Building a Language Model – Part I

The first project will be related to building a language model from scratch. In the first instance, the first thing that is required is that you prepare a corpus (choose only one) and carry out the necessary pre-processing.

Maltese Corpus: http://mlrs.research.um.edu.mt/index.php?page=downloads
(Baby) British National Corpus: http://ota.ox.ac.uk/desc/2553

First set of tasks:

1. Extract the selected corpus
2. Build frequency counts for n-grams

Things to keep in mind:

- Try to make your code modular. Of course, initially hard coding stuff is ok. But ideally, evolve your hard-coding to generic, modular code
- You might also want to hard code a specific input string so that you can ensure that your frequency counts are correct.
- Consider computational issues – check how long it takes to read the corpus, build frequency counts, how well it scales up, amount of RAM that the system ends up using, etc.
- When reading the files, you might want to consider using one file initially. But already start investigating using all files in the corpus. Remember the larger your corpus, the better the models will be.
- Think ahead – Once you get the frequency counts sorted, remember that like in any machine learning set up, we will be needing a training set and a test set. So you will eventually need to split your corpus into two distinct parts. More about this at a later stage, but it is good that you start thinking about this eventuality.

Building a Language Model – Part II

Now that you have selected the corpus, extracted it and built your frequencies, you can now start working towards building a language model from scratch.

We will be building a language model that will include the Unigram, Bigram and Trigram models and we will be building three different langauge models

1. The **`Vanilla Language model`** is your regular language model
2. The **`Laplace Language model`** will take the Vanilla as its basis, but you will now include Laplace smoothing.
3. The **`UNK Language model`** will take the Vanilla as its basis, but now you will set all the words that have a count of 2 or less as <UNK> tokens. And then recalculate accordingly. And recalculate Laplace smoothing on this model.

Next, create a function that will carry out **`Linear Interpolation`** – this function takes one of the above 3 flavours of language models and calculate the probability of a sentence using the following lambdas: trigram = 0.6; bigram = 0.3; unigram = 0.1.

Evaluation:

1. Take the test corpus, iterate through the sentences and calculate the probabilites for each sentence with every model. You will then use this to calculate the **`Perplexity`** and produce the following table:

	Unigram	Bigram	Trigram	Linear Interpolation
Vanilla				
Laplace				
UNK				

2. Create a function that allows the user to select one of the Language Models (Vanilla, Laplace, UNK) and input a phrase. The function will use this phrase to **`Generate`** the rest of the sentence until it encounters the end of sentence token (i.e. the LM says stop!)

Building a Language Model – Practicalities

Submission Requirements:

1. Code & Documentation – be concise – I don't want an explanation of the class notes. I want an explanation of your implementation choices. I also accept a submission of a jupiter notebook with documentation integrated in it.
2. Demo (arranged after submission)

Marking Scheme:
Preprocessing: 5 marks
Vanilla LM: 10 marks
Laplace LM: 10 marks
UNK LM: 10 marks
Interpolation: 10 marks
Perplexity Evaluation: 10 marks
Generation Evaluation: 10 marks
Documentation (justification of choices, etc.): 20 marks
Demo: 15 marks

Submission Date:
12th May midnight
Demos will probably be held on the 21st May between 10am – noon.

Important:

1. As part of your **documentation**, I would like you to include the following:
    a. How large is the corpus that you are working with?
    b. How much time does it take to build the language models?
    c. How much space did your datastructures require once all models were learnt?
    d. Specify any additional features that you'd like to point out.
2. For your **demo**, you can use models that have been saved earlier and loaded on the day.
3. Use the **VLE Q&A** forum to ask questions. Discussions are welcome, sharing of code isn't!
4. This is an individual task, **plagiarism** will not be tolerated. Reference all sources used appropriately. You should include a signed plagiarism form as part of your submission.
5. **Delayed** submissions will result in a 10% deduction per day.