	0.9960	0.8235	0.9333	0.8750
tetrapack	6 (0.6%)	0.9980	1.0000	0.6667	0.8000

(b) Classes with not perfect classification on GPT modified set with model 3

With the results from Table 4.8 that we have, we can clearly observe the degradation of the information before being classified. In the baseline column, we have computed the metrics of the same dataset before its modifications. We can also observe that the metrics of the two models are overall almost the same, the 4th model is a little bit better, but most importantly, we can observe that the translation quality significantly impacts the quality of the classification. It is conceivable that the results on a "real" dataset might be better.

Given that the goal of the project is to offer a good user experience, another important thing to consider is the inference time. The goal is to find a compromise between the performance of the model and the inference time. To achieve these measures, we are using a MacBook Air 2022 with an Apple M2 chip, 8GB of

helsinki	Deepl	baseline
- accuracy: 0.6820 - precision: 0.6836 - recall: 0.6737 - f1 score: 0.6491	- accuracy: 0.6940 - precision: 0.6959 - recall: 0.6617 - f1 score: 0.6500	- accuracy: 0.9620 - precision: 0.9716 - recall: 0.9538 - f1 score: 0.9546

Table 4.8: Metrics computed with whisper large-v2 and model 4

RAM, and no GPU. We do not know the specifications of the future machine that the company will use. Looking at Table 4.9, the computation time is linear with respect to the size of the model. In a practical situation, even if the performance of the medium model is much better, the computation time is not suited for the appropriate use, whereas the two others are more suited for daily use.

whisper size	inference time
base (74M)	$\sim$ 2.5 seconds
small (244M)	$\sim$ 6.5 seconds
medium (769M)	$\sim$ 18 seconds

Table 4.9: Inference time depending on the size of whisper

Only a few months later, after the first inferences on the pipeline, HuggingFace integrated the deployment of whisper using its pipeline tool, taking advantage of the transformers architecture and their parallelization capabilities. Using V100 GPUs rented from Google Colab notebooks, the inference time for all models is very similar for all model sizes (from tiny with 34 million parameters to large-V2 or large-V3 with 1.5 billion parameters), ranging from 1 to 1.4 seconds.

Since all the models have very similar results and behaviors, we only computed the inference on different whisper sizes on the 4th model. The other models have also very similar results.

Whisper Version	Accuracy	Precision	Recall	F1-Score
baseline	0.9620	0.9716	0.9538	0.9546
Tiny	0.1040	0.1141	0.0829	0.0818
Base	0.2580	0.3543	0.2330	0.2491
Small	0.4760	0.5089	0.4541	0.4400
Medium	0.664	0.697	0.630	0.634
Large-v2	0.6940	0.6959	0.6617	0.6500

Table 4.10: Results of the entire speech classification pipeline with different versions of whisper and model 4

The results align with our expectations. As the quality of the ASR component improves, we see a positive impact on the system's overall metrics. This highlights ASR's crucial role within the pipeline as the foundation for accurately interpreting user requests. By continuously refining the ASR module, we can ensure a more

robust system capable of consistently delivering exceptional performance. It is good to remember that these results are also informative on how well the pipeline would behave in a real-world application. The audio dataset is generated using transformers Deep Learning models, extracted from translated phrases from English to Italian using Deepl API, also a tranformers Deep Learning Model.

Chapter 5

Conclusion

Smart bins like NANDO can optimize waste management and recycling processes. This thesis contributes to NANDO's effectiveness by introducing a novel speech classification pipeline for accurately recognizing and categorizing user requests according to available recycling options.

The pipeline's success hinges on its modular design, incorporating components for voice acquisition, automatic speech recognition (ASR), and text classification. In this context, we crafted a novel text classification dataset, which includes 50 classes of waste objects that can be used to train and evaluate models for the identification of the correct bin for the disposal of a specific object. The system has shown promising results in both Italian and English languages.

Looking ahead, the future work could focus on multiple parts. The first could be to optimize the pipeline and expand its language capabilities. Such things can be done by modifying and upgrading the dataset and/or the training flow. This could include training DistilBERT on the output of the ASR outputs and creating new specific datasets to improve the quality of the evaluation of the component of the pipeline. Additionally, integrating this pipeline into a real-world NANDO system with a user feedback mechanism is crucial. Such a mechanism would allow users to correct misclassifications, fostering a human-in-the-loop system that continuously improves over time. This collaborative approach holds immense promise for the future of smart waste management and responsible resource recovery.

Annex: Tables and figures

0: aluminium can	1: aluminium sheet	2: aluminium tray
3: cigarette butt	4: cigarette pack	5: compostable packaging
6: condiment packets	7: covid test	8: crumbled tissue
9: cylindrical battery	10: face mask	11: glass bottle
12: glass jar	13: laptop charger	14: meds blister
15: metal cap	16: mixed paper-plastic packaging	17: organic scraps
18: paper bowl	19: paper cup	20: paper food packaging
21: paper magasine	22: paper packaging	23: paper plate
24: paper sheet	25: paper sugar bag	26: paper tray
27: phone charger	28: pizza box	29: plastic bag
30: plastic bottle	31: plastic bowl	32: plastic cap
33: plastic cup	34: plastic cutlery	35: plastic dish
36: plastic gloves	37: plastic packaging	38: plastic snack packaging
39: plastic sticks	40: plastic straw	41: plastic tray
42: receipt	43: smartphone	44: tea bag
45: tetrapack	46: tobacco pack	47: transport ticket
48: wooden cutlery	49: wooden sticks	

Table 5.1: All classes indexes and their names

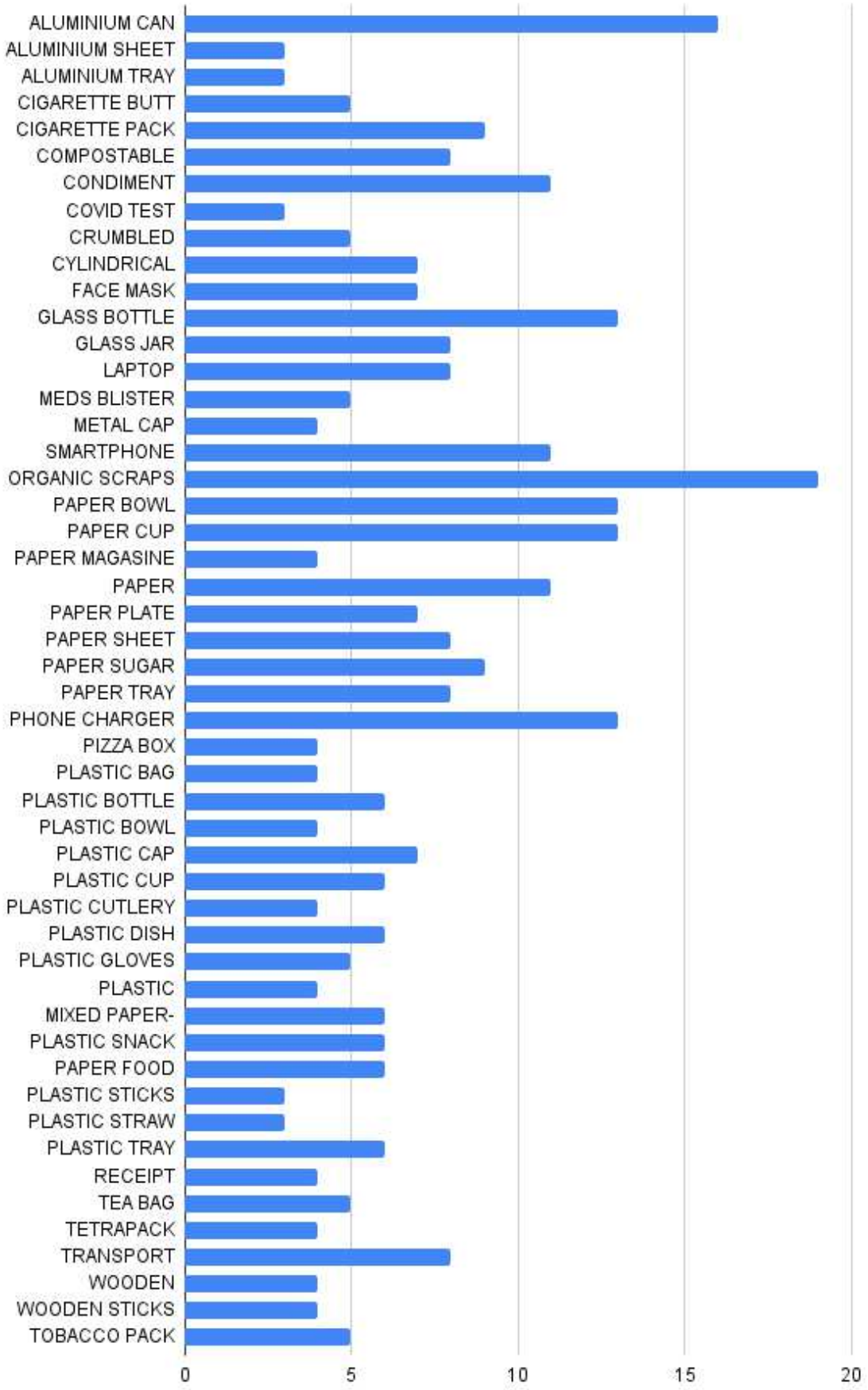


Figure 5.1: Number of objects per class

Bibliography

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[2] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[3] Radford, Alec, et al. "Language Models are Unsupervised Multitask Learners" (2019)

[4] La Quatra, M., Koudounas, A., Vaiani, L., Baralis, E., Cagliero, L., Garza, P., & Siniscalchi, S. M. (2024). Benchmarking Representations for Speech, Music, and Acoustic Events. arXiv preprint arXiv:2405.00934.

[5] Turian, J., Shier, J., Khan, H. R., Raj, B., Schuller, B. W., Steinmetz, C. J., ... & Bisk, Y. (2022, July). Hear: Holistic evaluation of audio representations. In NeurIPS 2021 Competitions and Demonstrations Track (pp. 125-145). PMLR.

[6] Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. Advances in neural information processing systems, 33, 12449-12460.

[7] Hsu, W. N., Bolte, B., Tsai, Y. H. H., Lakhotia, K., Salakhutdinov, R., & Mohamed, A. (2021). Hubert: Self-supervised speech representation learning by masked prediction of hidden units. IEEE/ACM transactions on audio, speech, and language processing, 29, 3451-3460.

[8] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023, July). Robust speech recognition via large-scale weak supervision. In International conference on machine learning (pp. 28492-28518). PMLR.

[9] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

[10] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

[11] FitzGerald, J., Hench, C., Peris, C., Mackie, S., Rottmann, K., Sanchez, A., ... & Natarajan, P. (2022). Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. arXiv preprint arXiv:2204.08582.

[12] Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., ... & Dureau, J. (2018). Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. arXiv preprint arXiv:1805.10190.

[13] Koudounas, A., La Quatra, M., Vaiani, L., Colomba, L., Attanasio, G., Pastor, E., ... & Baralis, E. (2023). Italic: An italian intent classification dataset. arXiv preprint arXiv:2306.08502.

[14] emphill, C.T., Godfrey, J.J., Doddington, G.R.: The ATIS spoken language systems pilot corpus. In: Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990 (1990). URL https://aclanthology.org/H90-1021

[15] arson, S., Mahendran, A., Peper, J.J., Clarke, C., Lee, A., Hill, P., Kummerfeld, J.K., Leach, K., Laurenzano, M.A., Tang, L., et al.: An evaluation dataset for intent classification and out-of-scope prediction. In: Proceedings of EMNLP- IJCNLP 2019, pp. 1311–1316 (2019)

[16] Barrault, L., Chung, Y. A., Meglioli, M. C., Dale, D., Dong, N., Duquenne, P. A., ... & Wang, S. (2023). SeamlessM4T-Massively Multilingual & Multimodal Machine Translation. arXiv preprint arXiv:2308.11596.

[17] Rubenstein, P. K., Asawaroengchai, C., Nguyen, D. D., Bapna, A., Borsos, Z., Quitry, F. D. C., ... & Frank, C. (2023). Audiopalm: A large language model that can speak and listen. arXiv preprint arXiv:2306.12925.

[18] Zhang, Y., Han, W., Qin, J., Wang, Y., Bapna, A., Chen, Z., ... & Wu, Y. (2023). Google usm: Scaling automatic speech recognition beyond 100 languages. arXiv preprint arXiv:2303.01037.

[19] Wang, C., Wu, A., & Pino, J. (2020). Covost 2 and massively multilingual speech-to-text translation. arXiv preprint arXiv:2007.10310.

[20] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461.

[21] Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., ... & Bowman, S. (2019). Superglue: A stickier benchmark for general-purpose language understanding systems. Advances in neural information processing systems, 32.

[22] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140), 1-67.

[23] Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

[24] OpenAI. 2023a. GPT-4 technical report. (2023)

[25] Smith, Shaden, et al. "Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model." (2022).

[26] Nicholson, Ann E., et al. "BARD: A structured technique for group elicitation of Bayesian networks to support analytic reasoning." (2020).

[27] Anil, Rohan, et al. "Palm 2 technical report." arXiv preprint arXiv:2305.10403 (2023).

[28] M. La Quatra, A.C. Marceddu, T. Avare, F. Fedi, M. Brusamento, B. Montruchio, "Improving Recycling Experience through a Novel Speech Classification Pipeline", Smart and Innovation, Systems and Technologies, Springer, 2024

[29] Jörg Tiedemann and Santhosh Thottingal "OPUS-MT — Building open translation services for the World" Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT) (2020) Lisbon, Portugal

[30] Sebo, P., & de Lucia, S. (2024). Performance of machine translators in translating French medical research abstracts to English: A comparative study of DeepL, Google Translate, and CUBBITT. PloS one, 19(2), e0297183. https://doi.org/10.1371/journal.pone.0297183
Download .nbib Format:
[31] BARK, suno-ai, https://github.com/suno-ai/bark/tree/main
[32] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
[33] Yuan, W., & Gao, K. X. (2020). EAdam optimizer: How ϵ impact adam. arXiv preprint arXiv:2011.02150.