MLPC 2024 Task 4: Challenge

Katharina Hoedt, Verena Praher, Paul Primus, Florian Schmid

Institute of Computational Perception
Johannes Kepler University Linz
May 27, 2024

Context

Our overall objective for this year's project is to develop a system that can detect speech commands in audio recordings. The purpose of this system is to allow users to control different devices in a smart home using speech commands. Read the *Project Description* on the Moodle page for a detailed project outline.

Developing such a speech command recognition system with machine learning entails the following steps:

1. record a training set of keywords and non-keyword sounds
2. compute a set of candidate audio features
3. perform a thorough analysis of the features and select useful features
4. train and evaluate a range of classifiers and their corresponding hyperparameters; find the ones that can distinguish between keywords and unrelated sounds
5. apply and evaluate the trained classifiers for detecting speech commands in everyday scenarios and select the model that works the best.

After joining our forces to collect a data set of appropriate size, we computed a variety of audio features for you (Steps 1 & 2). In the previous assignments (Steps 3 & 4), you then conducted a thorough analysis of the data set and the features and trained and evaluated three classifiers of your choice. The final task (Step 5) now is to optimize your classifier(s) for the application scenario outlined in the project description. As for the previous assignment, you will work on this task with your team.

Task Outline (27+3 points)

Deadline: June 19

Your main objective will be to tune your classifiers to detect the temporal occurrence of complete speech commands (consisting of a device keyword **and** an action keyword) in a continuous stream of audio.

To aid the development of such systems, the (fictitious) company SmartVoiceControl has released an additional data set with domestic soundscapes (folder: development_scenes). This set contains audio recordings of one or multiple speech commands and their corresponding temporal annotations (e.g., onsets and offsets; see

development_scene_annotation.csv). Use this dataset to assess your system's detection performance or (optionally) for training.

SmartVoiceControl has also released a specification of the costs associated with correct and incorrect predictions. This new evaluation metric is better aligned with the actual goals of the application and deviates significantly from the purely classification-based evaluation used in the previous assignment. The proposed cost function distinguishes between four types of events:

- True Positives: A speech command was issued and correctly detected by the system. A detection is considered correct if the **predicted timestamp is within the annotation's onset and offset.** The associated **cost is -1**.
- False Negatives: A speech command was issued, but the system did not detect a command. This is annoying but often a result of sloppy pronunciation; users will adjust their speech to the system over time. The associated **cost is 0.5**
- False Positives: The system wrongfully detects a speech command. Depending on the command, the consequences of wrongful detections range from spooky and bothersome to potentially dangerous (luckily, there will be some additional safeguards to avoid the worst). The associated costs, therefore, depend on the wrongfully detected command:
    - {"Fernseher", "Licht", "Radio", or "Staubsauger"} + {"an" or "aus"}: **2**
    - {"Heizung" or "Lüftung"} + {"an" or "aus"}: **3**
    - {"Ofen" or "Alarm" } + {"an" or "aus"}: **4**
- Cross-Triggers: Cross-Triggers are a special type of false positives. They occur when a user intends to interact with the system by issuing a speech command, but the system does not recognize the correct command. This lowers the perceived quality of the voice control system and leads to user frustration. However, Cross-Triggers are less expensive compared to general false positives, as the user receives feedback from the system and can manually intervene if the system makes a critical mistake (e.g., detecting "Ofen an", or "Alarm aus" instead of the intended speech command). In general, Cross-Triggers result in a **cost of 1**. Luckily, not all Cross-Triggers are that expensive; the system automatically corrects incorrect action keywords. For example, if the radio ("Radio") is on and the user issues "Radio aus" (i.e., radio off), but the system detects "Radio an" (i.e., radio on), the system will automatically correct the classification to "Radio aus" (i.e., radio off). The cost for incorrect action keywords with correct device keywords is, therefore, **0.1**.

The final performance (in terms of costs) will be measured on a private test set (i.e., scenes with unseen speakers) for which no temporal annotations are publicly available (folder: test_scenes; available on the 12th of June). The system with the **lowest costs** on this private test set will win the (fictitious) contract with SmartVoiceControl. To this end, **upload a CSV file** to Moodle which contains the detected speech commands and their corresponding timestamps. The CSV file must have the following structure (including the header):

filename	command	timestamp
5bcf8028f5	Radio aus	4.102
…	…	…
92ba262ee	Licht aus	12.012

Some important notes on the format of this CSV file:

- the header must be included
- use "," (comma) as delimiter
- filenames are without path and file ending
    - wrong: 'scenes\5bcf8028f5.mat'
    - correct: '5bcf8028f5'
- commands consist of device and action keywords separated by a white space
- the timestamp is in seconds (float)

The development set from the previous task (containing 1.1-second snippets of spoken words and other sounds) and an additional development set (consisting of scenes that contain speech commands with temporal annotations) are available on our Moodle page. The new data set also includes two metadata files:

- development_scenes.csv lists all files in the development set and indicates the identity of the speaker. Use the speaker ID indicated in this file to avoid data leakage when creating your custom training and validation split.
- development_scene_annotations.csv lists all events and their corresponding onsets and offsets

Our Moodle page also provides a Python script that computes the cost for your predictions on your custom validation split of the new development dataset according to the previously mentioned specifications. Install a recent Python version and use the following command to evaluate a CSV file containing the detected speech commands:

```
python score_predictions.py \
--predictions=predictions.csv \
--annotations=annotations.csv
```

- predictions.csv is a file containing your predictions on the development set (the structure is outlined in the table above)
- annotations.csv holds the ground truth events. The complete ground truth for the new development set is available via Moodle; remove the rows that correspond to your custom training split.
- use the `--check_format` flag to check the structure of your predictions file before uploading it to Moodle
- use the `--dump_result` flag to store the evaluation result as a JSON file

For your experiments, you may use any machine learning package you want, such as scikit-learn for Python or built-in toolboxes for Matlab. Precomputed features (for Python and

Matlab) and the raw waveforms of the new development set are available on our Moodle page.

For training your classifiers, you may want to exploit that neither all the features nor all the training examples are equally useful. Especially when working on (not-so-powerful) laptops, it can pay off to subsample the training data or select a set of non-redundant features to save computation time.

Participating in Task 4 is a requirement to pass the course.

Written Report (max. 27 points)

For the first part of the submission, you will have to write a report based on the **template** provided on Moodle (item *MLPC Report Template*). In this report, you will address the following questions/ aspects:

1. **Establish a naive Baseline system (i.e., a baseline without a classifier)**
    a. Describe what a naive baseline system for detecting speech commands in scenes could look like. How much cost can we expect from it?
2. **A starting point:** Build a simple speech command detection system (e.g., with one of the keyword classifiers you trained in the previous phase of the project).
    a. Describe how your system was used to detect keywords in the longer domestic recordings (windowing, hop size, etc.)
    b. How did you threshold and combine keyword predictions to detect full speech commands? Describe any heuristics or post-processing strategies used.
    c. What strategies did you apply to minimize the task-specific cost function?
    d. Describe your evaluation setup and provide evaluation results on (parts of) the public, annotated scenes.
    e. Does your simple speech command detection system achieve lower costs compared to the naive Baseline system you established in 1.? If not, describe possible issues.
3. **Improvements:** Investigate at least three diverse strategies to improve your starting point (e.g., via hyperparameter tuning, ensembling, data augmentation, cost-specific tuning, post-processing, using data from the new development set for training, etc.)
    a. For each of these strategies: Describe the main ideas you had and hypotheses you tried, as well as their outcome after experimentally verifying or falsifying the hypotheses (via validation on (parts of) the development set). Do not hesitate to report negative outcomes.
4. **Critical Reflection:**
    a. Do you think your final system could be deployed in a real-world application? In your opinion, which aspects of the project or your system would need to be adapted to fulfill possible real-world requirements?

In addition to addressing these questions, you will also have to add a **statement of the contributions** of all team members, as indicated in the template. The report must not exceed 6 pages, of which at most 4 should be **text**.

Slide Set (3 points)

The second part of the submission is a short presentation. This complementary slide set serves to present selected results clearly and concisely to your fellow course participants. More precisely, **describe the general architecture of your system** and **the most interesting hypotheses** you investigated and their **outcomes**.

The **upper limit for the number of slides** you should prepare is **5** (+ one additional title slide that should contain your group name and the member names).

Grading

The written reports for this task and its subtasks are evaluated according to the following criteria

- **Thoroughness & Completeness**: Have you thought about the problem and answered every question?
- **Clarity**: Are the ideas, features, algorithms, and results described clearly? Based on your descriptions, could the reader reconstruct your experiments?
- **Presentation**: Did you select an appropriate way of communicating your results, e.g., did you use meaningful plots where helpful?
- **Correctness**: Is the proposed procedure/experiment sound, correct?
- **Punctuality**: The reports must be submitted on time. Any delay will result in reduced grades. Specifically, submitting on June 20 will deduct 1/3 of the points, submitting on June 21 will deduct 2/3 of the points, and submissions on June 22 or later will be rejected.

For the slide set, you will be awarded points if you have a valid set (i.e. within the slide limit) submitted for the assigned topic.

Summary

- **Completing Task 4 is a requirement to pass this course.**
- **Upload** your **predictions** for the private test set to Moodle (*Step 2: Submit your predictions*). Check your predictions before uploading them to Moodle (use `python score_predictions.py --predictions=predictions.csv --check_format`)
- Look at the given questions and answer **all** of them appropriately in a written report. Make sure to use the **report template** provided to you via Moodle. Adhere to the given **page limit** (max. 6 pages where at most 4 pages can be text) and include a statement about the contributions of all team members.
- Create a set of slides describing your solution and some of the hypotheses you investigated. Make sure to adhere to the **slide limit** for this step as well (max. 5 + 1 title slide).
- Upload the written report as well as your slides to Moodle (*Step 3: Submit your team's Task 4 reports*) by **June 19th**.
- You will get a maximum number of 27 points for your written report and 3 points for the slide set.