Document similarity using unsupervised neural model

Proposal For Google Summer of Code 2018

Mentor - Ivan Menshikh

Contact information:

Name : Sharan RY

University : BITS Pilani

Email-id : sharanyalburgi@gmail.com

Telephone: +91 91464 54088

GitHub/Gitter : sharanry

Blog : sharanry.github.io

Time-zone : Indian Standard Time (UTC + 5:30)

Organization: NumFOCUS

Sub-organization: Gensim

Abstract

Gensim is a topic-modeling package in Python for *unsupervised* learning. Deep learning is subfield of machine learning which is based on learning data representations. The goal is to develop deep learning method for Gensim whose sole task is to give similarity measure between documents. Here the goal is not efficient representation of sentence with vectors, although it could be an intermediate step. This problem is referred to as Similarity Learning**.** And we intend to build a robust deep learning based tool to solve this problem.

Technical Details

There are two major classes of learning algorithms. Supervised and Unsupervised Algorithms. Supervised learning requires labelled data, often in large quantities which is hard to acquire. Unsupervised learning algorithms on the other hand do not require labelled data. Unlabelled data is easier to acquire in large amounts. This is why we intend to develop an Unsupervised Similarity Learning Model which can efficiently gauge similarity between documents.

Document Similarity

Document similarity (or distance between documents) is a one of the central themes in Information Retrieval. How humans usually define how similar are documents? Usually documents treated as similar if they are semantically close and describe similar concepts. On other hand “similarity” can be used in context of duplicate detection. We will review several common approaches. [3]

Why Deep Learning for document similarity

This study gives us glimpse that neural network based methods like skip-thoughts are slowly catching up with the baseline vector space methods.

Implementation Details

Model Selection

Here is a list of shortlisted models for benchmarking:

Skip thought vectors

Original: Skip Thought Vectors
Original Implementation by author: Github
Recent: Trimming and Improving Skip-thought Vectors

Features:

- Vocabulary expansion
- End-to-end
- Unsupervised

My Paper Summary: https://sharanry.github.io/post/skip-thought-vectors/

Variational Autoencoders(VAE)

[Neural Variational Inference for Text Processing]
An inference network conditioned on the discrete text input to provide the variational distribution. For generative document modelling.

Recent Paper: [Improved Variational Autoencoders for Text Modeling using Dilated Convolutions]

GAN-VAE

Combining GAN and VAE
[Autoencoding beyond pixels using a learned similarity metric]
An autoencoder that leverages learned representations to better measure similarities in data space. The paper is for image similarity. Need to test the same for our use case.

[Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks]
A more recent theoretically proven technique.

DSSM Models

[Convolutional Latent Semantic Model (CLSM)]

Using 1-D convolutions and n-gram spaces along with convolutional pooling to learn similarity measure. (Currently the best performing DSSM based model.

[Details of all the considered model architectures.]

Model Implementation

The first step in the coding period will be etwo implement the above mentioned models. This would require us to select a particular framework or benchmark across various frameworks.

Our two main candidate frameworks for implementation are PyTorch and Keras.

PyTorch

Dynamic Graph: Makes makes debugging hassle free.

Release management: Changelog is much more detailed. Debugging becomes easier when updating the version.

Community Support: Active community, can get issues resolved much more quickly.

Keras

Static Graph: Makes it difficult to debug.

Developer experience: Better higher level abstractions available. Leads to less boilerplate code. Eg. Training abstraction is readily available. Saves time.

I have some amount of experience using PyTorch and Tensorflow. You can look at various model implementations on my GitHub profile. Using Keras would also not be problem as I am familiar with its API.

Benchmarking

Benchmarking is the most crucial part of this project. We will be implementing the above mentioned models and testing them on various models and then comparing them based on various metrics.

Datasets

Sentences Involving Compositional Knowledge (SICK): Given two sentences, their goal was to produce a score of how semantically related these sentences are, based on human generated scores. Scores were from 1 to 5.

Microsoft Research Paraphrase Corpus: Contains 5800 pairs of sentences which have been extracted from news sources on the web, along with human annotations indicating whether each pair captures a paraphrase/semantic equivalence relationship.

Yahoo Answers: It includes all the questions and their corresponding answers. The corpus distributed here contains 4,483,032 questions and their answers.

Metrics

Pearson's r: Correlation between sets of data is a measure of how well they are related. The most common measure of correlation in stats is the Pearson Correlation. The full name is the Pearson Product Moment Correlation. n simple terms, it answers the question, *Can I draw a line graph to represent the data?*

Spearman's ρ: The Spearman's rank-order correlation is the nonparametric version of the Pearson's r. Spearman's correlation coefficient(ρ), measures the strength and direction of association between two ranked variables.

Mean Squared Error: The mean squared error tells you how close a regression line is to a set of points.

Rank-Based Measures
Source
Normalized Discounted Cumulative Gain@k(NDCG@k): (Multiple levels of relevance) Normalize DCG at rank n by the DCG value at rank n of the ideal ranking

Precision@K(P@K): Compute % relevant in top K(rank threshold). Ignores documents ranked lower than K

Mean Average Precision (MAP): Consider rank position of each relevant doc. MAP is Average Precision(average of P@K) across multiple queries/rankings.

Mean Reciprocal Rank (MRR): MRR is the mean Reciprocal Rank across multiple queries.

Timeline

Pre-GSoC

We have spent most of the time till now trying to find different approaches in research papers and find their qualitative strengths and weaknesses. Details of this phase can be found here.

During GSoC

We have three main phases in this project.

1. Benchmarking
2. Implementation/Integration
3. Documentation and Testing

Note: My summer vacations start on May 13th just before the start of the GSoC coding period. This will enable me to turn my focus entirely towards this project.

Community Bonding
April 23, 2018 to May 14, 2018

- Being a part of gensim community, I'll continue to help around with issues and bugs, and answer queries on the community channels of gensim.
- Complete my open PRs.
- Sharing knowledge and experience is an integral part of the open source culture. I will be initiating my dedicated GSoC blog here.

Coding Period
May 14, 2018 to
August 6, 2018

Week 1 May 14 - 20	• Benchmarking selected models.
Week 2 May 21 - 27	• Benchmarking continued.
Week 3 May 28 - June 3	• Compare the results of the different models and post results
Week 4 June 3 - 10	• Finish benchmarking.
Evaluation 1 June 11 - 15	• Discuss the benchmarking results and finalise which model to use.
Week 6 June 16 - 23	• Integration with Gensim • Test the model with different data sets and different available user configurations
Week 7 June 24 - 30	• Blog the testing methodology and results. • Analyse the results. • Fix bugs if any.
Week 8 July 1 - 8	• Write tutorials and blog posts explaining the usage of the similarity function.
Evaluation 2 July 9 - 13	• Address reviews on the tutorial.
Week 10 July 14 - 21	• Make any changes to the code/documentation if necessary.
Week 11 July 22 - 28	• Clean code make the use of the model user friendly.
Week 12 July 29 - August 5	• Clean code, add proper docstring, fix possible bugs.

Submit Code and Final Evaluations
August 6 - 14, 2018

- Have the PRs merged :)

Post GSoC

This phase will be mainly involve:

1. To maintain the code.
2. Fix any potential bugs.
3. Increase the ease of use for the users.
4. Answer communities queries.

Development Experience

(Models Implemented, SOP, SWD, ARC)

Currently working on two research oriented study projects:

Conditional Random Fields for Image Segmentation [Link]

This has till now been a study oriented project in the field of conditional random fields. We are trying to understand their significance in the field of semantic segmentation of images. With this understanding we aim to contribute to furthering of its capabilities and use to the field.

Nested Mini-Batch K-Means [GitHub]

We are exploring the use of this new and improved unsupervised learning technique in the domain of visual recognition.

Student Welfare Division [GitHub]

A large scale application by our institute for the everyday usage of the students to increase ease of access of campuses facilities. Aim was to make it future proof.

Academic Research Division [GitHub]

A course management tool for the use of institute. This automates repeated tasks and increases the productivity of the staff.

Models Implemented [GitHub 1] [GitHub 2] [GitHub 3]

Links to some of the models which I have implemented.

Open Source Contributions:

I have previously contributed to PyMC and Scikit-Learn.

My PRs are available here.

Pre-GSoC Involvement

Here is a list of PRs I have contributed to:

[MERGED]Add anaconda-cloud badge. Partial fix #1901

[MERGED]Add method that show base installation info about gensim & related packages. Fix #1902

[MERGED]Replace open() with smart_open() in notebooks. Fix #1789

Active issues:

Support for OOV words using only word-vectors

Why Me?

I am an enthusiastic self-learnt student of machine learning and a growing open source enthusiast. I have experience working with large code bases and small teams of developers. I have previous experience in building deep learning models, and so I am familiar with popular frameworks. I am keen on research, essential for this project in particular. I feel it is these small but put together significant things that makes me a good fit to take on such an ambitious project.

Why Gensim?

My first introduction to Gensim was by Lev Konstantinovskiy at his talk in our campus BITS Goa. I was not just fascinated by Gensim but also by the capabilities of Natural Language Processing techniques. This inspired me to start making small contributions from late 2017. Through my seniors in college I have gotten to understand the importance of Gensim in today's world of text modelling. It's important it keeps up with everyday developments in this field. I feel it's my duty as a student of machine learning to contribute to its maintenance and growth in any way I can. This year's GSoC is an opportunity to do the same.

Acknowledgement

I would like to thank Ivan Menshikh for being patient with me and answering all my queries.

Appendix and References

[1] http://text2vec.org/similarity.html

[2] https://radimrehurek.com/gensim/similarities/docsim.html

[3] https://cs224d.stanford.edu/reports/PoulosJackson.pdf

[4] http://ad-publications.informatik.uni-freiburg.de/theses/Bachelor_Jon_Ezeiza_2017.pdf

[5] Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features: https://arxiv.org/pdf/1703.02507.pdf

[7] http://forums.fast.ai/t/keras-vs-pytorch/2934

[8] https://web.stanford.edu/class/cs276/handouts/EvaluationNew-handout-6-per.pdf

[9] https://www.kaggle.com/wendykan/ndcg-example

■■■