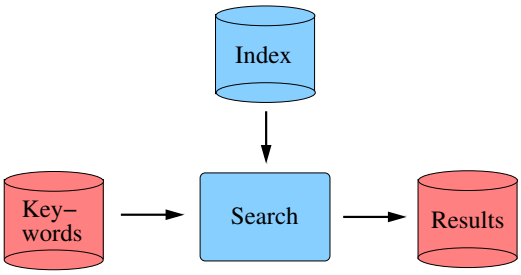
COM4511 Speech Technology - Practical Exercise - Keyword Search

Anton Ragni

1 Background

The aim of this task is to build and investigate the simplest form of a keyword search (KWS) system allowing to find information in large volumes of spoken data. Figure below shows an example of a typical KWS system which consists of an index and a search module. The index provides a compact representation of spoken data. Given a set of keywords, the search module



queries the index to retrieve all possible occurrences ranked according to likelihood. The quality of a KWS is assessed based on how accurately it can retrieve all true occurrences of keywords.

A number of index representations have been proposed and examined for KWS. Most popular representations are derived from the output of an automatic speech recognition (ASR) system. Various forms of output have been examined. These differ in terms of the amount of information retained regarding the content of spoken data. The simplest form is the most likely word sequence or *1-best*. Additional information such as start and end times, and recognition confidence may also be provided for each word. Given a collection of 1-best sequences, the following index can be constructed

$$\begin{array}{cccc} w_1 & (f_{1,1}, s_{1,1}, e_{1,1}) & \dots & (f_{1,n_1}, s_{1,n_1}, e_{1,n_1}) \\ w_2 & (f_{1,1}, s_{1,1}, e_{1,1}) & \dots & (f_{1,n_1}, s_{1,n_1}, e_{1,n_1}) \\ & & \vdots & \\ w_N & (f_{N,1}, s_{N,1}, e_{N,1}) & \dots & (f_{N,n_N}, s_{N,n_N}, e_{N,n_N}) \end{array} \tag{1}$$

where w_i is a word, n_i is the number of times word w_i occurs, $f_{i,j}$ is a file where word w_i occurs for the j-th time, $s_{i,j}$ and $e_{i,j}$ is the start and end time. Searching such index for single word keywords can be as simple as finding the correct row (e.g. k) and returning all possible tuples $(f_{k,1}, s_{k,1}, e_{k,1}), \dots, (f_{k,n_k}, s_{k,n_k}, e_{k,n_k})$.

The search module is expected to retrieve all possible keyword occurrences. If ASR makes no mistakes such module can be created rather trivially. To account for possible retrieval errors, the search module provides each potential occurrence with a relevance score. Relevance scores reflect confidence in a given occurrence being relevant. Occurrences with extremely low relevance scores may be eliminated. If these scores are accurate each eliminated occurrence will decrease the number of false alarms. If not then the number of misses will increase. What exactly an extremely low score is may not be very easy to determine. Multiple factors may affect a relevance score: confidence score, duration, word confusability, word context, keyword length. Therefore, simple relevance scores, such as those based on confidence scores, may have a wide dynamic range and may be incomparable across different keywords. In order to ensure that relevance scores are comparable among different keywords they need to be *calibrated*. A simple calibration scheme is called sum-to-one (STO) normalisation

$$\hat{r}_{i,j} = \frac{r_{i,j}^{\gamma}}{\sum_{k=1}^{n_i} r_{i,k}^{\gamma}} \tag{2}$$

where $r_{i,j}$ is an original relevance score for the j-th occurrence of the i-th keyword, γ is a scale enabling to either sharpen or flatten the distribution of relevance scores. More complex schemes have also been examined. Given a set of occurrences with associated relevance scores, there are several options available for eliminating spurious occurrences. One popular approach is *thresholding*. Given a global or keyword specific threshold any occurrence falling under is eliminated. Simple calibration schemes such as STO require thresholds to be estimated on a development set and adjusted to different collection sizes. More complex approaches such as Keyword Specific Thresholding (KST) yield a fixed threshold across different keywords and collection sizes.

Accuracy of KWS systems can be assessed in multiple ways. Standard approaches include precision (proportion of relevant retrieved occurrences among all retrieved occurrences) and recall (proportion of relevant retrieved occurrences among all relevant occurrences), mean average precision and term weighted value. A collection of precision and recall values computed for different thresholds yields a precision-recall (PR) curve. The area under PR curve (AUC) provides a threshold independent summative statistics for comparing different retrieval approaches. The mean average precision (mAP) is another popular, threshold-independent, precision based metric. Consider a KWS system returning 3 correct and 4 incorrect occurrences arranged according to relevance score as follows: ✓ , ✗ , ✗ , ✓ , ✓ , ✗ , ✗ , where ✓ stands for correct occurrence and ✗ stands for incorrect occurrence. The average precision at each rank (from 1 to 7) is $\frac{1}{1}$, $\frac{0}{2}$, $\frac{0}{3}$, $\frac{2}{4}$, $\frac{3}{5}$, $\frac{0}{6}$, $\frac{0}{7}$. If the number of true correct occurrences is 3, the mean average precision for this keyword 0.7. A collection-level mAP can be computed by averaging keyword specific mAPs. Once a KWS system operates at a reasonable AUC or mAP level it is possible to use term weighted value (TWV) to assess accuracy of thresholding. The TWV is defined by

$$\text{TWV}(\mathcal{K},\theta) = 1 - \left(\frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} P_{\texttt{miss}}(k,\theta) + \beta P_{\texttt{fa}}(k,\theta) \right) \tag{3}$$

where $k \in \mathcal{K}$ is a keyword, $P_{\texttt{miss}}$ and $P_{\texttt{fa}}$ are probabilities of miss and false alarm, β is a penalty assigned to false alarms. These probabilities can be computed by

$$P_{\texttt{miss}}(k,\theta) = \frac{N_{\texttt{miss}}(k,\theta)}{N_{\texttt{correct}}(k)} \tag{4}$$

$$P_{\texttt{fa}}(k,\theta) = \frac{N_{\texttt{fa}}(k,\theta)}{N_{\texttt{trial}}(k)} \tag{5}$$

where $N_{<\texttt{event}>}$ is a number of events. The number of trials is given by

$$N_{\texttt{trial}}(k) = T - N_{\texttt{correct}}(k) \tag{6}$$

where T is the duration of speech in seconds.

2 Objective

Given a collection of 1-bests, write a code that retrieves all possible occurrences of keyword list provided. Describe the search process including index format, handling of multi-word keywords, criterion for matching, relevance score calibration and threshold setting methodology. Write a code to assess retrieval performance using reference transcriptions according to AUC, mAP and TWV criteria using $\beta = 20$. Comment on the difference between these criteria including the impact of parameter β. Start and end times of hypothesised occurrences must be within 0.5 seconds of true occurrences to be considered for matching.

3 Marking scheme

Two critical elements are assessed: retrieval (65 marks) and assessment (35 marks). *Note:* Even if you cannot complete this task as a whole you can certainly provide a description of what you were planning to accomplish.

1. **Retrieval**

   1.1 **Index** Write a code that can take provided CTM files (and any other file you deem relevant) and create indices in your own format. For example, if Python language is used then the execution of your code may look like

   ```
   python index.py dev.ctm dev.index
   ```

   where `dev.ctm` is an CTM file and `dev.index` is an index.
   Marks are distributed based on handling of multi-word keywords

- Efficient handling of single-word keywords **(20 marks)**
- No ability to handle multi-word keywords **(extra 0 marks)**
- Inefficient ability to handle multi-word keywords **(extra 10 marks)**
- Or efficient ability to handle multi-word keywords **(extra 20 marks)**

1.2 **Search** Write a code that can take the provided keyword file and index file (and any other file you deem relevant) and produce a list of occurrences for each provided keyword. For example, if Python language is used then the execution of your code may look like

```
python search.py dev.index keywords dev.occ
```

where `dev.index` is an index, `keywords` is a list of keywords, `dev.occ` is a list of occurrences for each keyword.

Marks are distributed based on handling of multi-word keywords

- Efficient handling of single-word keywords **(4 marks)**
- No ability to handle multi-word keywords **(extra 0 marks)**
- Inefficient ability to handle multi-word keywords **(extra 2 marks)**
- Or efficient ability to handle multi-word keywords **(extra 4 marks)**

1.3 **Description** Provide a technical description of the following elements

- Index file format **(3 marks)**
- Handling multi-word keywords **(3 marks)**
- Criterion for matching keywords to possible occurrences **(2 marks)**
- Search process **(2 marks)**
- Score calibration **(3 marks)**
- Threshold setting **(3 marks)**

2. **Assessment** Write a code that can take the provided keyword file, the list of found keyword occurrences and the corresponding reference transcript file in STM format and compute the metrics described in the Background section. For instance, if Python language is used then the execution of your code may look like

```
python <metric>.py keywords dev.occ dev.stm
```

where `<metric>` is one of precision-recall, mAP and TWV, `keywords` is the provided keyword file, `dev.occ` is the list of found keyword occurrences and `dev.stm` is the reference transcript file.

Hint: In order to simplify assessment consider converting reference transcript from STM file format to CTM file format. Using indexing and search code above obtain a list of true occurrences. The list of found keyword occurrences then can be assessed more easily by comparing it with the list of true occurrences rather than the reference transcript file in STM file format.

2.1 **Implementation**

- **AUC (4 marks)** Integrate an existing implementation of AUC computation into your code. For example, for Python language such implementation is available in `sklearn` package.
- **mAP (4 marks)** Write your own implementation or integrate any freely available.
- **TWV (4 marks)** Write your own implementation or integrate any freely available.

2.2 **Description**

- **AUC (7 marks)** Plot precision-recall curve. Report AUC value . Discuss performance in the high precision and low recall area. Discuss performance in the high recall and low precision area. Suggest which keyword search applications might be interested in a good performance specifically in those two areas (either high precision and low recall, or high recall and low precision).
- **mAP (5 marks)** Report mAP value. Report mAP value for each keyword length (1-word, 2-words, etc.). Compare and discuss differences in mAP values.
- **TWV (10 marks)** Report TWV value. Report TWV value for each keyword length (1-word, 2-word, etc.). Compare and discuss differences in TWV values. Plot TWV values for a range of threshold values. Report maximum TWV value or MTWV. Report actual TWV value or ATWV obtained with a method used for threshold selection.
- **Comparison (2 marks)** Describe the use of AUC, mAP and TWV in the development of your KWS approach. Compare these metrics and discuss their advantages and disadvantages.

4 Hand-in procedure

All outcomes, however complete, are to be submitted jointly in a form of a package file (`zip/tar/gzip`) that includes directories for each task which contain the associated required files. Submission will be performed via MOLE.

5 Resources

Three resources are provided for this task:

- 1-best transcripts in NIST CTM file format (`dev.ctm`,`eval.ctm`). The CTM file format consists of multiple records of the following form

  ```
  <F> <H> <T> <D> <W> <C>
  ```

  where `<F>` is an audio file name, `<H>` is a channel, `<T>` is a start time in seconds, `<D>` is a duration in seconds, `<W>` is a word, `<C>` is a confidence score. Each record corresponds to one recognised word. Any blank lines or lines starting with `;;` are ignored. An excerpt from a CTM file is shown below

  ```
  7654 A 11.34 0.2 YES 0.5
  7654 A 12.00 0.34 YOU 0.7
  7654 A 13.30 0.5 CAN 0.1
  ```

- Reference transcript in NIST STM file format (`dev.stm`, `eval.stm`). The STM file format consists of multiple records of the following form

  ```
  <F> <H> <S> <T> <E> <L> <W>...<W>
  ```

  where `<S>` is a speaker, `<E>` is an end time, `<L>` topic, `<W>...<W>` is a word sequence. Each record corresponds to one manually transcribed segment of audio file. An excerpt from a STM file is shown below

  ```
  2345 A 2345-a 0.10 2.03 <soap> uh huh yes i thought
  2345 A 2345-b 2.10 3.04 <soap> dog walking is a very
  2345 A 2345-a 3.50 4.59 <soap> yes but it's worth it
  ```

  Note that exact start and end times for each word are not available. Use uniform segmentation as an approximation. The duration of speech in `dev.stm` and `eval.stm` is estimated to be 57474.2 and 25694.3 seconds.

- Keyword list `keywords`. Each keyword contains one or more words as shown below

  ```
  PROUD OF YOURSELF
  DON
  TWENTY THOUSAND POUNDS
  LIB DEM
  LIABILITY
  ```

Use development set files (`dev.ctm` and `dev.stm`) for developing your approaches. Use evaluation set files (`eval.ctm` and `eval.stm`) for evaluation only.

6 Reading List

[1] M. Saraclar, R. Sproat, "Lattice-Based Search for Spoken Utterance Retrieval", Proc. NAACL, 2004. Available online: `https://www.aclweb.org/anthology/N04-1017.pdf`

[2] D. Can, M. Saraclar, "Lattice Indexing for Spoken Term Detection", IEEE Tran Audio, Speech and Language, 2011. Available on-line: `https://wiki.inf.ed.ac.uk/twiki/pub/CSTR/ListenSemester2201314/taslp_2011.pdf`

[3] D. Karakos *et al*, "Score normalization and system combination for improved keyword spotting", Proc. ASRU, 2013. Available online: `https://www.fit.vutbr.cz/research/groups/speech/publi/2013/karakos_asru2013_0000210.pdf`