COM4511 Speech Technology - Practical Exercise - Language Models

Anton Ragni

1 Background

Statistical language models play an important role in many speech technology and natural language processing tasks. To be useful, these models need to be trained on large quantities of representative text data. Such quantities, however, are rarely available for any given task, which leads to non-robust parameter estimates. In order to improve robustness in limited resource conditions, it is often possible to augment available data with large quantities of less representative data. Such additional data may come from a range of diverse sources. As such it is often unclear which of the available sources provides more useful data. Language model interpolation provides a convenient methodology for improving robustness of language models in limited resource conditions and providing interpretable information about usefulness of diverse data sources. Knowing which data source is most useful can help to further improve the underlying model.

N-gram models are one of the most popular types of statistical language models. These models estimate probabilities of sentences $\boldsymbol{w}_{1:K} = w_1, \dots, w_K$ by

$$P(\boldsymbol{w}_{1:K}) \triangleq \prod_{k=1}^{K} P(w_k | w_{k-1}, \dots, w_{k-n+1}) \tag{1}$$

where $P(w_k | w_{k-1}, \dots, w_{k-n+1})$ is a probability of n-gram (tuple of n words) $w_{k-n+1}, \dots, w_{k-1}, w_k$. The quality of language models is assessed using perplexity (PPL). The perplexity of sentence $\boldsymbol{w}_{1:K}$ is

$$\text{PPL}(\boldsymbol{w}_{1:K}) = \exp\left(-\frac{1}{K}\log(P(\boldsymbol{w}_{1:K}))\right) \tag{2}$$

The perplexity of a data set $\mathcal{W} = \{\boldsymbol{w}^{(1)}, \dots, \boldsymbol{w}^{(S)}\}$ is

$$\text{PPL}(\mathcal{W}) = \exp\left(-\frac{1}{|\mathcal{W}|}\sum_{s=1}^{S}\log(P(\boldsymbol{w}^{(s)}))\right) \tag{3}$$

where $|\mathcal{W}|$ is the number of words in $\mathcal{W}$.

Given a set of n-gram language models trained on diverse sources of data, the probabilities from all models can be combined

$$P(w_k^{(s)} | w_{k-1}^{(s)}, \dots, w_{k-n+1}^{(s)}; \boldsymbol{\lambda}) = \sum_{l=1}^{L} \lambda_l P_l(w_k^{(s)} | w_{k-1}^{(s)}, \dots, w_{k-n+1}^{(s)}) \tag{4}$$

where $P_l(w_k^{(s)} | w_{k-1}^{(s)}, \dots, w_{k-n+1}^{(s)})$ and λ_l are n-gram probability and interpolation weight for language model l. Interpolation weights are expected to satisfy standard statistical constraints

$$\lambda_l \geq 0 \quad \text{for all } l \tag{5}$$

$$\sum_{l=1}^{L} \lambda_l = 1 \tag{6}$$

Interpolation weights can be set automatically by minimising perplexity on some held-out data $\mathcal{W}$. Let $\boldsymbol{\lambda}^{(0)}$ be the initial set of weights. The probability of n-gram $w_{k-n+1}^{(s)}, \dots, w_{k-1}^{(s)}, w_k^{(s)}$ is

$$P(w_k^{(s)} | w_{k-1}^{(s)}, \dots, w_{k-n+1}^{(s)}; \boldsymbol{\lambda}^{(0)}) = \sum_{l=1}^{L} \lambda_l^{(0)} P_l(w_k^{(s)} | w_{k-1}^{(s)}, \dots, w_{k-n+1}^{(s)}) \tag{7}$$

The perplexity of interpolated language model $\boldsymbol{\lambda}^{(0)}$ on data set $\mathcal{W}$ is

$$\text{PPL}(\mathcal{W};\boldsymbol{\lambda}^{(0)}) = \exp\left(-\frac{1}{|\mathcal{W}|}\sum_{s=1}^{S}\log(P(\boldsymbol{w}^{(s)};\boldsymbol{\lambda}^{(0)}))\right) = \exp\left(-\frac{1}{|\mathcal{W}|}\sum_{s=1}^{S}\sum_{k=1}^{|\boldsymbol{w}^{(s)}|}\log(P(w_k^{(s)}|w_{k-1}^{(s)},\ldots,w_{k-n+1}^{(s)};\boldsymbol{\lambda}^{(0)}))\right) \tag{8}$$

where $|\boldsymbol{w}^{(s)}|$ is the number of words in the s^{th} sentence. The posterior probability that this n-gram was generated from language model l is given by

$$P(l|w_{k-n+1}^{(s)},\ldots,w_{k-1}^{(s)},w_k^{(s)},\boldsymbol{\lambda}^{(0)}) = \frac{\lambda_l^{(0)}P_l(w_k^{(s)}|w_{k-1}^{(s)},\ldots,w_{k-n+1}^{(s)})}{\sum_{l=1}^{L}\lambda_l^{(0)}P_l(w_k^{(s)}|w_{k-1}^{(s)},\ldots,w_{k-n+1}^{(s)})} \tag{9}$$

The average posterior probability of generating n-grams from language model l is

$$\lambda_l^{(1)} = \frac{1}{|\mathcal{W}|}\sum_{s=1}^{S}\sum_{k=1}^{|\boldsymbol{w}^{(s)}|}P(l|w_{k-n+1}^{(s)},\ldots,w_{k-1}^{(s)},w_k^{(s)},\boldsymbol{\lambda}^{(0)}) \tag{10}$$

The average posterior reflects how well language model l matches the data set $\mathcal{W}$. Using $\boldsymbol{\lambda}^{(1)}$ in place of $\boldsymbol{\lambda}^{(0)}$ is guaranteed[1] not to increase perplexity of the interpolated language model on the date set $\mathcal{W}$. This can be expressed as

$$\text{PPL}(\mathcal{W};\boldsymbol{\lambda}^{(i+1)}) \leq \text{PPL}(\mathcal{W};\boldsymbol{\lambda}^{(i)}) \quad \text{for all } i \tag{11}$$

Once perplexity is no longer decreasing the process can be terminated.

2 Objective

The above iterative process can be simply implemented if n-gram probabilities from all L language models were known for each n-gram in the data set $\mathcal{W}$. Given L development sequences of n-gram probabilities, write a code to estimate optimal weights $\boldsymbol{\lambda}^{*}$. Evaluate optimal weights on L evaluation sequences of n-gram probabilities. Discuss *(i)* how would you set $\boldsymbol{\lambda}^{(0)}$, *(ii)* impact of alternative choices of $\boldsymbol{\lambda}^{(0)}$, *(iii)* termination criterion, *(iv)* impact of low-probability n-grams, *(v)* alternative estimation schemes. Report perplexities at the beginning and end of training. Plot perplexity progression during estimation on both development and evaluation data. Report on generalisation and provide some interpretation of learnt weights.

3 Marking scheme

Two critical elements are assessed: implementation (50 marks) and description (50 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. **Implementation** Write a code that can take L provided sequences of n-gram probabilities (and any other file you deem relevant), estimate a set of interpolation weights and write the estimated weights into an output file. For example, if Python language is used then the execution of your code may look like

   ```
   python estimate.py dev/*.probs weights
   ```

   where `dev/*.probs` is a list of L provided sequences of n-gram probabilities, `weights` is an output file for storing the estimated weights.

   Write another piece of code that can take a set of weights and L provided sequences of n-gram probabilities (and any other file you deem relevant) and compute perplexity of the resulting interpolated n-gram language model. For example, if Python language is used then the execution of your code may look like

   ```
   python apply.py weights eval/*.probs
   ```

   where `weights` is a set of weights, `eval/*.probs` is a list of L provided sequences of n-gram probabilities.

   1.1 **Data structure (5 marks)** for holding L provided sequences of n-gram probabilities. Implement a data structure enabling efficient access to n-gram probabilities.

[1] This is one of the simplest examples of Expectation-Maximisation.

1.2 **Interpolated n-gram probability (5 marks).** Implement an efficient approach for computing the interpolated n-gram probability in equation (4).

1.3 **Perplexity (5 marks).** Implement an efficient approach for computing perplexity in equation (8) given a sequence of (interpolated) n-gram probabilities.

1.4 **Posteriors (5 marks).** Implement an efficient approach for computing posterior probabilities in equation (9).

1.5 **Average posteriors (5 marks).** Implement an efficient approach for computing average per word posterior probabilities for each interpolated n-gram language model in equation (10).

1.6 **Iterative weight update (25 marks).** Implement an efficient approach for incrementally updating weights, recomputing perplexity and verifying the convergence guarantee in equation (11).

2. **Description**

2.1 **Benchmark results (5 marks).**
Report your best perplexity values on the development and evaluation data. *Important* Whenever perplexity is reported you must specify the number of words used to compute it ($|\mathcal{W}|$ in equation (8)). Provide interpretation of the learnt weights given the names of language models provided in Resource section.

2.2 **Perplexity curves (15 marks).**
- Plot how perplexity of the interpolated n-gram language model changes on each iteration for development data.
- Plot the same for evaluation data.
- Discuss whether the learnt weights generalise to evaluation data or not.

2.3 **Initial weights (5 marks).**
- Report your initial set of interpolation weights.
- Discuss the impact of using alternative initial weights.

2.4 **Termination criterion (5 marks).**
Report the choice of termination criterion used to stop iterative weight estimation.

2.5 **Low-probability n-gram probabilities (10 marks).**
- Report whether low-probability n-gram probabilities are used or not used for weight estimation and why.
- In either case report the impact on generalisation (perplexity on evaluation data) if low-probability n-gram probabilities are excluded from weight estimation.
- Describe the scheme used to eliminate low-probability n-gram probabilities.

2.6 **Alternative weight estimation scheme (10 marks).**
Discuss alternative options to estimate weights. Options may include (a) introducing more weights, (b) changing the objective function, (c) changing optimisation.

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

Two sets of resources are provided for this task:

- development n-gram probabilities extracted from 10 language models. The language models were built using AMI, Fisher, Google, Hub4, ICSI, MGB, News, SDR, TDT and TED data respectively. n-gram probabilities for each source are located in `dev/<stream>.probs` file where `<stream>` corresponds to one of the data sets.

- evaluation n-gram probabilities extracted from the same set of language models. n-gram probabilities for each source are located in `eval/<stream>.probs` file where `<stream>` corresponds to one of the data sets.

Folders `dev` and `eval` are provided in both `tar.gz` and `zip` compressed formats.

6 Reading List

[1] E. Pusateri *et al*, ”Connecting and comparing language model interpolation techniques”, Proc. Interspeech, 2019. Available online: `https://www.isca-speech.org/archive/Interspeech_2019/pdfs/1822.pdf`

[2] H. Sak *et al*, ”Mixture of mixture n-gram language models”, Proc. ASRU, 2013. Available online: `https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43902.pdf`