Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра системного программирования

Отчёт по спецкурсу «Математические методы анализа текста»

Обучение нейронной сети для машинного перевода на основе параллельного корпуса текстов

Работу выполнили
студенты 528 группы:
Машонский И. Д.
Пархоменко П. А.
Шишватов В. А.
Преподаватель:
Dr Мстислав Масленников

Москва 2016

1 Постановка задачи

Целью данной работы является построение системы машинного перевода. Алгоритм работы системы должен быть основан на нейронной сети. Обучение и тестирование системы должно происходить с помощью параллельных корпусов. Система должна работать для следующих пар языков:

- русский - английский;
- русский - немецкий;
- английский - немецкий.

Для выполнения этой цели необходимо найти существующие методы использования нейронной сети в задаче машинного перевода и исследовать реализации этих методов. В качестве параллельных корпусов для обучения модели должны быть использованы параллельные корпуса, составленные одним из слушателей курса в 2015 году [4]. Также необходимо измерить качество работы системы с помощью функций оценки качества, используемых в других работах, посвященных этой задаче.

2 Теоретическое описание решения

В ходе исследования существующих работ по данной задаче было выявлено, что самой популярной является работа [1]. Решение, представленное в данной статье, заключается в использовании двух рекуррентных нейронных сетей (RNNs [7]). Одна из них (encoder) преобразует входную последовательность слов в вектор фиксированного размера. Другая (decoder) восстанавливает последовательность слов по вектору фиксированной длины. Базовая архитектура модели представлена на рисунке 1.

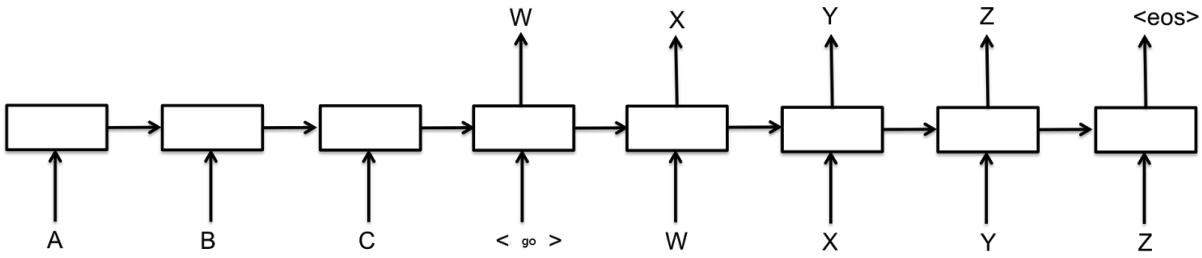

Рис. 1. Базовая архитектура sequence-to-sequence модели

На рисунке прямоугольниками обозначены ячейки RNN, более точно, ячейки LSTM [6]. Encoder и decoder могут использовать как общий набор весов, так и использовать различные наборы параметров. Каждый входной элемент модели должен быть закодирован как вектор состояния фиксированного размера. Входы encoder обозначены символами A, B, C, входы decoder - символами GO, W, X, Y, Z.

Параллельные предложения разных языков состоят из различного числа слов. Так как предложение первого языка длины L1 передается на вход encoder, а предложение второго языка длины L2 передается на вход decoder, в общем случае необходимо создавать sequence-to-sequence модель для каждой пары длин (L1, L2 + 1), что приводит к большому числу похожих подграфов в результирующем графе. Для эффективной обработки предложений разных размеров в модели перевода используется метод bucketing, заключающийся в том, что пары параллельных предложений различной длины выравниваются по предопределенным значениям длины. В работе использовались следующие наборы длин: (5, 10), (10, 15), (20, 25), (40, 50), где первое число в паре - длина выровненного предложения первого языка и второе - длина выровненного предложения второго языка.

3 Практический этап

В ходе исследования существующих программных средств были выявлены 2 системы, в которых был реализован метод, описанный в работе [1]: Zoph_RNN [2] и Tensorflow [3].

3.1 Zoph_RNN

Zoph_RNN - система, решающая задачу машинного перевода.
Характеристики системы:

- работает параллельно на нескольких GPU;
- написана на языке C++.

Были проведены попытки использовать данную систему для решения поставленной задачи. При этом были выявлены проблемы совместимости с операционной системой Windows: в OS Windows для работы с CUDA требуется компилятор Visual Studio версии, не выше 14, однако проект Zoph_RNN написан на C++11, имеющем проблемы с поддержкой компилятором версии Visual Studio 14. Для решения данной проблемы требуется переписывать существующий проект под Windows с учетом описанных ограничений.

3.2 TensorFlow

TensorFlow - система с открытым кодом, разработанная компанией Google, позволяющая использовать различные алгоритмы нейронных сетей для анализа данных. Система написана на языке C++, API предоставлено для языков C++ (версия с некоторыми ограничениями) и Python. В силу того, что TensorFlow Python API имеет более развитую поддержку и в нем отсутствуют ограничения, на сайте разработчиков рекомендуется использовать версию для Python.

В ходе работы над практической частью для реализации итогового решения была использована система TensorFlow.

3.3 Данные

В качестве параллельных корпусов использовались данные, полученные одним из слушателей курса [4].

Параллельные корпуса были составлены для следующих пар языков:

- русский - английский;
- русский - немецкий;
- английский - немецкий.

Данные устроены следующим образом: они состоят из последовательности троек, каждая из которых состоит из следующих элементов:

- предложение первого языка;
- предложение второго языка;
- численная характеристика, определяющая близость предложений.

Для набора данных были написаны вспомогательные скрипты на языке Python. Первый скрипт переводит данные в формат, требуемый Tensorflow. Этот скрипт включает в результирующий датасет только те предложения, для которых численная характеристика, определяющая близость предложений, больше или равна 0.1. Предобработка данных происходила на двух уровнях:

- уровень предложения;
- уровень слова.

Уровень предложения включает в себя следующие действия:

- замена символа перевода строки (‘\n’) на символ пробела(‘ ‘). Это необходимо для избежания смещения предложений в параллельных корпусах, когда предложение одного языка содержит символ перевода строки, а второго - нет.
- разбиение предложения на токены с помощью стандартной функции split() языка Python.

Уровень слова включает в себя следующие действия:

- применение к каждому слову регулярного выражения `r'^\W*(\w[\w\W]*\w)\W*$|^\W*(\w)\W*$|^(\W*)$'`. Суть регулярного выражения - удаление символов пунктуации (не букв и не цифр) из начала и конца слова. Регулярное выражение состоит из 3 альтернатив:
  - первая альтернатива удаляет символы из начала и конца слова и длина слова больше 1;
  - вторая альтернатива удаляет символы из начала и конца слова и длина слова равна 1;
  - третья альтернатива удаляет символы из начала и конца слова и длина слова равна 0;
- перевод каждого символа слова в нижний регистр.

Второй скрипт разбивает данные на тренировочную и тестовую выборки в соотношении 70/30, предварительно перемешав предложения датасета (сохраняя при этом соответствие между параллельными предложениями).

Визуальный просмотр имеющихся наборов данных показал их относительно низкое качество. Так, например, в параллельном корпусе для русского и английского языков преобладают предложения вида “сказал он”, “спросил я”, “сказала я” и другие подобные предложения, имеющие низкий уровень смысловой нагрузки, причем каждое из таких предложений встречается два и более раз.

Кроме того, некоторые предложения с высокими значениями численной характеристики, определяющей близость предложений, имеют некорректный перевод. Например, предложение русского языка *“- Свинья”* имеет перевод на английский язык *“"Swine," pursued Mr. Wopsle, in his deepest voice, and pointing his fork at my blushes, as if he were mentioning my Christian name, "swine were the companions of the prodigal.”*, а предложение *“И… и главное, он такой грубый, грязный, обращение у него трактирное; и… и, положим, он знает, что и он, ну хоть немного, да порядочный же человек… ну, так чем же тут гордиться, что порядочный человек”* переводится как *“"And.”*

3.4. Эксперименты

Структура нейронной сети имеет следующий вид:

- 3 слоя;
- 1024 нейрона на каждом слое.

В качестве метода автоматической оценки качества машинного перевода была использована метрика BLEU - простая в вычислении, не зависящая от языков, к которым она применяется, и коррелирующая в высокой степени с оценкой реальных людей [5].

Значение BLEU - число из отрезка [0.0; 1.0], которое показывает, насколько близок перевод модели к истинному переводу. Чем ближе значение к 1.0, тем лучшим считается перевод, однако в силу того, что для одного предложения возможно несколько различных вариантов перевода, как правило, значения, близкие к 1.0 достижимы только для переводов реальными людьми.

В таблице 1 приведены данные о параллельных корпусах, использованных при обучении и тестировании моделей машинного перевода.

Табл. 1. Данные использованных параллельных корпусов

Параллельный корпус	Количество предложений (обучение)	Количество предложений (тестирование)	Размер словаря первого языка	Размер словаря второго языка
RU - EN	20736	48382	66857	24919
EN - DE	9387	4023	7359	13238
RU - DE	4664	10882	22099	14295

В таблице 2 приведены результаты проведенных экспериментов для различных наборов данных.

Табл. 2. Результаты экспериментов

Пара языков (с которого переводят - на какой переводят)	Перплексия (при окончании обучения)	BLEU (тестирование)	Время обучения (в днях)
RU→EN	1.03	0.334	~5
EN→RU	1.06	0.373	~4
EN→DE	1.1	0.340	~3.5
DE→EN	1.08	0.378	~3
RU→DE	1.12	0.403	~3

DE→RU	1.09	0.378	~3

Согласно работе [1], state of the art метод машинного перевода показывает результат BLEU 0.37. Полученные в результате проведенных экспериментов данные о качестве построенного решения могут говорить о приемлемом качестве машинного перевода. При этом данные результаты не могут быть однозначно сопоставлены с результатами, представленными в работе [1] в силу того, что эксперименты проводились для разных языков на разных наборах данных.

4. Попытка запустить модель с помощью C++

Была произведена попытка загрузить обученную модель с помощью кода, написанного на языке C++. Эта попытка не увенчалась успехом. Ниже описаны шаги, которые были предприняты для решения данной проблемы.

В качестве C++ кода, отвечающего за загрузку обученной модели, был взят код из следующих источников:

- https://medium.com/jim-fleming/loading-a-tensorflow-graph-with-the-c-api-4caaff88463f#.pvybb7ivm
- https://www.tensorflow.org/versions/r0.8/api_docs/cc/index.html

При попытке скомпилировать исходный код, выдавалась ошибка об отсутствии некоторых файлов (в частности, файла graph.pb.h). Данный файл подключался в файле tensorflow/core/public/session.h, который, в свою очередь, непосредственно подключался в C++ коде.

При поиске решения данной проблемы было выявлено, что необходимо собрать фреймворк Tensorflow заново, из исходных кодов. Сборка из исходников осуществлялась с помощью утилиты Bazel (http://bazel.io/) в соответствии с issue, заведенном в git-hub репозитории Tensorflow (https://github.com/tensorflow/tensorflow/issues/1890).

После сборки с исходников были сгенерированы некоторые недостающие файлы. При попытке запустить скрипт появились проблемы с библиотекой Eigen (http://eigen.tuxfamily.org/index.php?title=Main_Page). Eigen, предоставляемый Bazel, не включал в себя некоторые файлы, которые были необходимы для запуска Tensorflow. Была произведена попытка загрузить последнюю стабильную версию Eigen с официального сайта. Эта попытка не оказалась удачной, так как в последней стабильной версии Eigen отсутствовали некоторые необходимые файлы. Тогда была предпринята попытка скачать неофициальную версию (http://fossies.org/linux/privat/eigen-3.2.8.tar.gz/). В ней были найдены некоторые необходимые файлы, но все требуемые файлы так и не были получены, в том числе файл FixedSizeVector.h. Поиск этого файла и ошибки, которую выдавал Tensorflow при запуске, в Интернете не дали положительных результатов.

Выводы

В ходе работы были исследованы существующие методы машинного перевода, основанные на нейронных сетях, и определен наиболее релевантный. Были найдены программные средства, реализующие этот метод. С использованием одного из программных средств были обучены модели для трех пар языков: русский-английский, английский-немецкий, немецкий-русский. Для подготовки данных для обучения и тестирования моделей были разработаны вспомогательные программные средства, преобразующие исходные наборы данных [4] в формат, совместимый с TensorFlow. Было проведено экспериментальное тестирование выбранного метода на основе реализованной для этой цели метрики BLEU, позволяющей оценить качество работы машинного перевода.

Для улучшения качества решения можно предложить следующие направления.

1. Использование другого набора данных. В ходе работы с предложенным набором данных было выявлено большое количество некорректной разметки, что могло отрицательно сказаться на качестве метода.
2. Использовать большее количество данных для обучения. Количество уникальных слов для каждого языка было небольшим (в среднем, около 5000 - 10000), что не позволяло переводить слова, которые не встречались в тренировочной выборке.
3. Использовать GPU. К сожалению, имеющееся в наличии оборудование не позволяло использовать GPU для обучения модели, что сказалось на скорости обучения и итоговом качестве решения.. Использование GPU может в значительной степени ускорить процесс обучения модели.
4. Одним из возможных направлений в дальнейшей работе является подбор параметров сети и модификация существующего метода. Так, например, для решения задачи можно применить GRU сеть [8], вместо LSTM.

Используемые источники

[1] Sutskever I., Vinyals O., Le Q. V. Sequence to sequence learning with neural networks //Advances in neural information processing systems. – 2014. – С. 3104-3112.

[2] https://github.com/isi-nlp/Zoph_RNN

[3] https://www.tensorflow.org/versions/master/tutorials/seq2seq/index.html

[4] https://github.com/mathtexts/AnAligner15

[5] Papineni K. et al. BLEU: a method for automatic evaluation of machine translation //Proceedings of the 40th annual meeting on association for computational linguistics. – Association for Computational Linguistics, 2002. – С. 311-318.

[6] Hochreiter S., Schmidhuber J. Long short-term memory //Neural computation. – 1997. – Т. 9. – №. 8. – С. 1735-1780.

[7] Chambers D., Mandic J. Recurrent neural networks for prediction: learning algorithms architecture and stability //John Wiley & Sons, Ltd., Chichester. – 2001. – Т. 18. – С. 32.

[8] Chung J. et al. Empirical evaluation of gated recurrent neural networks on sequence modeling //arXiv preprint arXiv:1412.3555. – 2014.