UNIVERSITY OF CAMBRIDGE

Devan Kuleindiren
Robinson College
dk503

Computer Science Tripos - Part II Project Proposal

Language Modelling for Text Prediction

October 21, 2016

supervised by
Dr. Marek Rei & Dr. Ekaterina Shutova

Introduction

Language models (LMs) produce 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 memory hungry. The aim of this project is to investigate this area by implementing and benchmarking various language models, with an aim of providing useful insights into how their the accuracy, speed and resource consumption compare.

I will begin with the traditional n-gram language models and their various smoothing techniques, before proceeding onto neural network-based language models. Combinations of such models, which can be achieved by interpolating their probability distributions, will also be included in the benchmark.

In the context of text prediction, one issue with existing language models is that they assume that their input is error (such as spelling or grammatical errors) free. With the presence of typos, human error and people learning new languages, this assumption is often not so reliable. I aim to include some extensions to existing language models that seek to provide better predictions over error-prone text, and I will assess whether they offer any performance improvements by including them in the benchmark.

Starting point

Code

TensorFlow, an open-source machine learning library, will assist in the construction of the neural network-based language models.

Computer Science Tripos

The following 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.

Experience

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.

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.

I intend to train the language models on my laptop. When experimenting with the neural network-based language models on larger datasets, I may utilise the University's High Performance Computing Service[1] in order to run the models on a GPU cluster.

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.[2]
- The One Billion Word dataset.[3]
- The CLC FCE dataset.[4]

[1] https://www.cl.cam.ac.uk/local/sys/resources/hpc/
[2] http://www.cis.upenn.edu/~treebank/
[3] http://www.statmt.org/lm-benchmark/
[4] http://ilexir.co.uk/applications/clc-fce-dataset/

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.

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 [1], Katz smoothing [2], absolute discounting [3], Kneser-Ney smoothing [4] and modified Kneser-Ney smoothing [5].

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 an RNN (recurrent neural network) based language model [6], and then I will advance to the more complex LSTM (long short-term memory) architecture [7].

Combinations of the neural network and n-gram based language models, and extensions to the aforementioned models that aim to improve performance over error-prone text, will also be implemented.

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 both Python and C++ APIs, but the Python API is more complete and easier to use[5], and so I will construct my language models in Python.

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 and guessing entropy.
- **Speed and resource usage:** Time and memory usage at training and inference.

Given that I will be evaluating the language models in the context of text prediction, I will also include some metrics for how useful the predictions are for users, such as the number of letters the user could avoid typing as the result of a correct prediction.

[5] https://www.tensorflow.org/versions/r0.11/api_docs/cc/index.html

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 met:

- Language models (LMs) using the following techniques are implemented:
  - N-gram LMs with various smoothing techniques.
  - A vanilla RNN-based LM.
  - An LSTM-based LM.
- Comprehensible and reliable comparisons between the various LM implementations and their combinations are made regarding their accuracy, speed and resource consumption during both training and inference.
- A simple console application is developed to demonstrate the capability of the aforementioned language models in the context of next-word prediction.

Possible extensions

Mobile Keyboard

One possible extension would be to integrate the language model implementations into a system-wide mobile keyboard on iOS - made possible as of iOS 8.[6] This would involve constructing a simple keyboard interface which displays suggestions for the next word in the user's sentence, by querying the language model(s) and filtering the results as the next word is typed. TensorFlow supports exportation of models into a language-independent format (using Google's protocol buffers), which means I could train and export my language models in Python and then load them into iOS using TensorFlow's C++ API.

[6] https://developer.apple.com/library/content/documentation/General/Conceptual/ExtensibilityPG/CustomKeyboard.html

Timetable

Michaelmas term

06/10 - 19/10	• Refine project idea and write project proposal. • Background reading on n-gram language models and smoothing techniques.
Milestone	*Project proposal finished.*
20/10 - 26/10	• Implement a simple bigram language model, without any smoothing. • Background reading on recurrent neural networks and long short-term memory in particular.
Milestone	*A working implementation of a basic bigram language model completed.*
27/10 - 09/11	• Implement a test framework for generating evaluation metrics for a given language model.
Milestone	*Test framework complete.*
10/11 - 16/11	• Implement the various smoothing techniques outlined in 'Work to be done'.
Milestone	*A working implementation of n-gram language models with various smoothing techniques completed.*
17/11 - 23/11	• Implement vanilla RNN language model.
Milestone	*A working implementation of RNN-based language model.*
24/11 - 30/11	• Implement LSTM-based language model.
Milestone	*A working implementation of LSTM-based language model.*

Christmas vacation

01/12 - 14/12	• Experiment with extending the existing language models to improve performance on error-prone text.
05/01 - 18/01	• Benchmark and evaluate the various language models. • Start implementing an application to demonstrate the capability of the language models in the context of next-word prediction.
Milestone	*Language models benchmarked.*

Lent term

19/01 - 01/02	• Write progress report. • Complete demonstration application.
Milestone	*Progress report and demonstration application complete.*
02/02 - 22/02	• Rigorous testing and bug-fixing of project. • If time allows, implement project extensions. • Present project to overseers.
Milestone	*Project presentation given and implementation complete.*
23/02 - 15/03	• Start writing dissertation.

Easter vacation

16/03 - 19/04	• Finish writing dissertation and proof-read it.
Milestone	*Final dissertation draft submitted to supervisor and Director of Studies.*

Submission deadline on 19/05/2017.

References

[1] W.E. Johnson. Probability: deductive and inductive problems *Mind*, Vol. 41, No. 164, pp. 409-423, 1932.

[2] S.M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 35, No. 3, pp. 400-401, 1987.

[3] H. Ney, U. Essen and R. Kneser. On structuring probabilistic dependences in stochastic language modeling. *Computer, Speech, and Language*, Vol. 8, No. 1, pp. 1-38, 1994.

[4] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 181-184, 1995.

[5] S.F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, Vol. 13, No. 4, pp. 359-394, 1999.

[6] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, Vol. 2, pp. 3, 2010.

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, Vol. 9, No. 8, pp. 1735-1780, 1997.