Similarity learning with Gensim

Contact Details

Name: Mritunjay Mohitesh

University: National Institute of Technology, Delhi

E-mail : mritunjaymohitesh@gmail.com

Phone: +91 9540092315

Twitter : @MMohitesh

Skype : mritunjaymohitesh@outlook.com

Github : github.com/MritunjayMohitesh

Abstract

Gensim is a nlp library that stands for generate similar. To assess the similarity between sentences/ words, the traditional methods use some similarity functions like cosine, jaccard after generating the vector representations of words/sentences. The sequential nature of natural language is taken into account mostly through word n-grams and skip-grams which capture distinct slices of the analyzed texts but do not preserve the order in which they appear. Hence, I aim to introduce implement a new similarity measure called TexFlow on the top of gensim. It is a supervised learning that captures the semantics with full sequence of words with a few parameters. NNs are used to train the respective parameters.

Next, I aim to improve various modules of summarization sub-package. I have been analyzing various bottlenecks across this package and quite a few improvements can be worked around. This projects aims at cython implementation of some time consuming modules of summarization package.

Technical Details

Second Phase:

This proposed model called TextFlow works under supervised mode. The intuition is inspired from the dot matrix representation used in pairwise DNA sequence alignment (fig-1). For practical purposes, we transform the fig-2 using delta of positons (of words) where the Y axis is the delta of positions of a word occurring in the two texts being compared. If the word does not occur in the target text, the delta is considered to be a maximum value (fig-3). So, the bigger the

area under the curve the more dissimilar they are. This method not takes into consideration not only matched words but also the sub-sequences at the same time. The obtained value called XF value lies in [0,1]. Even in the worst case , XF computation is performed in O(mn) time where m and n are lengths of two sequences. We introduce three parameters - (Position factor α), missing words (sensitivity factor β), and sub-sequence matching (sequence factor γ). Various canonical parameters used in this setting can be learned on training data for specific task. The NN has been designed with a hidden layer to compute the exact XF–value(fig-4).

To evaluate the model, a new evaluation measure called Consistent Performance(CORE) has been devised. For a system m, a set of datasets D, a set of systems S, and an evaluation measure E ∈ {F1,Precision,Recall,Accuracy}:

$$\underset{D,S,E}{CORE}(m) = \frac{\underset{p \in S}{MIN}\left(\underset{d \in D}{AVG}(R_S(E_d(p)) + V_{d \in D}(R_S(E_d(p)))\right)}{\underset{d \in D}{AVG}\left(R_S(E_d(m))\right) + V_{d \in D}\left(R_S(E_d(m))\right)}$$

Where $R_s(E_d(m))$, the rank of m according to the evaluation measure E on dataset d w.r.t. competing systems S and V_d $(R_s(E_d(m)))$, is the rank variance of m over datasets.

Since this is a supervised task, a concrete idea regarding dataset to be used is very important. Hence, I have made a list regarding the relevant datasets to be used and ways to incorporate data from various sources.

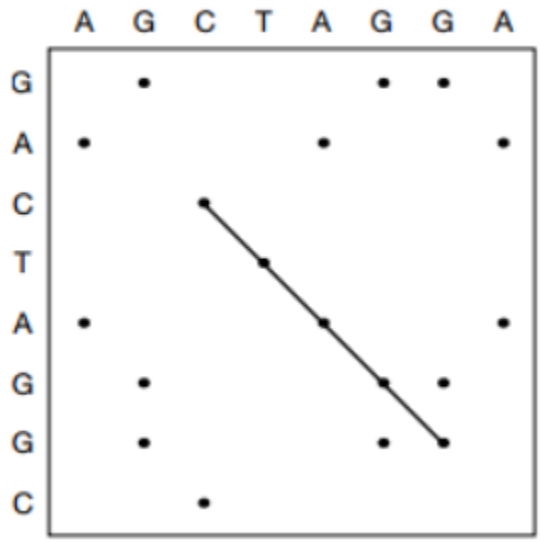

fig 1 – Dot matrix for DNA sequences

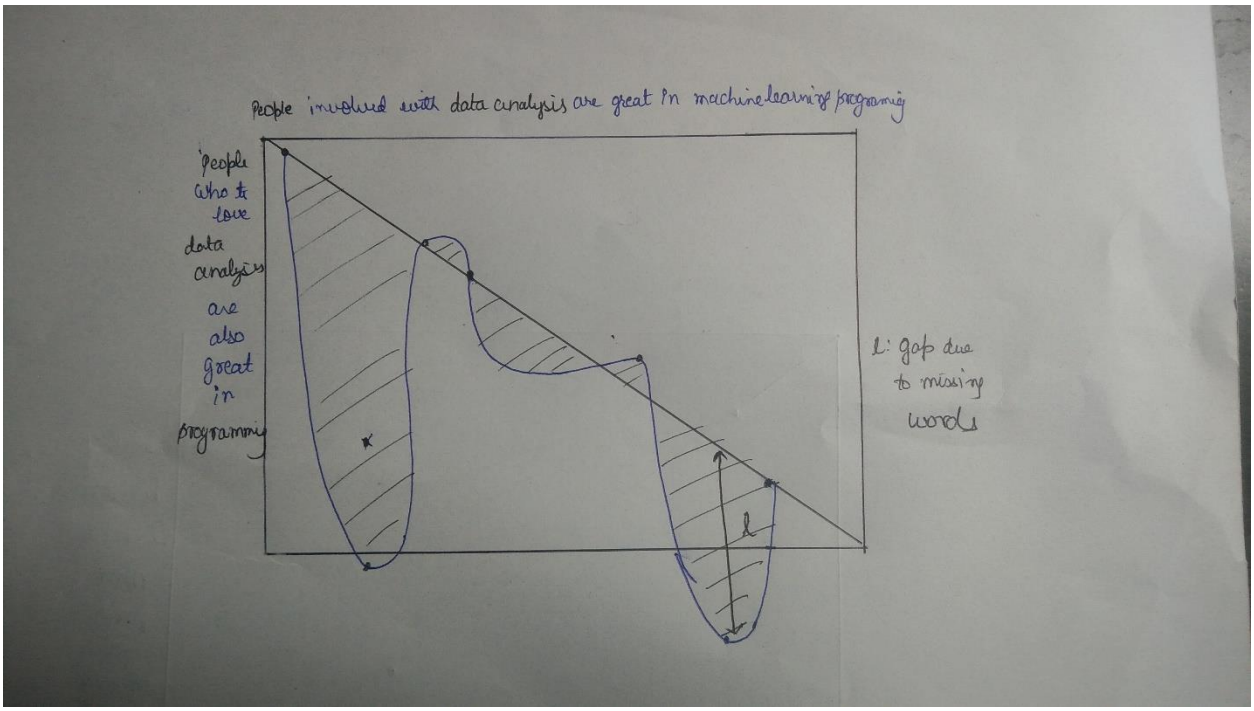

Fig 2- TextFlow Intution

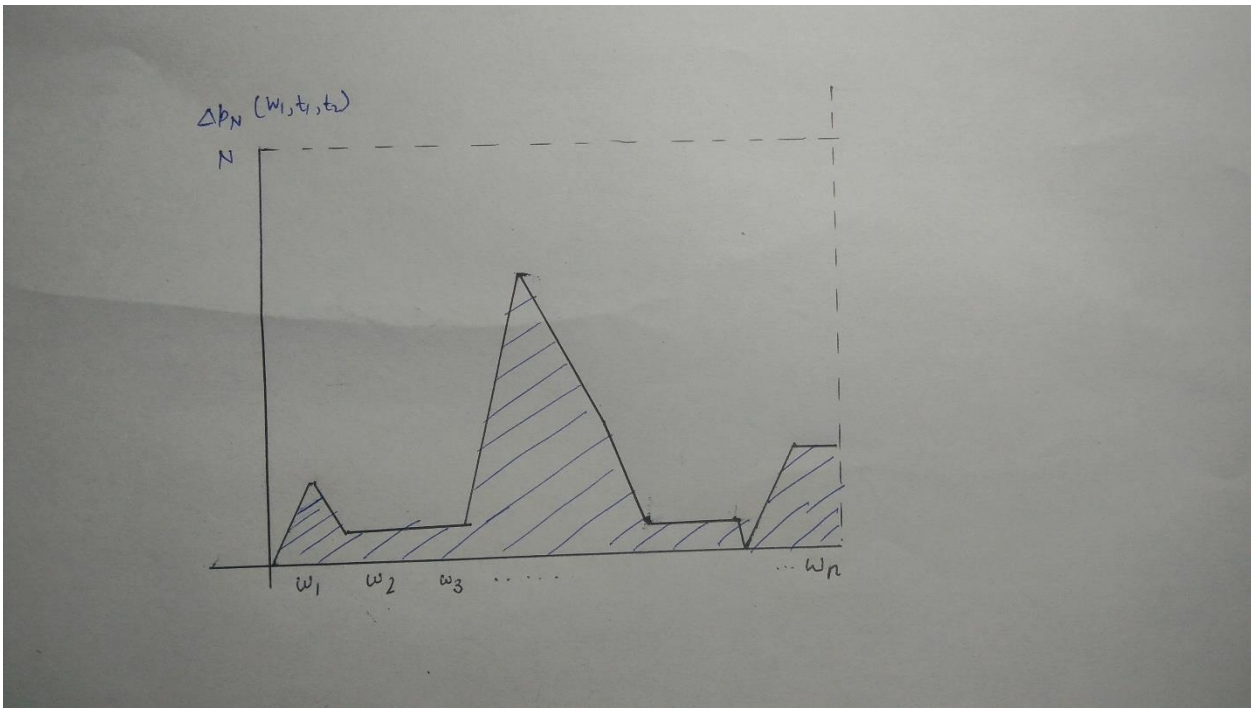

Fig 3 - Practical Texflow Computation

First Phase:

Gensim has a summarization sub-package that contain various modules for finding the summary of text using a variation of TextRank algorithm. One such module is summarizer.py. This module constructs a graph where node represents the sentences and edges represent the weights. This module has a summarize_corpus() function that takes a preprocessed corpus and finds its summary by constructing an immediate graphical interface.

```
 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
```

```
255(_extract_important_sentences)
       1    0.000    0.000    0.000    0.000 summarizer.py:284(_format_results)
       1    0.000    0.000    0.000    0.000 summarizer.py:304(<listcomp>)
       2    0.000    0.000    0.004    0.002 summarizer.py:307(_build_hasheable_corpus)
       2    0.004    0.002    0.004    0.002 summarizer.py:321(<listcomp>)
       1    0.034    0.034   40.102   40.102 summarizer.py:324(summarize_corpus)
    1294    0.002    0.000    0.005    0.000 summarizer.py:371(<lambda>)
       1    0.001    0.001    0.001    0.001 summarizer.py:373(<listcomp>)
       1    1.580    1.580   43.145   43.145 summarizer.py:376(summarize)
     258    0.000    0.000    0.000    0.000 summarizer.py:441(<lambda>)
       1    1.811    1.811   23.567   23.567 summarizer.py:72(_set_graph_edge_weights)
    1294    0.003    0.000    0.003    0.000 syntactic_unit.py:30(__init__)
```

We can see that summarize_corpus() accounts for the maximum time. The function _set_graphh_edge_weights is called inside summarize_corpus() that set edge weights using bm25 algorithm. The whole implementation is in pure python. So, speed is quite slow as of now.

Besides, the implementation of BM25 algorithm is also straightforward and it can also be tuned. We can see that that get_bm25_weights turns out to be a bottleneck.

```
      23    0.000    0.000    0.000    0.000 base.py:100(get_shape)
       1    0.000    0.000    0.000    0.000 base.py:1028(_process_toarray_args)
       3    0.000    0.000    0.000    0.000 base.py:1111(isspmatrix)
      18    0.000    0.000    0.000    0.000 base.py:193(nnz)
       1    0.000    0.000    0.086    0.086 base.py:249(asformat)
       1    0.000    0.000    0.017    0.017 base.py:691(todense)
       5    0.000    0.000    0.000    0.000 base.py:70(__init__)
       6    0.000    0.000    0.000    0.000 base.py:77(set_shape)
 1340964    7.625    0.000    7.625    0.000 bm25.py:104(get_score)
    1158    2.017    0.002    9.971    0.009 bm25.py:131(get_scores)
       1    0.005    0.005   10.027   10.027 bm25.py:155(get_bm25_weights)
```

Possible Improvements:

- Efficient memory access
- Numpy(for bm25)
- Matrix in place of for loops(for _set_graphh_edge_weights())
- Cythonize
- Parallelize

Schedule of deliverables

I have been using genism since the past year for various nlp assignments and I am trying to get acquainted with the genism code base for almost one and a half month. So, my main motive would be to implement the proposed plans in a way that users like me don't have to go in trouble for understanding and visualizing the implementation of model.

Community Bonding Period

- One of my main motive during this period would be to discuss and think of various possible modifications around the TextFlow model.
- To get more familiar with the code base by participating in various community discussions.
- As a part of genism community, I'll help around with issues or bugs or any query and will continue to take participate in mailing list.
- Get more confident on deep learning libraries especially PyTorch.
- Discuss possible modifications in summarizer module

Week 1(May 14- May 20)

- Start with summarize() function in summarizer module and refactor using various optimization methods.
- Benchmark the improvements
- Write cython code for summarize() and also work on se_graph_edge_weights() function

Week 2(May 21-May 27)

- Refactor bm25.py module and create cython version for it
- Try to create a multi-core version for summarizer.py

Week 3(May 28-June 3)

- Write a blog on the work done so far
- Finish multi-core versions

- Create a jupyter notebook tutorial(if needed) and benchmark the improvements

Week 4(June 4 –June 10)

- Finish anything if remaining
- Organize everything done so far in the standard genism way.
- Test them and debug them

Mid Term evaluation (June 11- June 15)

- Work on any feedback received
- Give a final touch to phase 1
- Phase 1 ends

Week 5(June 16 – June 22)

- Discuss the model ideas with mentors and get feedback
- Finalize the design and architecture
- Work on API design and decide the user interface

Week 6-7(June 23 – July 8)

- Start implementing the model
- Work on parallel training code on cython
- Write a blog on the work done so far

Phase 2 evaluation (July 9 – July 13)

- Clean and refine the code written so far
- Work on feedback

Week 9(July 14-July 21)

- Work on remaining optimization tasks and create benchmark
- Start working on a new evaluation measure called CORE

Week 10(July 22 – July 28)

- Finish the evaluation task and generate benchmark results

Week 11(July 29 – August 5)

- Create a notebook tutorial regarding the model
- Update the blog
- Discuss and address reviews and comments regarding the model

Week 12(August 6 – August 14)

- Give final touch to everything for final submission

- Add proper docstrings
- Fix bugs(if any)
- Work on feedback(if any)
- In short, bundle up everything

About Me

I am a third year Computer Science and Engineering undergrad at National Institute of Technology, Delhi. I have a deep interest in data analysis and visualization, and Deep learning. Any kind of data makes me curious and I always try to come up with some practical visualization from the data. I have worked on Web Development, Android development (not a lot) as well. But my focus has always been on data and machine learning. I have been working on this almost two years now and I am pretty confident to make some remarkable contributions in this field.

Proficiency:-

Programming Languages: C/C++, Python, Cython
Libraries : sklearn, keras, genism , PyTorch
IDE : Anaconda (IPython Notebook and Spyder) , Sublime Text

Open Source Development Experience

To be honest, I am kind of beginner in the open source world. I started contributing with genism only and have been active in discussions regarding various issues on github. I worked on following PRs

#1945 (merged) Fix #1937 : **Fix deprecated parameters in `D2VTransformer` and `W2VTransformer`**

Why this project

I started using genism almost a year and half before when I was assigned the task of finding the similar and/or fake reviews on Amazon electronics review dataset. I came across the world of nlp and have been using various libraries for various nlp tasks. After I started studying the code(on genism, nltk), I came to know about the real essence of various machine learnings tasks like topic modelling and similarity learning. This has inspired me to take a part in this esteemed community and contribute something significant.

Acknowledgment

- Ivan Menshikh @menshikh-iv . Thanks for replying to all of my mails.
- Radim Řehůřek @piskvorky . Thanks for your valuable feedback
- Gensim community

References

1. Gsoc 2018 guide
2. TextFlow
3. Text Similarity approaches
4. Text Similarity Demo
5. Multiple cores with Python