1 Language Modelling

One of the most basic tasks for handling textual context is language modelling, which is an unsupervised machine learning problem. In this assignment, you will explore the usage of language modelling for predicting which emoji a tweet will likely end with. This could later form the basis of an opinion mining model.

1.1 Academic Code of Conduct

You are welcome to discuss the project with other students. However, any sharing of code and/or text is not permitted, and each person must submit their own project. Plagiarism tools will be used in your submissions. **If you have questions regarding the project, post them on the discussion forum**.

1.2 Preliminaries

You may use any programming language, and basic libraries such as NumPy to solve this project. Make sure that we can run your code with these dependencies when you submit. However, you are **not allowed to use pre-implemented functions** to build the language model - i.e. you are required to implement n-gram construction, smoothing and language model prediction yourself. **If in doubt, please ask in the discussion forum**.

2 Project Task

You are given the multilingual emoji prediction dataset from 2018. Your task is to implement and evaluate a language model that, given a sentence, predicts the most likely emoji to complete the sentence. This section provides details on how to do this.

To help you in solving this project, I *strongly recommend* you keep "Speech and Language Processing" by Jurafsky and Martin, particularly Chapter 4 on language modelling with n-grams at hand. You can find it here: https://web.stanford.edu/~jurafsky/slp3/

2.1 Data

We will use the *English part* of the data from are centre search competition, SemEval 2018Task2on Multilingual emoji prediction: https://competitions.codalab.org/competitions/17344.

You can download the dataset with already crawled English tweets, to be used in this assignment, here: https://www.dropbox.com/s/cjoi8y8sgdur77t/Semeval2018-Task2-Emoji-Detection.zip?dl=0 . Please read the contained README files carefully. The dataset is prepared in a way such that it is split into a "training" and "trial" part. Tweets and the corresponding emojis the tweets end with are stored separately.

2.2 Data Preprocessing

Construct the vocabulary (i.e. the set of all unique words and emojis observed) over the training part of the dataset. This should include an "unknown" token for words unseen at test time.

2.2 Language Model Construction

Implement several variants of a language model (simple n-gram model, n-gram model with add-k smoothing, n-gram model with interpolation), which you will later evaluate and contrast, as follows:

2.2.1 Construct a n-gram based language model over the training set. Choose "n" freely (though note that you will have to motivate and criticize this choice as part of the questions later). The n-gram language model should predict the probability of a sentence occurring in the training dataset.

2.2.2 As outlined in Chapter 4 Section 4.4.2 and 4.4.3 of the "Speech and Language Processing" book, implement add-k smoothing and interpolation.

2.4 Evaluation

You will evaluate your resulting language model on the "trial" part of the dataset in two ways.

2.4.1 Internal evaluation: you will evaluate the language model using perplexity. The perplexity of a language model P for a test set $W = w_1 w_2 ... w_N$ is defined as follows:

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{p(w_i | w_1 ... w_{i-1})}}$$

2.4.2 External evaluation: you will evaluate how well the model can predict emojis which should follow tweets. For this, calculate the probability of the language model observing each tweet in the test set ending in each of the emojis. The final predicted emoji for the test set should be the one with the highest probability over the emoji choices. Evaluate the performance of your resulting unsupervised emoji prediction model using the macro F1 over emoji labels.

For each of the emojis, calculate the F1 as follows:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \; ; \; recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

Note down the F1 achieved for every emoji separately, then also compute the macro F1 of the individual F1 scores by averaging them over the number of emojis.

Question1: What size did you use for “n” in the n-gram language model, what values did you use for “k” for add-k smoothing and what values of “λ” for interpolation? Why is that? Explain the advantages and disadvantages of your choices.

Question2: Computethescoresfortheintrinsicandextrinsicevaluationforeachofthelanguage model variants. What do the scores you achieved for the intrinsic and extrinsic evaluation mean? Describe in your own words.

Question3: Instead of using the language model to predict the most likely emoji to complete the sentence, use each of the language models to predict the most likely word following each of the trial tweets, not adding the emojis at the end. What do you observe? Discuss.

Question4: What are the advantages and disadvantages of using a n-gram based language model, as you have implemented here, over a neural language model, as also introduced in the language modelling lecture?

Question5: One of the main problems with language processing models, including language models, is that they are limited to the words they have seen, which were part of the original vocabulary constructed. One scenario in which this issue arises is misspelled words. Discuss why this is problematic and how a language model could be used to correct spelling mistakes.

Question6: Augment your language model resulting from 2.2.2 to handle unseen word sat test time. The new language model should be able to: 1) assign a more meaningful probability to these words than that corresponding to the “unknown” back-off word; 2) continue a sentence in which the previous word is an unseen word in a more meaningful way than your language model constructed resulting from 2.2.2 would. Assess this by measuring and contrasting perplexity.

Question7: Test the usefulness of your new language model resulting from Question6 for the task of correcting spelling mistakes. 1) For each of the tweets in the development set, create a corresponding version with at least 3 types of automatically introduced spelling mistakes. You can introduce those by permuting two neighbouring characters in each word of the tweet, e.g. “characters” -> “charatcers”. The position of the first character to be permuted should be randomly chosen, though for simplicity reasons you can restrict this to the second half of each word; 2) Use your new language model to correct those mistakes, i.e. to reconstruct the original tweets from the tweets with automatically introduced spelling mistakes; 3) Measure the performance of your system by computing the F1 on a word-level.