UNIVERSITY OF CAMBRIDGE

Devan Kuleindiren
Robinson College
dk503

DRAFT

Computer Science Tripos - Part II Project Proposal

Language Modelling for Text Prediction

October 11, 2016

Project Originators: Devan Kuleindiren & Dr. Marek Rei

Project Supervisor: Dr. Marek Rei
Signature:

Project Co-Supervisor: Dr. Ekaterina Shutova
Signature:

Director of Studies: Dr. Alastair Beresford
Signature:

Project Overseers: Dr. Markus Kuhn & Prof. Peter Sewell
Signatures:

Introduction

A language model is a probability distribution over a sequence of words, which can be used to estimate the relative likelihood of words or phrases occurring in various contexts. The predictive power of such models is useful in speech recognition, text prediction, machine translation, handwriting recognition, part-of-speech tagging and information retrieval. For instance, in a text prediction context, if a user has typed:

`‘Do you want to grab a ’`

then a language model could be used to suggest probable next words such as `‘coffee’`, `‘drink’` or `‘bite’`, and these predictions could further be narrowed down when the user starts typing the next word.

With the recent shift from desktop to mobile computing, it has become increasingly important to facilitate accurate, high-speed language models that aren't too energy or memory hungry. The aim of this project is to investigate this area by implementing and benchmarking existing language models, with an aim of providing useful insights into how their the accuracy, speed and resource consumption compare.

Starting point

Code

There is no prior codebase, and so the project will be written from scratch. TensorFlow, an open-source machine learning library, will assist in the construction of the neural network-based language models.

Knowledge

Language modelling draws ideas from natural language processing and machine learning. I have no prior experience in the former field, but in the latter I have an understanding of neural networks (including RNNs) that comes from both the Artificial Intelligence I course and personal reading.

The following Computer Science Tripos courses will be useful when undertaking my project:

- **Natural Language Processing** - n-gram language models.
- **Artificial Intelligence I** - neural networks.
- **Machine Learning and Bayesian Inference** - tips and tricks around supervised learning.
- **Mathematical Methods for Computer Science** - Markov chains.
- **Information Theory** - entropy.

Algorithms, **Object-Oriented Programming**, **Software and Interface Design**, and **Software Engineering** will also prove useful for general programming and software engineering.

There are aspects of language modelling not covered by the Tripos, including various smoothing techniques for n-gram LMs and recurrent neural networks. I will fill this gap with personal reading.

Resources required

Machines

The development of my project will be carried out on my personal laptop, a 2015 MacBook Pro running macOS Sierra. I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure.

In order to train language models on large datasets, I will make use of Cambridge's HPCS ***TODO:*** confirm this with Computer Lab and check HPC supports TensorFlow.

Datasets

In order to train and evaluate the various language models, I will need to make use of various datasets, including, but not limited to:

- The Penn Treebank (PTB) dataset. [1]
- The One Billion Word dataset. [2]
- The CLC FCE dataset. [3]

Version Control & Backup

I will use Git for version control, with one Git repository for all project documentation and another for all project source code. Both repositories will be hosted on GitHub, and all local commits will be pushed no more than a few hours after they are created, meaning that I should never lose more than half a day's worth of work.

I will also make weekly backups on an external hard drive, so that in the unlikely event that both GitHub goes down and my laptop fails, I will be able to recover my project on a new machine. ***TODO:*** Check whether MCS supports TensorFlow.

Work to be done

Language models

The first stage of the project is to implement a variety of language modelling algorithms. Starting with the simplest, I will implement n-gram models with varying values of n. I

will also investigate the effect of add-1 smoothing [4], Katz smoothing [5], absolute discounting [6], Kneser-Ney smoothing [7] and modified Kneser-Ney smoothing [8].

Once the n-gram models along with their corresponding smoothing techniques have been implemented, I will implement some neural network based language models, which have recently proven very effective. I will start with the first incarnation of neuro-probablistic language models from Bengio et al. [9] which utilises a feed-forward neural network. Next, I will progress to an RNN-based language model [10], and finally, I will advance to the slightly more complex LSTM (long short-term memory) architecture [11].

In order to implement the neural network-based language models, I will make use of TensorFlow, Google's open-source machine learning library. TensorFlow has APIs for both Python and C++, so I will implement my language models in either one of those languages.

Benchmarking framework

Once a series of language models have been implemented, the next stage would be to implement a framework into which each language model can be inserted. The framework should run a series of tests and generate evaluation for:

- **Accuracy:** Perplexity, cross-entropy, direct comparison of models.
- **Speed:** Training times, inference times.
- **Resource usage:** Memory (and energy?) usage during both training and inference.

Demonstration application

In order to demonstrate the language models in a practical context, I aim to integrate them into a simple console application that predicts the next word in the user's sentence.

Success criteria

This project will be deemed successful if the following criteria are reached:

- Language models (LMs) using the following techniques are implemented:
  - n-gram LMs with add-one, Katz, absolute discounting, Kneser-Ney, modified Kneser-Ney and no smoothing.
  - Bengio's NN-based LM.
  - A vanilla RNN-based LM.
  - LSTM (long short-term memory)-based LM.
- Comprehensible and reliable comparisons between the various LM implementations are made across the following factors:
  - The accuracy of the model's text prediction.
  - The resources (time and space) required to train the model.

  - The resources (time and space) to run inference on the model after training.

- A simple console application is developed to demonstrate the capability of the aforementioned language models in the context of next-word prediction.

Possible extensions

- Implement more language models.
- Integrate the language models into a mobile keyboard.

Timetable

Preliminary reading for the project begun on 6/10/2016, however, the project more officially begins on 22/10/2016, the day after the deadline of the project proposal.

Michaelmas term

Week 1 - 2	• Refine project idea and start project proposal. • Background reading on n-gram language models and smoothing techniques.
Milestone	*Project proposal finished.*
Week 3	• Background reading on recurrent neural networks and long short-term memory in particular.
Week 4	• Implement a simple bigram language model, without any smoothing.
Milestone	*A working implementation of a basic language model completed.*
Week 5 - 6	• Implement further n-gram language models, with: ◦ Add-1 smoothing ◦ Good-Turing smoothing ◦ Kneser-Ney smoothing ◦ Modified Kneser-Ney smoothing
Milestone	*A working implementation of n-gram language models with various smoothing techniques completed.*
Week 7 - 8	• Implement Bengio's NN-language model. • Start implementation of vanilla RNN language model.
Milestone	*A working implementation of Bengio's NN-based language model.*

Christmas vacation

	• Finish implementation of RNN language model. • Implement LSTM language model. • Implement framework to benchmark a given language model over the evaluation metrics highlighted in the 'Work to be done' section. • Start implementing an application to demonstrate the capability of the language models in the context of next-word prediction.
Milestone	*RNN and LSTM language models, and test framework complete.*

Lent term

Week 1 - 2	• Write progress report. • Complete demonstration application.
Milestone	*Progress report complete.*
Week 3 - 5	• Rigorous testing and bug fixing of project. • If time allows, implement project extensions. • Present project to overseers.
Milestone	*Project presentation given and implementation complete.*
Week 6 - 8	• Start writing dissertation.

Easter vacation

	• Finish writing dissertation. • Proof-read dissertation.
Milestone	*Final dissertation draft submitted to supervisor and Director of Studies.*

TODO: Format references properly.

References

[1] http://www.cis.upenn.edu/ treebank/

[2] http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41880.pdf

[3] http://ilexir.co.uk/applications/clc-fce-dataset/

[4] ***TODO:***

[5] ***TODO:***

[6] ***TODO:***

[7] ***TODO:***

[8] http://www.speech.sri.com/projects/srilm/manpages/pdfs/chen-goodman-tr-10-98.pdf

[9] ***TODO:***

[10] http://www.iro.umontreal.ca/ felipe/IFT6010-Automne2011/resources/tp3/kombrink_interspeech2

[11] https://arxiv.org/abs/1409.2329