# Арасланова Анна

# Практикум.

# LSTM NN на Python. Знакомство с библиотеками Gensim и TensorFlow

В данной работе представлены 2 наиболее известных метода работы с текстом: Word2Vec и LSTM-сеть. В тексте не даны теоретические справки, но приведены ссылки на подробные материалы, которые позволят самостоятельно понять, что происходит [т.к. автору работы показалось, что лучше сразу прочесть качественный и подробный материал, чем читать переписанный им свод известных вещей].

К каждому теоретическому разделу приписано количество часов, которое автор практикума потратил на изучение.

# Оглавление (ссылки работают)

[Установка TensorFlow](#_Начнём_с_установки)

Теоретическая часть

[Word2Vec](#_Word2Vec_[1_час])

[LSTM](#_LSTM_NN_(Long)

[TensorFlow](#_TensorFlow_[1.5_-)

Практическая часть

[Задание](#_Задача)

[Решение](#_Решение)

Результаты

[Часть 1 (Word2Vec)](#_Часть_1)

[Часть 2 (LSTM)](#_Часть_2)

Почему TensorFlow?

- Потому что это быстро развивающаяся библиотека с подробной документацией, официальными туториалами и большим множеством поклонников:

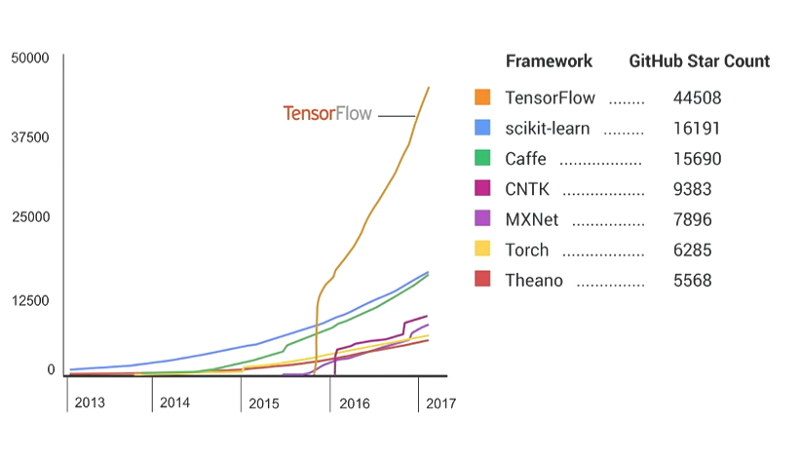


Figure 1 аналитика с github для затравки

# Начнём с установки TensorFlow, т.к. это нетривиальный момент

Инструкция есть на официальном сайте: <https://www.tensorflow.org/install/install_windows>

Установка библиотеки TensorFlow требует немного больше усилий, чем обычная установка библиотек через pip

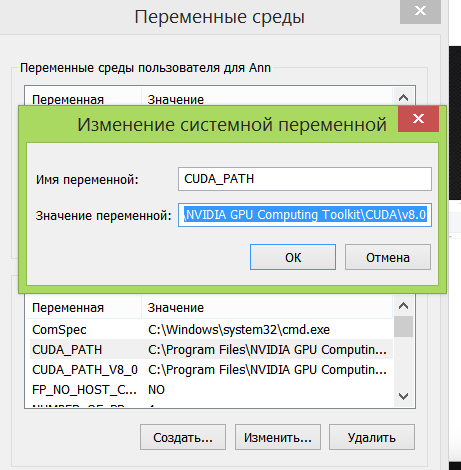
Перечислим основные моменты. Лучше следить с официальным описанием, т.к. что-нибудь может измениться.

Подготовительные работы: (предполагается, что Python уже установлен. В моём случае Anaconda)

\* Для начала надо установить CUDA Toolkit. Стоит обратить внимание, что это может занять больше часа

<https://developer.nvidia.com/cuda-downloads>

\* так же надо создать системную переменную среды с путём к CUDA (она может не создаться автоматически)



\* Надо обновить NVIDIA-драйвера

<http://www.nvidia.com/Download/index.aspx>

\* Затем устанавливаем cuDNN

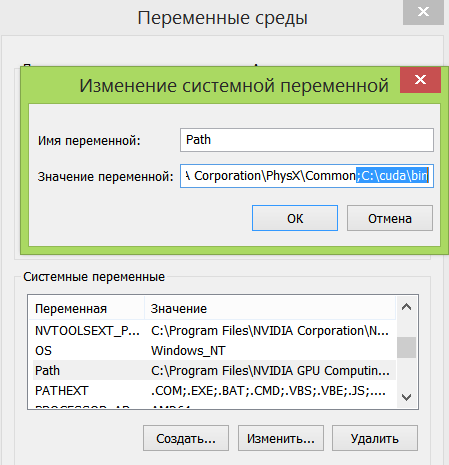
<https://developer.nvidia.com/cudnn>

Чтобы скачать, вас попросят пройти процедуру авторизации. Зарегистрируйтесь, если аккаунта ещё нет

Для TensorFlow нужно скачать версию 5.1 (не 6)

Папку “cuda” (она в нём одна) из скаченного архива надо распаковать. Например, на диск С.

\* Добавляем путь к нашей папке в переменную среды path



Всё, подготовительные этапы закончены

\* Я устанавливаю через Anaconda, поэтому надо сделать один лишний шаг в консоли конды:

conda create -n tensorflow

эта команда создала директорию в папке envs у анаконды

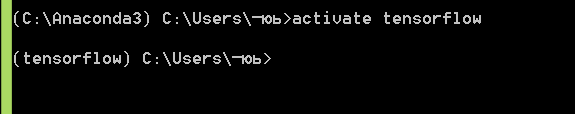
если установлен Python версии выше, чем 3.5, то вместо предыдущей команды пишем

conda create -n tensorflow python=3.5 anaconda

активируем окружение (на будущее: для деактивации пишем deactivate tensorflow)

activate tensorflow

Заметим, что имя окружения изменилось:



\* Устанавливаем. вводим одной строкой строку с официального сайта:

pip install --ignore-installed –upgrade https://storage.googleapis.com/tensorflow/windows/gpu/tensorflow\_gpu-1.1.0-cp35-cp35m-win\_amd64.whl

\* Проверяем (в окружении tensorflow)



Всё. Теперь можно начинать работать

# Word2Vec [1 час]

Отличную теорию на русском языке можно прочитать в этой статье: <http://nlpx.net/archives/179> (алгоритм в разделе «Как всё это работает?»).

Сегодня будем использовать алгоритм skip-grams

# LSTM NN (Long Short Term Memory) [2.5 часа]

Очень подробное описание LSTM сети можно прочитать, например, здесь

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Раньше был хороший перевод этой статьи, может быть восстановят:

<http://alexsosn.github.io/ml/2015/11/16/LSTM.html> [2.5 часа]

Если вдруг возникнут какие-то проблемы с переводом терминов, стоит заглянуть в эту статью <http://www.machinelearning.ru/wiki/images/7/78/2017_417_DrapakSN.pdf>

Не будем приводить теоретическую справку, т.к. она уже много где написана, но теорию (по ссылке выше или где-то ещё) надо прочесть обязательно

# TensorFlow [1.5 - 4 часа]

Введение в библиотеку можно прочесть, например, в этой лекции

<https://cs224d.stanford.edu/lectures/CS224d-Lecture7.pdf> [1.5 часа]

Дополнительно: про нейронные сети, которых мы сегодня коснёмся также рассказано в лекции 10 (замените цифру 7 в ссылке) [2.5 часа]. Но и основной теории по LSTM хватит

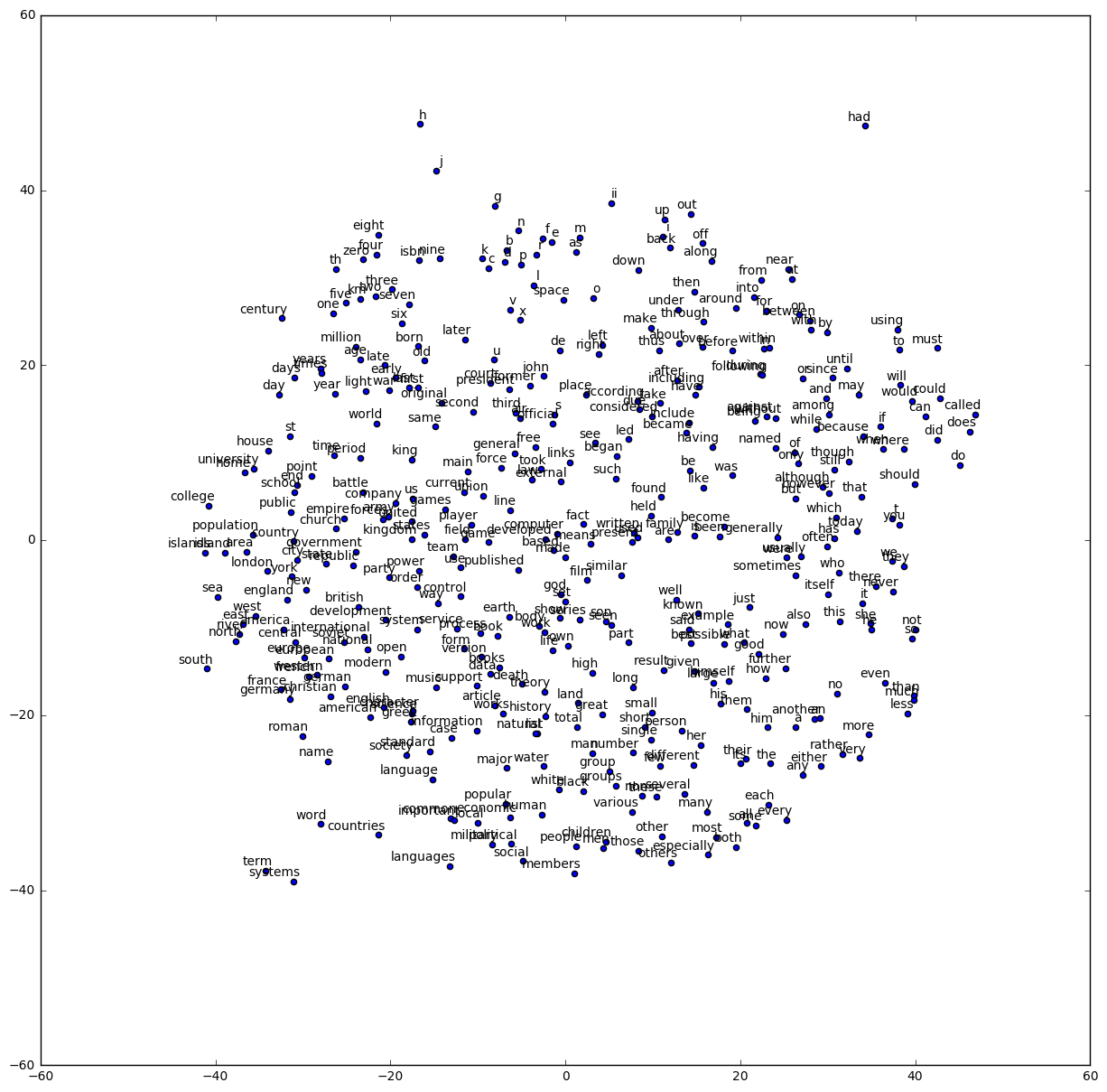
# Задание

Часть 1

Для знакомства в сем, на что способны алгоритмы машинного обучения и в т.ч. алгоритмы нейронных сетей, поработаем с word2vec. Для работы возьмём текст романа «Война и мир», который можно скачать по этой ссылке: <http://vojnaimir.ru/download.html>

Для работы с word2vec будем пользоваться библиотекой genism, потому что она легка в установке и использовании (всё скрыто от пользователя, в т.ч. архитектура). Наши цели в части 1 – демонстрация возможностей, поэтому будем пользоваться этой библиотекой

Word2vec можно собрать и в tensorflow, более того, возможности расширятся: если добавить слой embedding, сокращающий размерность, можно построить граф близости слов, который выглядит примерно так:



В части 2 описан более мощный алгоритм – нейронная сеть LSTM

Текст, с которым будем работать, можно скачать по ссылке <http://mattmahoney.net/dc/text8.zip>, это большая выборка статей из англоязычной википедии. Большим плюсом такого текста является разнообразие тематик и, соответственно, ситуаций, в которые попадают слова.

Задачей сети будет сгенерировать цепочку символов, которые образуют лексически правдоподобный кусочек предолжения.

Заметим, что наиболее известным работающим методом генерации цепочек является, метод основанный на Марковских цепях. Интересующиеся могут познакомиться с темой здесь <https://tproger.ru/translations/markov-chains/>

# Решение

Решение с описанием в двух частях находится в папке с проектом в двух форматах на выбор для вашего удобства: .ipynb и .html

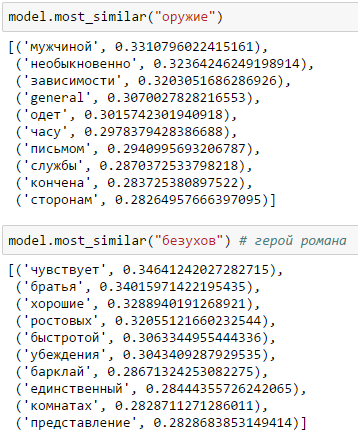
Обе части были сделаны, используя документацию и примеры из открытых источников.

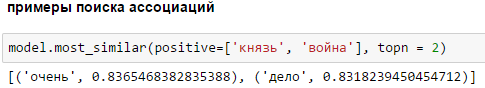
Наибольшее внимание было уделено первой части – алгоритму word2vec (он основывается на нейронных сетях): приведён пример подбора близких слов, ассоциаций, поиск лишнего слова из набора

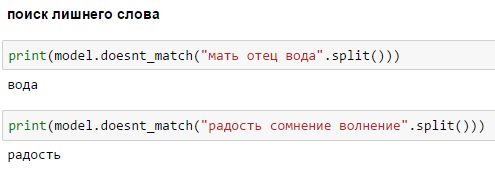
Во второй части приведены примеры сгенерированных цепочек и построен график изменения ошибки при тренировке сети.

# Результаты работы

## Часть 1







## Часть 2

Далее 5 сгенерированных текстовых последовательностей

