DataOps/MLOps Project Assignment

Neil Scheidwasser
PhD student
University of Copenhagen
neil.clow@sund.ku.dk

Raffaele Pojer
PhD student
Aalborg University
rafpoj@cs.aau.dk

November 3, 2024

Contents

1 Phase 1: Application Context and Data Selection

1.1 Summary

Over the past decade, recent advancements in deep learning and artificial intelligence (AI) have catalysed tremendous progress in natural language processing (NLP). In the eyes of the public and the markets, products powered by large language models (LLMs) such as ChatGPT (OpenAI 2022; Achiam et al. 2024; OpenAI 2024), Claude (Anthropic 2023), Gemini (Google Deepmind 2023; Georgiev et al. 2024), and LLama (Touvron, Lavril, et al. 2023; Touvron, Martin, et al. 2023; Dubey et al. 2024) have garnered widespread attention. In addition to their simple, yet enticing, user interface, the underlying models have shown impressive capabilities in a wide range of NLP tasks that extend beyond realistic language generation, including text summarisation, code completion and debugging, image captioning and prompt-based image generation.

At the inception of the "Transformer era" of NLP (Bahdanau, Cho, and Bengio 2015; Vaswani et al. 2017), machine translation was the task that propelled the development of current LLMs. Despite significant progress in that area, text-based machine translation quality remains poorer than expert translation (Yan et al. 2024). Worse still, although a promising solution in healthcare and judiciary facilities where human translation is not always possible, speech-based machine translation presents even greater challenges. On the data side, speech data is much less available and more difficult to collect than text data, especially in low-resource languages. Moreover, speech is inherently more diverse than text - consider, for example, the range of dialects, and accents that cannot be represented in written form. On the model side, speech translation models are rarely standalone, but instead part of a pipeline that often comprises automatic speech recognition (ASR) and language identification (LID).

In particular, LID is important in multilingual settings (e.g., parliaments where speakers speak in their native language). Indeed, identifying language with high accuracy could help minimise potential errors in machine translation by correctly identifying the

source language, and is a useful feature for translation software (e.g., the "Detect Language" button in Google Translate). In this project, we will develop a prototype for North Germanic languages, which mainly encompasses Danish, Norwegian, Swedish, Icelandic, and Faroese. Potential challenges we anticipate regarding training and inference data are language (i.e., label), dialect (i.e., intra-language variation) and gender imbalance. Should more data be collected to mitigate those balances, data quality variability will also be challenging, with background noise and recording quality being potential disturbances. Lastly, the dynamic nature of language (evolving accents and pronounciation, increasing influence of English...) could lead to data drift on a medium-to-long term perspective. Here, we will benchmark and train open-source speech models (XLS-R (Babu et al. 2022), MMS (Pratap et al. 2024), Whisper (Radford et al. 2023), ECAPA-TDNN (Desplanques, Thienpondt, and Demuynck 2020)) on a corpus of open-source datasets comprising read samples in the aforementioned languages. Our metric will consist of a composite score including multi-class accuracy to measure performance and inference speech to measure computational efficiency metrics. Although not included in the prototype, one could also include imbalance-agnostic performance metrics such as macro-averaged recall or F1 scores (Barrault et al. 2023) to consider all classes fairly in case of drastic imbalance.

Our implementation, which comprises source code, notebooks, installation steps, metadata, MLOps outputs, and plots is available at: `https://github.com/Neclow/mlops_project`.

2 Phase 2: DataOps/MLOps Pipeline Conceptualization

2.1 Data Quality Assurance

Bringing our project from the idea stage to a minimum viable product would involve two main stages: i) developing a prototype based on pre-trained models and open-source data, and ii) fine-tuning models on new recordings collected from new sources. For the latter, one could first envisage conducting data collection campaigns (with or without the help of subcontractors, e.g., with universities or recording studios). Although this approach would guarantee a high data quality standard, crowdsourcing is less costly and logistically difficult. For these reasons, crowdsourcing has been employed in large-scale multilingual data collection projects such as Common Voice (Ardila et al. 2020) (Mozilla Foundation) or Aya (Üstün et al. 2024) (Cohere). However, data quality is less trivially guaranteed, as speakers do not necessarily have the means or access to a high-quality recording environment. Thus, data quality assurance is crucial to ensure service robustness. For this prototype, we will assume that the open-source datasets have been correctly labelled as they were produced by trusted sources (e.g., FLEURS-R (Conneau et al. 2023; Ma et al. 2024) by Google, Common Voice (Ardila et al. 2020) by Mozilla, and Ravnursson (Hernandez Mena and Simonsen 2022) by Reykjavík University). Moreover, all these datasets were published in Hugging Face, a platform to share machine learning models and datasets, and are thus subject to review by the public. Finally, we do not have the expertise to fact-check the labelling at this stage.

With that in mind, because our corpus (Table 3.1) encompasses several datasets, we will first use the `Great Expectations` package to examine the possibility of duplicated data as well as metadata adequacy. This would include checking that the audio duration is adequate for model training (1-60 seconds; we hypothesise that 60 seconds of audio is more than sufficient to identify language) and that the associated metadata provided alongside the datasets is consistent (e.g., floating-point audio durations, standard audio sample rates (16/24/32/48 kHz)). Subsequently, we will examine for potential biases

in the open-source data, including language and gender imbalance (age data is unavailable in all datasets). Lastly, we will implement a simple data quality check based on pre-trained speech quality assessment models such as `torchaudio-squim` (Kumar et al. 2023), which can help provide objective measures of speech quality and intelligibility. Voice activity detection (VAD) could also be used for data quality and cleaning. Regarding data quality, recordings that are mostly or entirely silent would be significantly altered/trimmed by VAD transform, indicating poor signal quality. As to data cleaning, passing a VAD transform would help eliminate background noise or silence from the input.

2.2 Data Management

The data management strategy for our MLOps pipeline is structured to ensure data accessibility, consistency, and traceability across all phases of model development and deployment. Given the multilingual nature and diversity of our datasets (in this case, FLEURS-R, Common Voice, and Ravnursson), this section outlines our approach to data storage, versioning, quality checks, and lineage, all of which are essential for effective training and evaluation.

2.2.1 Data Versioning and Lineage

In our concept, we implement versioning for raw and processed data to facilitate reproducibility and compliance. DVC (*DVC* n.d.) will be used to version and track changes. To ensure that each model training instance can be replicated precisely, it will be done on both of the changes, the dataset and features levels. Metadata, including data source, collection date, and preprocessing steps, will be stored to track data lineage. This will allow us to trace the origin of data and ensure that the data used for training is consistent with the data used for evaluation.

2.2.2 Feature Store

Feature engineering is a crucial step in model development, and it is essential to store and manage features effectively. We will use a feature store to store and manage features, enabling us to reuse features across different models and experiments. This will also allow us to track the lineage of features and ensure that the features used for training are consistent with the features used for evaluation. In our context, by storing key features such as language embeddings, audio metadata, and linguistic patterns in a structured

repository, we create a reliable source of truth for model inputs that can be shared and reused by different models within the pipeline. Language-specific features, such as *frequency patterns* or *language embeddings*, can be extracted, processed, and stored once in the feature store. This eliminates the need to recompute these features each time a new model is trained, saving time and computational resources. For instance, the feature store could contain the processed embeddings for Danish, Norwegian, and other languages from datasets like *Common Voice* and *FLEURS-R*, making them accessible to all models that require these representations. For this project, implementing a feature store such as Feast (an open-source feature store) (Feast 2024) could help by providing both batch (for training) and real-time (for serving) access to features.

2.2.3 Data Management Strategy

To effectively manage crowd-sourced audio data, a comprehensive data management strategy is essential. This approach centers on automating workflows, ensuring data quality, and maintaining rigorous traceability throughout the data lifecycle. The first priority is implementing stringent data quality control measures. Given the variability in audio sources, consistent quality checks for factors such as background noise and clarity are crucial. This ensures that only reliable data is used for modeling, which is vital for maintaining the integrity of the models being developed. To facilitate the growing volume of audio data, a scalable and organized storage solution will be established. Utilizing cloud services such as AWS S3 or Google Cloud Storage allows for easy expansion of storage capacity as data increases. Data versioning and automation are critical components of this strategy. By employing tools such as Apache Airflow or Metaflow, we can automate the processes of data ingestion, quality checks, and subsequent storage. This automation reduces the need for manual intervention, thereby enhancing consistency and reliability in the dataset. Additionally, scheduling regular processing through these tools ensures that new data is continuously integrated into the system, ready for retraining the models as required.

2.3 Model Development and Versioning

In the model development stage, we aim to benchmark several language identification models on our training corpus, with the overarching goal of serving two models to our customers: a “premium” model with state-of-the-art accuracy and a “‘mini” model, that is less precise but faster at inference time. Thus, in addition to performance metrics such as multi-class accuracy and macro-averaged recall (also called unweighted average recall

(UAR)), we will track system metrics such as runtime, FLOPs and memory requirements. To that end, we will rely on *Weights & Biases* (W&B), a user-friendly tool for model development and versioning. On the training side, W&B allows for tracking and visualising training experiments with minimum friction, with all runs being displayed in a dashboard that can be easily shared with other team members. For hyperparameter optimization, W&B also includes a "sweep" mode to automate hyperparameter searching processes by coordinating multiple experiments. This mode allows users to specify a range or distribution of possible values for each hyperparameter and to run experiments across these values using optimization strategies like grid search, random search, or Bayesian optimization. W&B can also save logs and a "code state" for git-based projects to reproduce the training experiments, thereby versioning the tested models. In addition, W&B's artefact storage allows for retrieving hyperparameters, model weights, and optimizer states after training, which can be accessed later for deployment or fine-tuning. W&B also facilitates dataset versioning, allowing us to track different dataset versions as models evolve, and provides a model registry to track and organize various model versions. These capabilities, combined with collaboration tools like shared reports and team comments, allow for seamless documentation, tracking, and reproducibility of the model development process.

2.4 Continuous Training

Continuous training is crucial for maintaining model relevance, especially given the fact of evolving language variations and accents in North Germanic languages. Our approach to continuous training will focus on establishing retraining triggers and, validation gates.

2.4.1 Retraining Triggers

Using the continuous monitoring component, retraining will be triggered when specific conditions are met, such as a significant decline in model accuracy or data distribution shifts, detected through metrics like covariate shift or concept drift in language detection accuracy. We will also consider, to prevent performance degradation from outdated data, a sliding window approach will be used, where the latest data (most recent language samples) replaces the oldest data in the training set. This approach will allow the model to adapt incrementally to recent linguistic trends.

2.4.2 Validation Gates

Before new data enters the training pipeline, it will undergo validation checks to ensure that it aligns with established data quality thresholds (e.g., language and gender balance, no missing or null values). Data that does not meet these criteria will be flagged and excluded from the training set. This will help maintain the quality of the training data and prevent the model from learning from noisy or biased data.

2.5 Model Deployment and Prediction Serving

A possible deployment strategy for our project could involve using `FastAPI` to create a microservice that wraps the trained model. The model could also be packaged within a Docker container, which ensures portability across different environments. Given that language identification typically requires real-time processing, on-demand streaming serving is the most suitable serving method. We could implement an event processor to handle incoming audio streams and provide near real-time predictions using the model serving runtime and the trained model. This method optimises latency, adapts to changing contexts, and ensures resources are used efficiently.

In addition, we plan to deploy two models: a large, accurate model and a small, fast model. Thus, when a request arrives, the system could first utilise the small model to provide a quick, initial prediction. Should the confidence of this prediction be below a certain threshold, the user could prompt the system to pass the requests to the large model for a more accurate prediction. This would help balance accuracy and speed depending on the customer's needs. Determining the optimal confidence threshold will require careful tuning, which could be optimized offline in the model development stage.

2.6 Monitoring and Logging

In the context of our project, where the model is deployed on crowd-sourced audio data, effective monitoring and logging are essential to manage the variability in data quality, recording environments, and speaker characteristics. These factors introduce challenges in maintaining consistent model performance and reliability over time, making monitoring data drift, performance degradation, and prediction errors critical. Specific requirements are addressed:

- **Data Variability Management**: Crowd-sourced audio data is subject to fluctuations in quality due to diverse recording environments and devices. This vari-

ability necessitates continuous monitoring to detect changes that could impact the model's performance.

- **Error Tracking in Uncontrolled Environments**: Operating in real-world environments introduces unpredictable issues, such as noise interference, that could affect the accuracy and reliability of model predictions. Logging helps capture these occurrences, providing insights to improve resilience.

- **Real-Time Performance Insights**: Given the dynamic nature of crowd-sourced data, tracking performance metrics in real-time is essential to identify degradation quickly and take corrective action.

Data Drift Detection: Data drift is a significant concern with crowd-sourced due to several factors:

- **Environmental Noise**: Contributors record audio in different settings, introducing varying background noise levels. For example, recordings made outdoors or in public spaces may contain traffic, voices, or mechanical noise.

- **Device Quality Variability**: The audio is recorded on a wide range of devices, from professional microphones to basic smartphone microphones, resulting in inconsistencies in audio clarity and frequency response.

- **Speaker Variability**: Differences in accents, speech speed, and pronunciation create variability in the audio features extracted from the recordings.

. These factors cause data drift, where the production data differs from the training data. Monitoring and managing this drift is crucial to ensuring consistent model performance. Using Evidently AI, we monitor for shifts in the distribution of input features, such as Mel-frequency cepstral coefficients (MFCCs), spectral centroid, and zero-crossing rate, which can indicate changes in audio quality, device type, or environment. Evidently AI provides automated drift detection through the `DataDriftReport`, generating regular comparisons between the reference dataset (baseline) and the current production data.

Monitoring for data drift enables the team to respond proactively to changes in the data, which could arise from different recording devices or ambient noise levels in the recordings. Drift alerts inform the need for retraining or model adjustments, ensuring the model continues to perform well under evolving conditions.

3 Phase 3: Prototype Implementation

3.1 Data quality assurance

3.1.1 Data

Our corpus included three open-source datasets: FLEURS-R (Ma et al. 2024), Common Voice 17.0 (Ardila et al. 2020), and Ravnursson (Hernandez Mena and Simonsen 2022). All datasets are freely available on Hugging Face, and we provide scripts to download and prepare the datasets (https://github.com/Neclow/mlops_project/tree/main/scripts). Details about the datasets are described in Table 3.1.

Table 3.1: Dataset corpus. All samples are composed of read speech from native speakers, and are accompanied by a transcript in the metadata.

Name	FLEURS-R	Common Voice 17.0	Ravnursson	
Samples (M)	0.3	79	0.07	
Languages	102	124	1	
Size (k hours)	1.3	20.4	0.1	
North Germanic languages		Samples (K)		Total (K)
Danish	3.5	8.1	-	11.6
Faroese	-	-	71.9 *(selected: 16.3)*	16.3
Icelandic	1.0	0.014	-	1.014
Norwegian	2.8	1.2	-	4.0
Swedish	3.5	18.2	-	21.7
Total				54.6

3.1.2 Implementation

A notebook implementation to reproduce the results is available at: https://github.com/Neclow/mlops_project/blob/main/notebooks/data_quality.ipynb.

Our data corpus presented in Table 3.1 contains approximately 55,000 speech samples in five North Germanic languages: Danish, Faroese, Icelandic, Norwegian, and Swedish. Whilst Common Voice 17.0 classifies its Norwegian samples under "Norwegian Nynorsk",

and FLEURS-R under "Norwegian Bokmål", these can be grouped under one "Norwegian" class. Indeed, Nynorsk and Bokmål are official forms of *written* Norwegian, but *Norwegian* is the only official *spoken* language.

Figure 3.1 shows the distribution of samples over the languages of interest. The dataset is highly imbalanced, with the majority class (Swedish; 21.7K samples) having roughly 20 times more samples than the minority class (Icelandic; 1K samples). Moreover, the data is also imbalanced with respect to the number of native speakers. Norwegian has approximately 4-5 million speakers (Rehm and Uszkoreit 2012) is starkly underrepresented, having approximately the same number of speakers as Danish (6 million) and half those of Swedish (9.5 million) (Haugen and Borin 2018), but respectively two and four times less samples in the dataset. On the other hand, Faroese (approx. 50-100,000 speakers) is largely overrepresented (16.3K samples) due to the inclusion of the more specialised Ravnursson dataset (Hernandez Mena and Simonsen 2022).

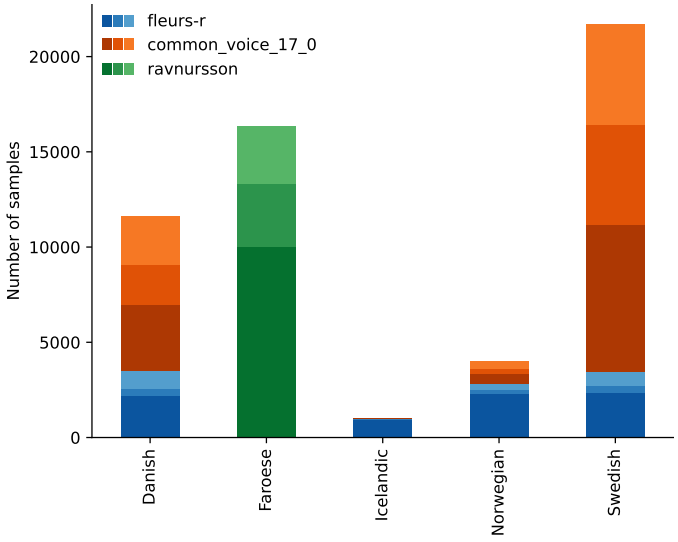

Figure 3.1: Overview of the corpus presented in Table 3.1.

All datasets contain metadata including transcriptions, audio duration and speaker gender. A summary of gender balance over datasets and language is depicted in Figure 3.2. The datasets are roughly well-balanced in Ravnursson and FLEURS-R, whereas Common Voice shows high gender imbalance, as a majority of speakers are males. However, a significant proportion of gender data is missing for that dataset.

Subsequently, we used `Great Expectations` for data quality assurance on the metadata. In particular, we checked that metadata on audio duration, sample rate, file size, and file name were unique and within expected ranges. A list of the checked attributes

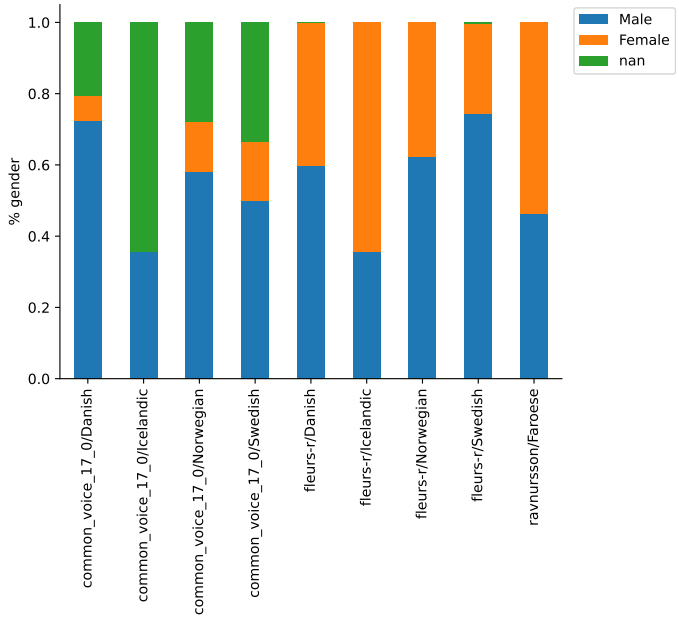

Figure 3.2: Gender balance of the selected data corpus

is presented in Table 3.2.

Surprisingly, audio durations and file sizes were not unique in the Common Voice dataset, which could potentially indicate duplicate files. However, no duplicate rows (which include other features such as a transcript of the audio file and the speaker index) in the Common Voice 17.0 metadata. Thus, the samples are unique to the best of our knowledge. Running such a duplicate check is especially important to ensure that model performance is not biased by data leakage, where duplicate samples could be found in the training and development/testing sets.

In addition, a check on speaker data leakage was run to check that the dataset splits were speaker-independent, revealing no abnormalities. Ensuring that splits are speaker-independent is important to prevent a less obvious form of data leakage, as models could learn from speaker patterns in the training phase. However, for languages present in several datasets FLEURS-R and Common Voice 17.0, we could not fully ascertain speaker independence due to FLEURS-R not having information on the speaker IDs.

Lastly, a key component of our project is to assess and verify the speech recordings. To that end, we used the `torchaudio-squim` (Kumar et al. 2023) model to estimate speech quality and intelligibility. To assess quality, we relied on the same three objective metrics supported by the model: Short-Time Objective Intelligibility (STOI; score range: -1 – 1, higher is better) (Taal et al. 2010), Perceptual Estimation of Speech Quality

Table 3.2: Checked metadata attributes using `Great Expectations`. A report is available at https://github.com/Neclow/mlops_project/blob/main/mlops/great_expectations.

Feature	Attribute	Passed
Duration	Values are not null	✓
	Values are all numerical	✓
	Values are between 1 and 60	✓
	Values are unique	✗
Sample rate	Not null	✓
	Either 16/24/32/48 kHz	✓
File size	Values are not null	✓
	Values are all integers	✓
	Values are between 2 kB and 22 MB	✓
	Values are unique	✗
File name	Values are not null	✓
	Values are unique	✓

(PESQ; score range: -0.5 – 4.5, higher is better) (International Telecommunication Union 2005), and Scale-Invariant Signal-to-Distortion Ratio (SI-SDR in dB) (Le Roux et al. 2019). As shown in Figure 3.3, all recordings generally scored highly across all metrics. FLEURS-R recordings are the most consistent and qualitative, whilst more variations were present in Common Voice. This is not an unexpected result, as FLEURS recordings were controlled (Conneau et al. 2023) (recruitment of native speakers and assessors to check the recordings) and FLEURS-R (Ma et al. 2024) is an improvement of the original dataset though speech restoration. On the other hand, Common Voice 17.0 recordings are crowdsourced, thus making the recording quality more uncertain.

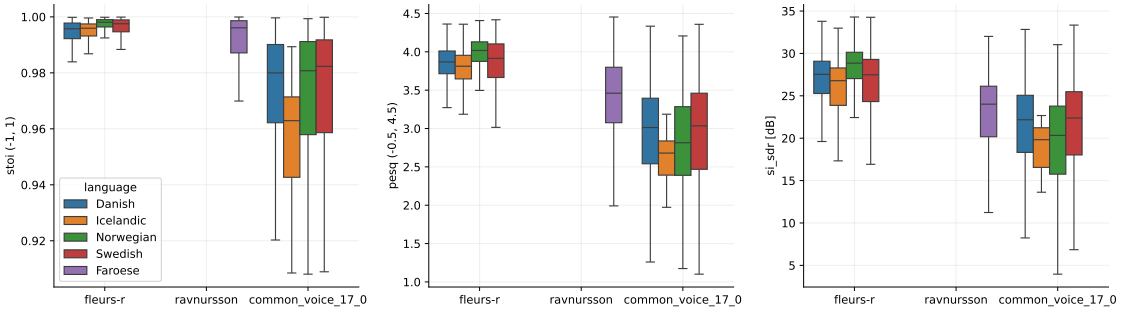

Figure 3.3: Speech quality of the selected data corpus

3.2 Model Development and Versioning

3.2.1 Models

Table 3.3: Language identification model zoo.

Name	Reference	Params (B)
ECAPA-TDNN	Desplanques, Thienpondt, and Demuynck 2020	0.022
Titanet-LID (LangID Ambernet)	Jia et al. 2023	0.029
Whisper (tiny)	Radford et al. 2023	0.04
Whisper (base)	Radford et al. 2023	0.07
XLS-R (LID)	Babu et al. 2022	0.3
MMS (126)	Pratap et al. 2024	1

3.2.2 Implementation

A notebook implementation to reproduce the results is available at: https://github.com/Neclow/mlops_project/blob/main/notebooks/model_development.ipynb. Table 1 shows the performance and efficiency of the model zoo presented in Table 3.3. Using Weights & Biases (W&B), we can easily log model performance and system metrics during training, and retrieve logged data using the W&B application programming interface (API). Moreover, W&B allows the saving of various utilities for model versioning. Indeed, for each run, it registers the last commit before training, logs metadata and saves a snapshot to restore the model architecture and weights.

Table 3.4: Performance and efficiency of the models described in Table 3.3, benchmarked against the Table 3.1 corpus. Note: the runs' system metrics (e.g., GPU memory) are not shown, as W&B does not take into account whether multiple processes were run concurrently. See https://docs.wandb.ai/guides/app/features/system-metrics.

W&B run ID	model	Test accuracy (%)	Training runtime (min)	GFLOPS (Forward pass for 1 s of audio)
igyrfwxv	XLS-R (LID)	**97.7**	41.7	35.6
16ryf90y	MMS (126)	97.5	77.5	99.5
cnidpi03	Titanet-LID	92.5	**19.5**	5.0
znd36c9w	Whisper (base)	91.2	34.2	97.0
d8a1buho	ECAPA-TDNN	90.0	24.0	**3.8**
7zi2am3w	Whisper (tiny)	87.9	34.4	40.6

We observe that all models classified the audio samples from Table 3.1 with high accuracy. Two models stand out in particular: XLS-R (LID) achieved the highest performance (97.7% test accuracy) with reasonable training runtime and inference speed, while Titanet-LID performed reasonably (92.4% test accuracy) with the shortest training runtime (19.5 min) and second fastest inference speed. Depending on budget and compute constraints, choosing either of these two models may be optimal, with XLS-R (LID) recommended for applications prioritizing maximum accuracy and Titanet-LID suited for scenarios where rapid deployment and low computational resources are paramount.

3.2.3 Model versioning

All runs can be viewed at: ttps://wandb.ai/neclow/mlops_project_eval_nort3160. A critical feature of W&B for model versioning is its API (`wandb.Api`) to retrieve information from training experiments. Calling the API on a W&B project using a run ID will retrieve the run's performance and system metrics (`run.history`) logged throughout the experiment, summary metrics such as test set metrics and runtime (`run.summary`), and printed logs (`run.file('output.log')`).

For model reproducibility, W&B saved the git state and the command used to run the experiment. For instance, to reproduce the XLS-R (LID) run, one can run:

```
git clone https://github.com/Neclow/mlops_project.git
git checkout -b "daily-music-9" ad179f780c71e76f571a74cc0120aa20daf2b854
python -m core.eval nort3160 facebook/wav2vec2-xls-r-300m speech --transform --max-duration 10
```

Further, W&B also saves a snapshot of the environment used to perform the run. For Python projects, this will be a *requirements.txt* file that can be retrieved using the `file` function. Finally, whilst model artefacts including checkpoints and optimizer states are automatically saved locally, they can be logged in the W&B cloud (e.g., using the `log_model` flag in Lightning's `WandLogger`).

3.3 Model Deployment

A notebook implementation to reproduce the results is available at: https://github.com/Neclow/mlops_project/blob/main/notebooks/model_deployment.ipynb. Using our bank of tracked runs described in Section 3.2, we can trivially load a model with its training checkpoint from its W&B run ID. We then used `FastAPI` and `uvicorn` to implement a simple API, where a user can upload an audio file and get the model's

language prediction and confidence. Future work could ensure that the input is a valid audio file and that its duration lies within pre-defined bounds (e.g., 1 to 30 seconds).

3.4 Monitoring and Logging

A notebook implementation to reproduce the results is available at: https://github.com/Neclow/mlops_project/blob/main/notebooks/model_monitoring.ipynb. In our MLOps pipeline, effective monitoring and logging are essential to maintain model reliability and performance, particularly in managing data drift. Data drift occurs when the statistical properties of the data in production change relative to the training data, which can degrade model accuracy.

3.4.1 Simulating Data Drift in the Project

To study and manage data drift, we simulated it by introducing noise to the audio data. This simulation should replicates real-world issues, such as background noise and device variability, and allows us to systematically monitor the model's response to drift. The process involved:

1. **Feature Extraction**: We processed audio data to extract relevant features, such as Mel-frequency cepstral coefficients (MFCCs), spectral centroid, spectral contrast, and zero-crossing rate.

2. **Adding Noise**: To simulate drift, Gaussian noise was added to the audio data before feature extraction.

3.4.2 Data Drift Detection Setup with Evidently AI

To monitor data drift, we used `Evidently` AI's `DataDriftReport`, which compares feature distributions between the reference dataset (original training data) and the current dataset (new production data). The setup included: The reference dataset (original data) and current dataset (drifted data) were loaded as Pandas DataFrames. These DataFrames were passed to Evidently's `DataDriftReport` to compare feature distributions across both datasets.

4 Evaluation and Reflection

The prototype demonstrates key aspects of a DataOps/MLOps pipeline, including:

- **Data Quality Assurance**: The implemented data validation checks helped to identify and address data issues before they could affect model performance, ensuring a cleaner dataset.
- **Model Development and Versioning**: Version control for models allowed us to track different iterations and assess improvements systematically.
- **Deployment and Monitoring**: The deployment strategy proved effective for the application context. The monitoring tools provided valuable insights into model performance in real-time, detecting data drift and facilitating prompt error identification.

Despite these strengths, the prototype has some limitations due to limited resources, certain components were implemented on a small scale, which may not easily translate to larger, production-grade systems.

This project underscored the importance of automation in DataOps/MLOps, particularly for maintaining consistency and reducing the time spent on repetitive tasks. Implementing data versioning and model tracking provided valuable insights into the importance of organized, reproducible workflows. The experience also highlighted the value of robust monitoring tools to maintain model quality over time.

4.0.1 Potential Improvements

Future improvements could focus on enhancing orchestration, perhaps by integrating a more advanced workflow management tool to simplify automation and minimize manual intervention. Additionally, exploring more scalable solutions for data processing and storage would help prepare this pipeline for handling larger datasets.

4.0.2 Scalability Considerations

To scale this project to a real-world scenario, several aspects would require attention:

- **Enhanced Infrastructure**: Leveraging cloud resources could allow for real-time processing and larger data volumes.
- **Advanced Monitoring Systems**: Implementing robust monitoring and alerting systems would ensure models perform optimally over time and adapt to data changes.
- **Continuous Training**: Establishing a reliable continuous training loop would ensure the models remain accurate and relevant.

References

Achiam, Josh et al. (2024). *GPT-4 Technical Report*. arXiv: 2303.08774 [cs.CL].

Anthropic (2023). *Introducing Claude*. https://www.anthropic.com/news/introducing-claude. Accessed: 2024-10-03.

Ardila, R. et al. (2020). "Common Voice: A Massively-Multilingual Speech Corpus". In: *Proc. LREC*, pp. 4211–4215.

Babu, Arun et al. (2022). "XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale". In: *Proc. Interspeech*, pp. 2278–2282.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *International Conference on Learning Representations*. URL: https://doi.org/10.48550/arXiv.1409.0473.

Barrault, Loic et al. (2023). "SeamlessM4T-Massively Multilingual & Multimodal Machine Translation". In: *arXiv preprint arXiv:2308.11596*.

Conneau, Alexis et al. (2023). "Fleurs: Few-shot learning evaluation of universal representations of speech". In: *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 798–805.

Desplanques, Brecht, Jenthe Thienpondt, and Kris Demuynck (2020). "ECAPA-TDNN: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification". In: *arXiv preprint arXiv:2005.07143*.

Dubey, Abhimanyu et al. (2024). *The Llama 3 Herd of Models*. arXiv: 2407.21783 [cs.AI].

DVC (n.d.). https://dvc.org/doc.

Feast (2024). https://docs.feast.dev/master. Accessed: 2024-10-25.

Georgiev, Petko et al. (2024). *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. arXiv: 2403.05530 [cs.CL].

Google Deepmind (2023). *Gemini Models*. https://deepmind.google/technologies/gemini. Accessed: 2024-10-03.

Haugen, Einar and Lars Borin (2018). "Danish, Norwegian and Swedish". In: *The world's major languages*. Routledge, pp. 127–150.

Hernandez Mena, Carlos Daniel and Annika Simonsen (2022). *Ravnursson Faroese Speech and Transcripts*. URL: http://hdl.handle.net/20.500.12537/276.

International Telecommunication Union (2005). *P.862.2: Wideband extension to recommendation P.862 for the assessment of wideband telephone networks and speech codecs*. Tech. rep.

Jia, F. et al. (2023). "A Compact End-to-End Model with Local and Global Context for Spoken Language Identification". In: *Proc. Interspeech*, pp. 5321–5325. DOI: 10.21437/Interspeech.2023-2310.

Kumar, Anurag et al. (2023). "Torchaudio-squim: Reference-less speech quality and intelligibility measures in torchaudio". In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 1–5.

Le Roux, Jonathan et al. (2019). "SDR–half-baked or well done?" In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 626–630.

Ma, Min et al. (2024). "FLEURS-R: A Restored Multilingual Speech Corpus for Generation Tasks". In: *arXiv preprint arXiv:2408.06227*.

OpenAI (2022). *Introducing ChatGPT*. https://openai.com/index/chatgpt. Accessed: 2024-10-03.

– (2024). *Hello GPT-4o*. https://openai.com/index/hello-gpt-4o. Accessed: 2024-10-03.

Pratap, Vineel et al. (2024). "Scaling speech technology to 1,000+ languages". In: *Journal of Machine Learning Research* 25.97, pp. 1–52.

Radford, Alec et al. (2023). "Robust speech recognition via large-scale weak supervision". In: *ICML*. PMLR, pp. 28492–28518.

Rehm, G. and H. Uszkoreit (2012). *The Norwegian Language in the Digital Age: Nynorskversjon*. White Paper Series. Springer Berlin Heidelberg. ISBN: 9783642314339.

Taal, Cees H et al. (2010). "A short-time objective intelligibility measure for time-frequency weighted noisy speech". In: *2010 IEEE international conference on acoustics, speech and signal processing*. IEEE, pp. 4214–4217.

Touvron, Hugo, Thibaut Lavril, et al. (2023). *LLaMA: Open and Efficient Foundation Language Models*. arXiv: 2302.13971 [cs.CL].

Touvron, Hugo, Louis Martin, et al. (2023). *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv: 2307.09288 [cs.CL].

Üstün, Ahmet et al. (Aug. 2024). "Aya Model: An Instruction Finetuned Open-Access Multilingual Language Model". In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Lun-Wei Ku, Andre Martins, and Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, pp. 15894–15939. DOI: 10.18653/v1/2024.acl-long.845. URL: https://aclanthology.org/2024.acl-long.845.

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Yan, Jianhao et al. (2024). *GPT-4 vs. Human Translators: A Comprehensive Evaluation of Translation Quality Across Languages, Domains, and Expertise Levels*. arXiv: 2407.03658 [cs.CL].