Replicate QDLinker with Google Colab

Wenbo Wang, New Jersey Institute of Technology

Agenda

- Background
- QDLinker introduction
- Related work
- Why I choose Google colab
- Steps to replicate

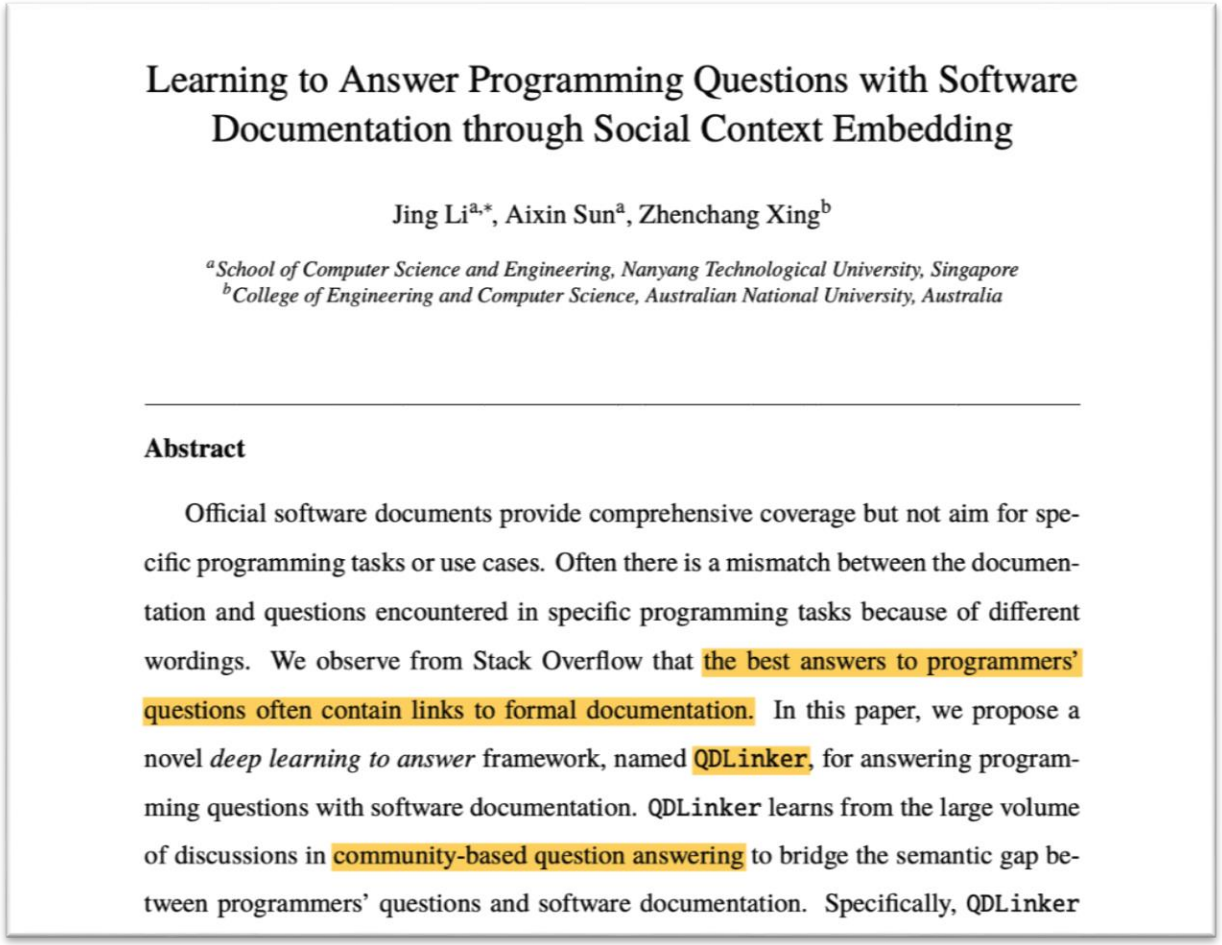
Background

- The growing needs to explore more and more spatial data and concepts.
- GIS related software have too many functions [1]
- There are more than 26,000 questions waiting to be answered in the GIS Q&A forum.

[1] S. Gao and M. F. Goodchild, "Asking Spatial Questions to Identify GIS Functionality," 2013 Fourth International Conference on Computing for Geospatial Research and Application, San Jose, CA, 2013, pp. 106-110, doi: 10.1109/COMGEO.2013.18.

QDLinker Introduction

QDLinker



Learning to Answer Programming Questions with Software Documentation through Social Context Embedding

Jing Li[a,*], Aixin Sun[a], Zhenchang Xing[b]

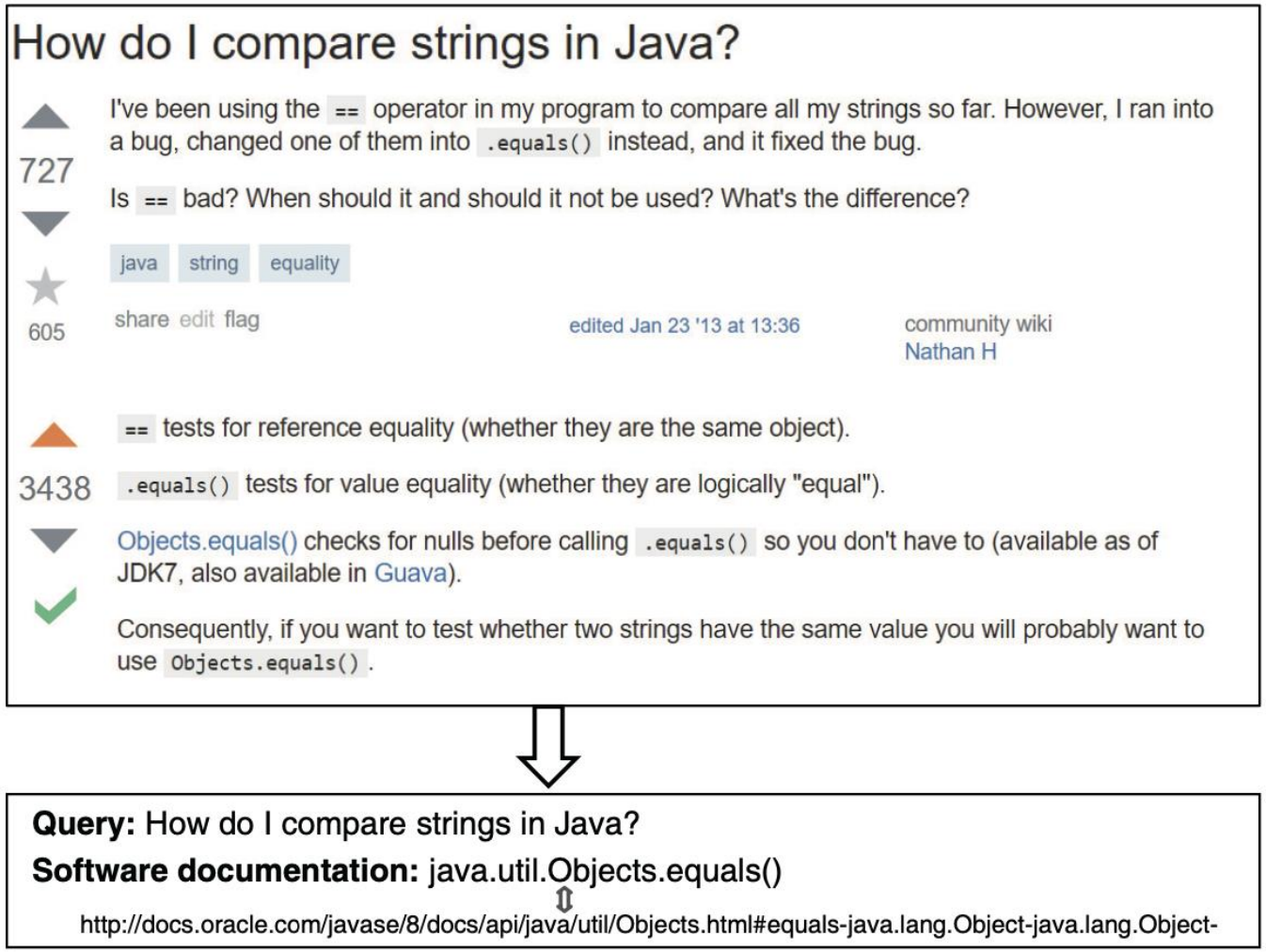
[a]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[b]College of Engineering and Computer Science, Australian National University, Australia

Abstract

Official software documents provide comprehensive coverage but not aim for specific programming tasks or use cases. Often there is a mismatch between the documentation and questions encountered in specific programming tasks because of different wordings. We observe from Stack Overflow that the best answers to programmers' questions often contain links to formal documentation. In this paper, we propose a novel *deep learning to answer* framework, named QDLinker, for answering programming questions with software documentation. QDLinker learns from the large volume of discussions in community-based question answering to bridge the semantic gap between programmers' questions and software documentation. Specifically, QDLinker

li, Jing & Sun, Aixin & Xing, Zhenchang. (2018). Learning to answer programming questions with software documentation through social context embedding. Information Sciences. 448. 10.1016/j.ins.2018.03.014.

An example of dataset



Empirical study

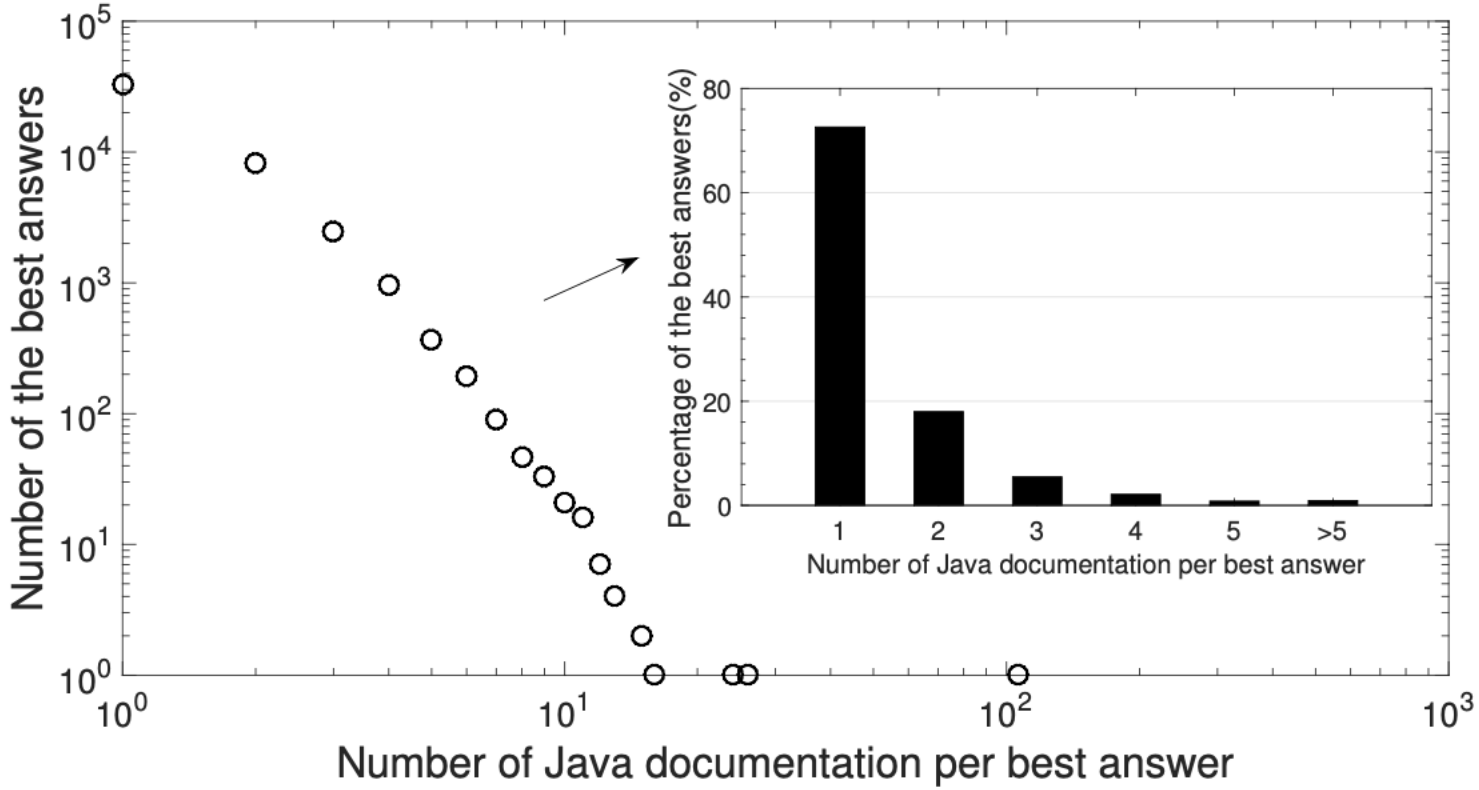
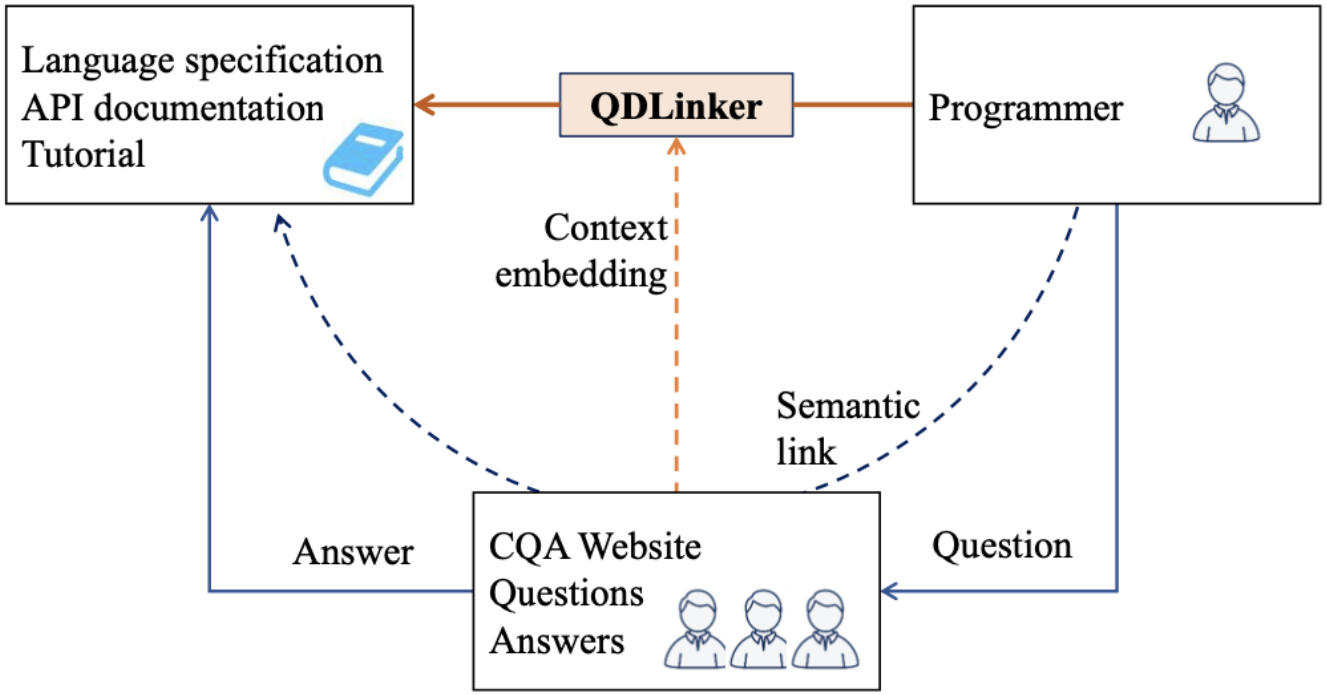
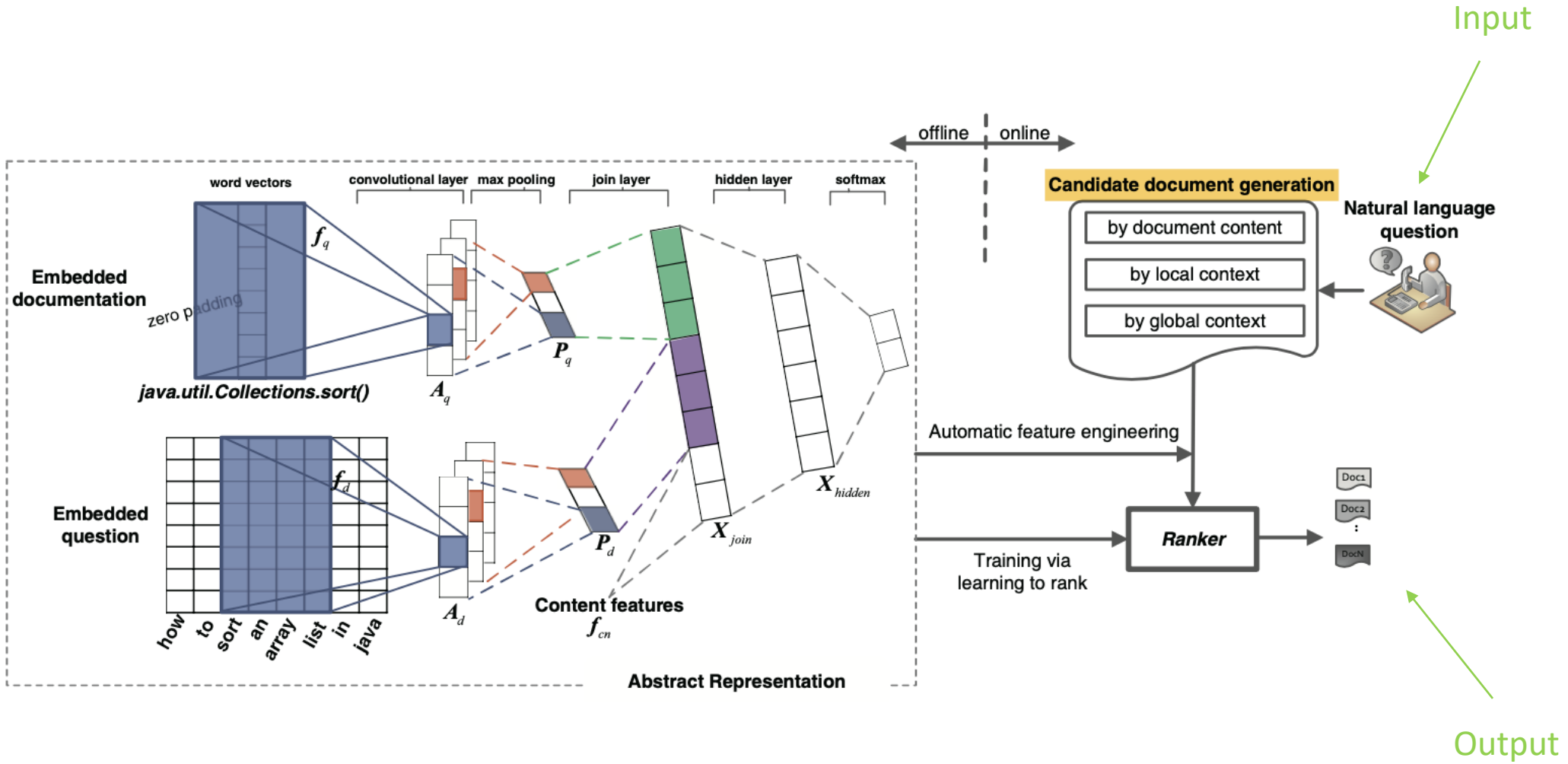


Figure 1: Distribution of links to Java documentation per best answer, among 45,288 best answers from Stack Overflow. The absolute numbers are plotted in log scale, and the percentages are plotted in bar chart.

Overview of QDLinker



Architecture of QDLinker



a) Social Context Embedding

- Two vocabularies
  - English words
  - Links to software documentation.
- Models
  - Mikolov's continuous bag-of-words and skip-gram languages (implemented in Word2vec)

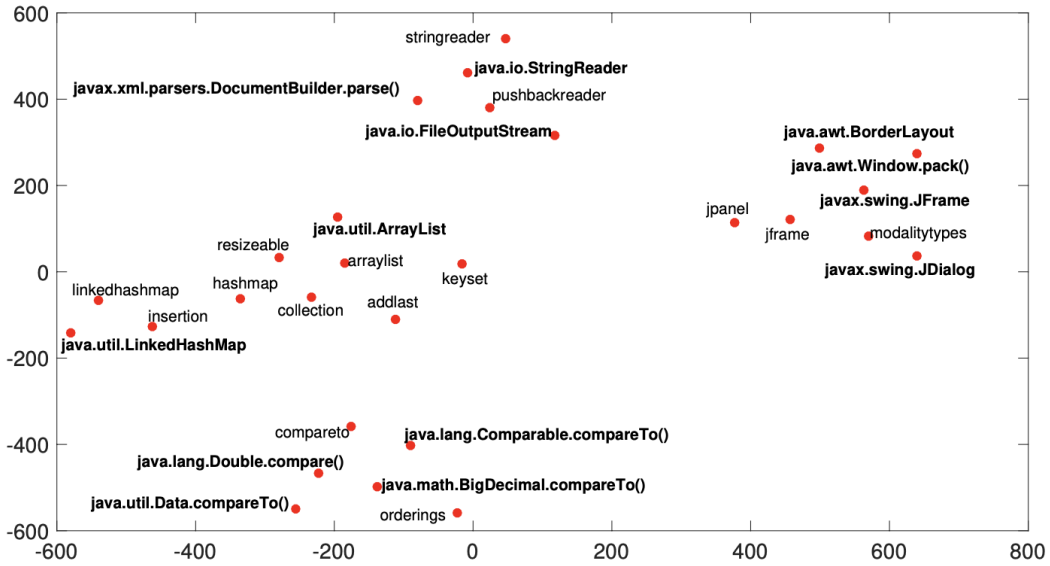


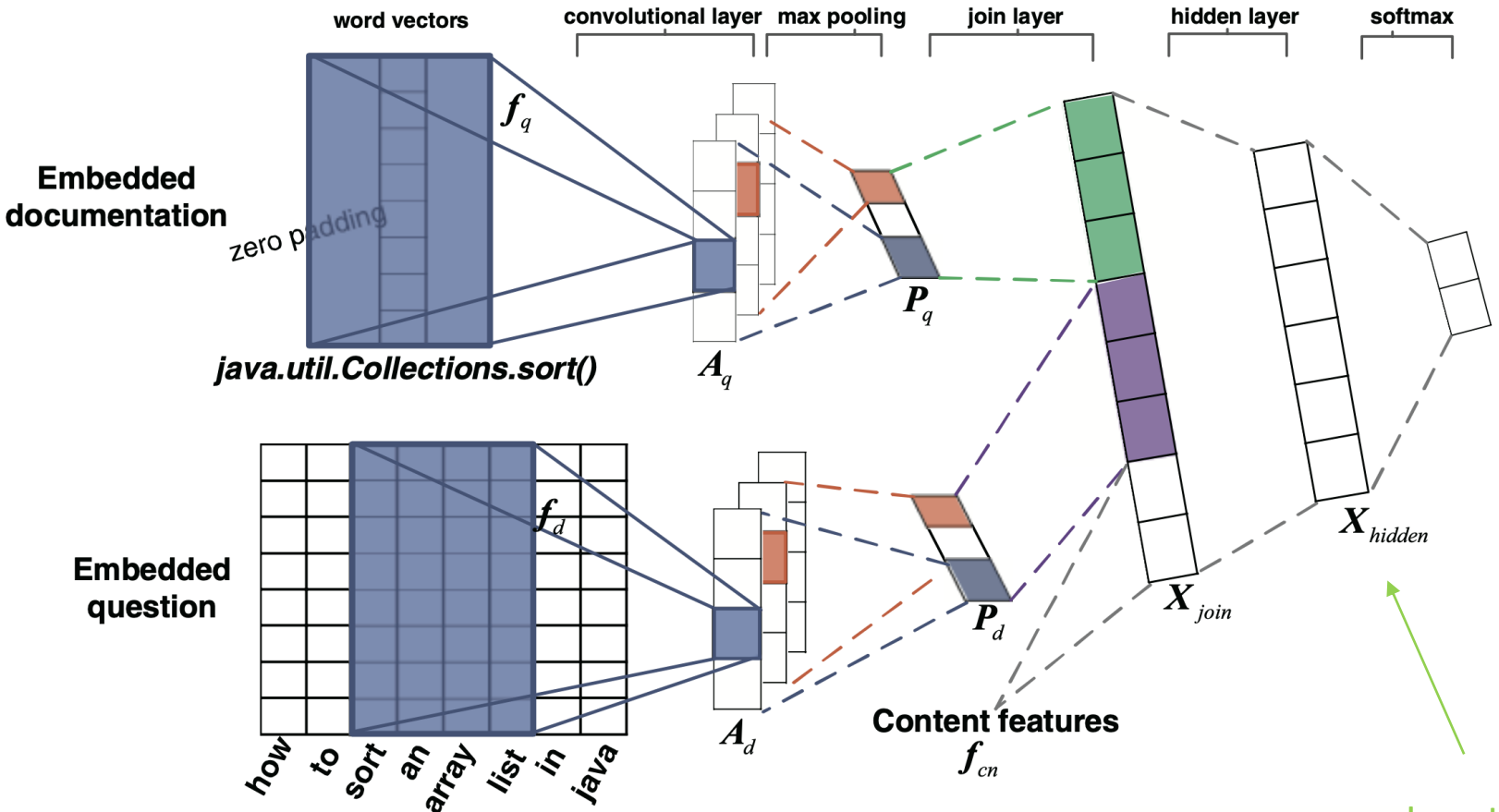
Figure 5: A 2D projection of embedding natural language terms and API documentation using PCA (API documentation in bold font and natural language terms in non-bold font).

Mikolov, Tomas & Sutskever, Ilya & Chen, Kai & Corrado, G.s & Dean, Jeffrey. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems. 26.

b) Candidate Document Generation

- **Document Content.** The content of software document reflects its relevance to a given query.
- **Local Context.** When a software document appears in a discussion thread, its surround texts reflect its relevance to the query question.
- **Global Context.** The global context of a software document is the collection of all its local contexts.

c) Four-layer Deep Neural Network



Input of Learning2rank

d) Learning a Ranker

- Learn a scoring function *F (q, d)*:

$$F(q,d) = \sum_{k=1}^{K} \omega_i \cdot \phi_i(q,d)$$

where each feature $\phi_i(q,d) \in \boldsymbol{X}_{hidden}$ measures a specific relationship between the query and a candidate software document. ω_i is the weight of the i-th feature (among the total K features), and is learned during the training process.

- Train the ranking model using LambdaMART

Related Work

Related work - 1

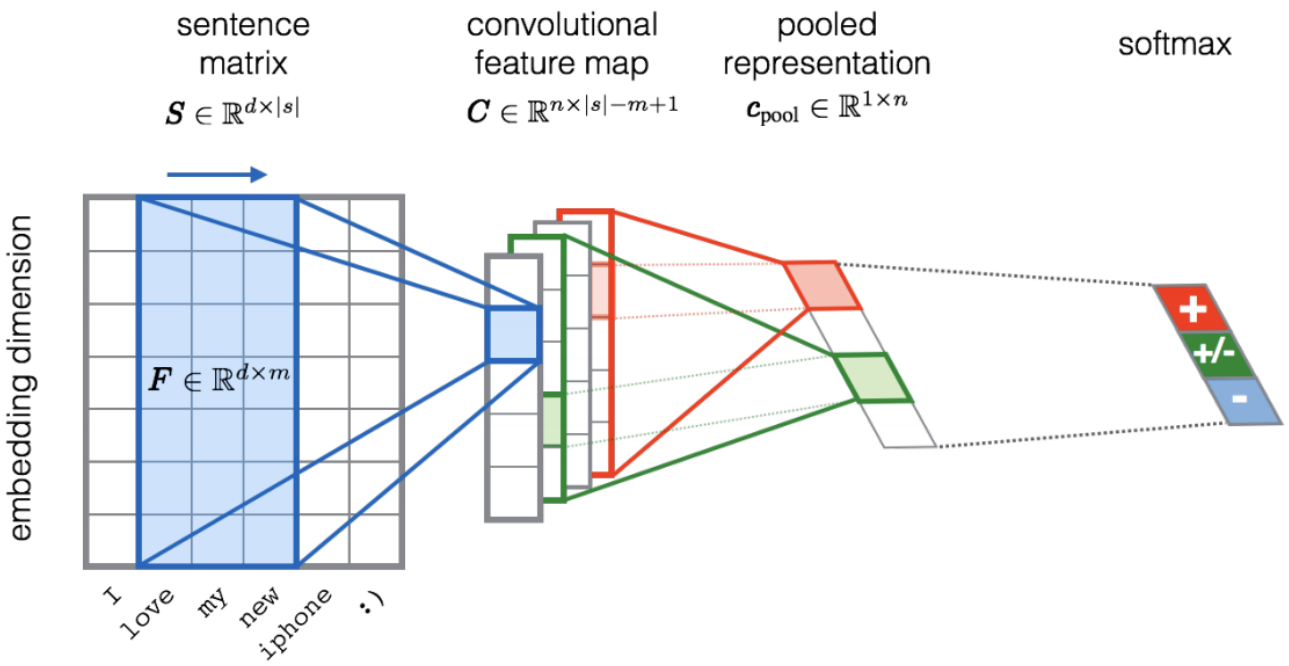


Figure 1: The architecture of our deep learning model for sentiment classification.

Severyn, Aliaksei & Moschitti, Alessandro. (2015). UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. 10.18653/v1/S15-2079.

Related work - 2

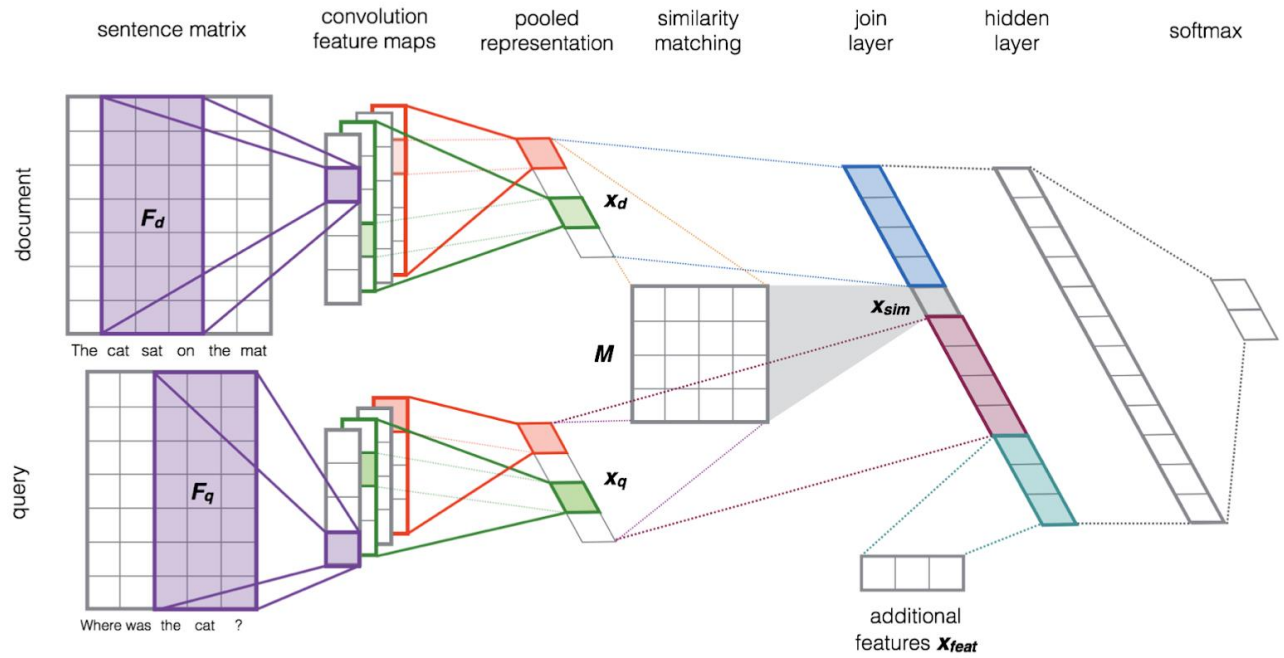


Figure 2: Our deep learning architecture for reranking short text pairs.

A. Severyn, A. Moschitti, Learning to rank short text pairs with convolutional deep neural networks, in: SIGIR, 2015, pp. 373–382.

Related work - 3

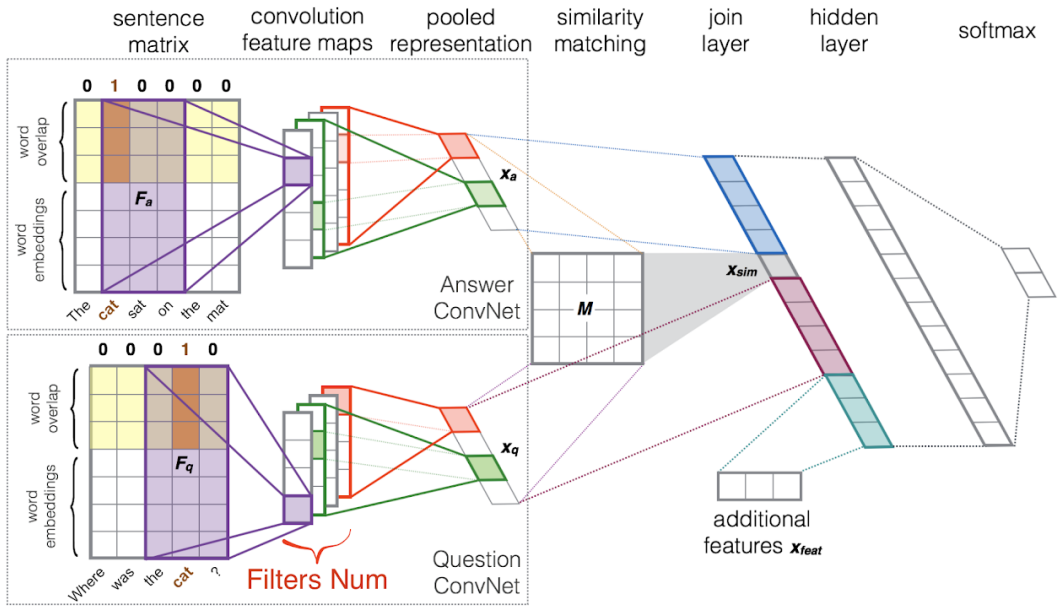
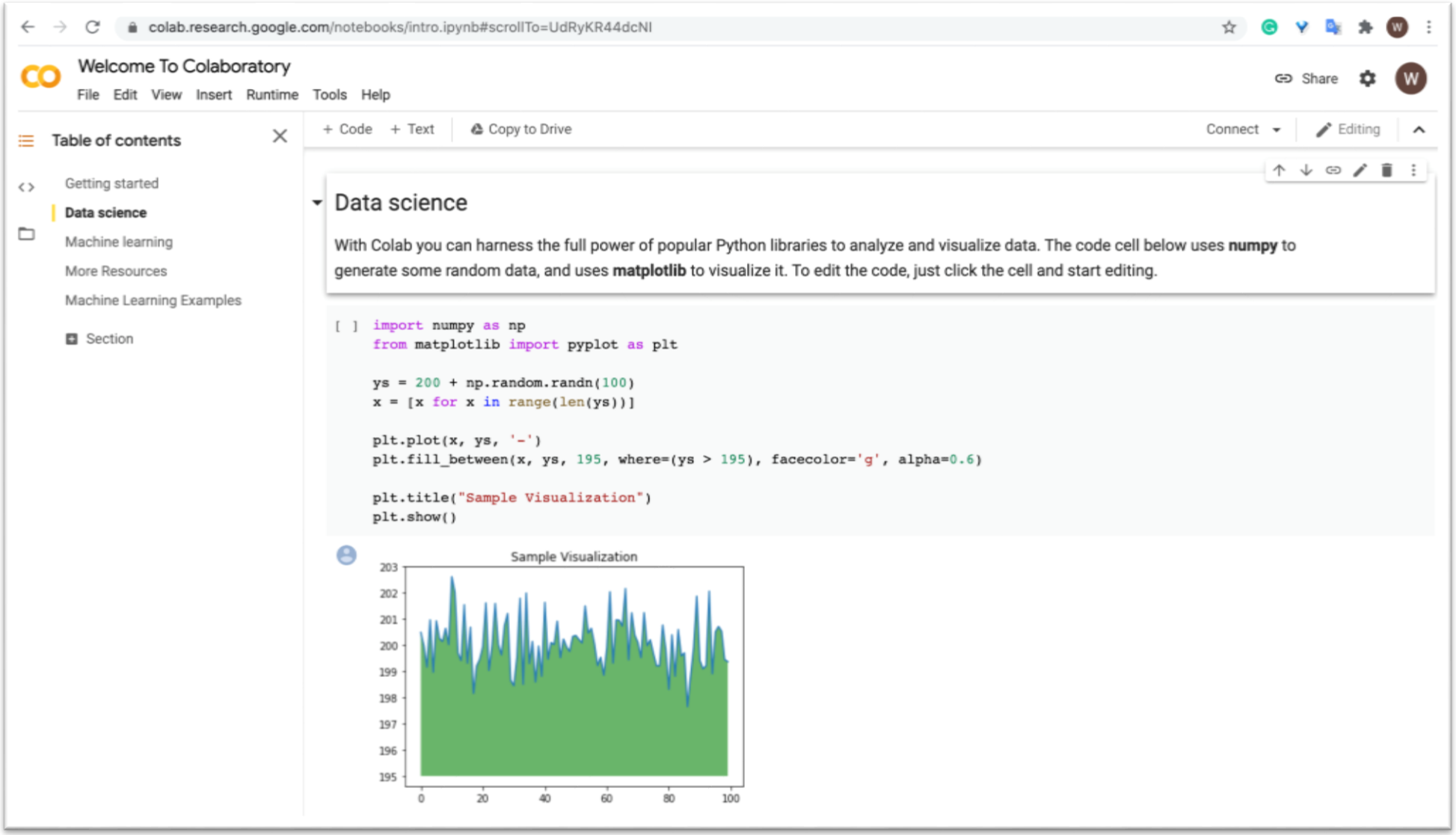


Figure 2: Our deep learning architecture for reranking question-answer pairs. The relational information in a pair is modelled by augmenting word embeddings with additional dimensions to encode overlapping words, e.g., we feed the network with additional *word overlap* indicator features whose values equal to 1 correspond to words that overlap in a pair, e.g., a non-stop word `cat`.

Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. arXiv preprint 1243 arXiv:1604.01178 (2016).

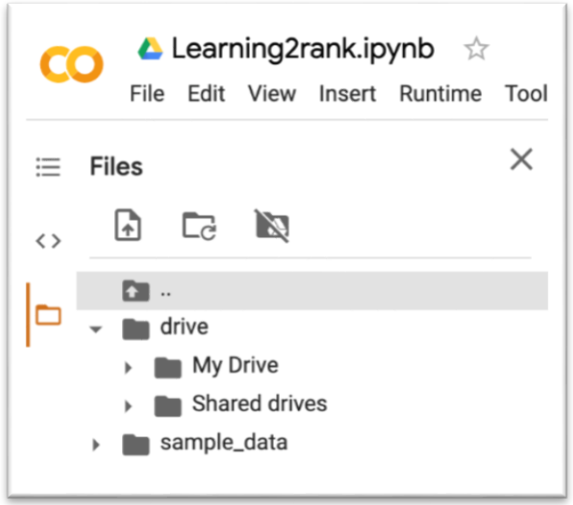
Why I choose Google colab

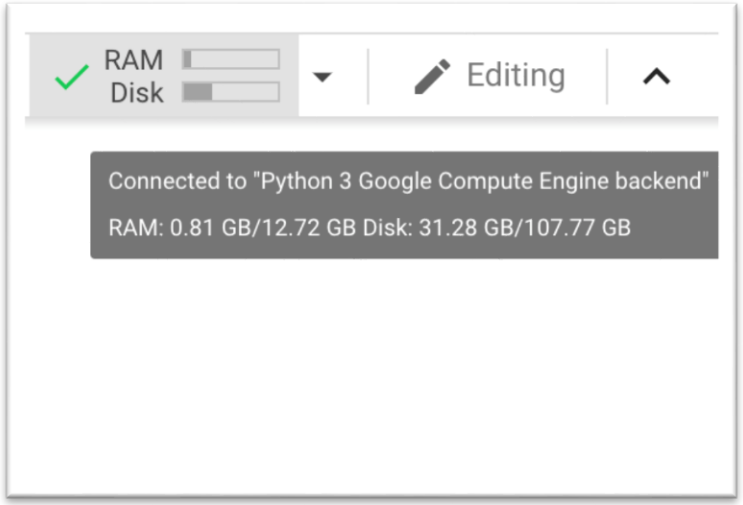
Google Colab

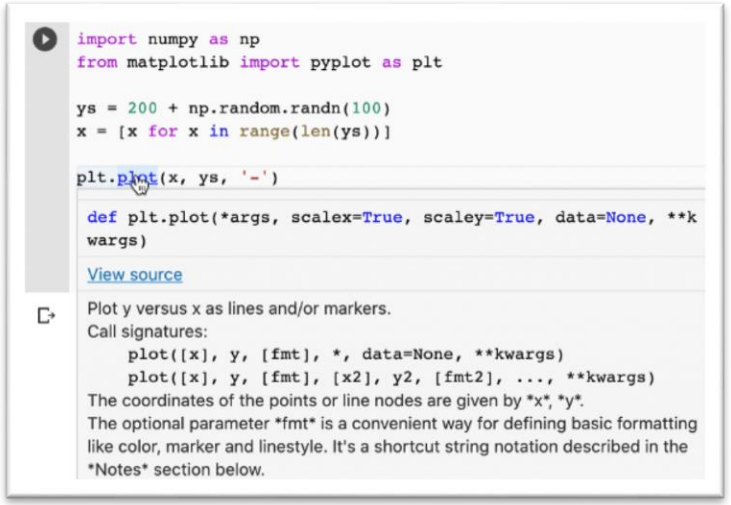


Why I choose Google Colab

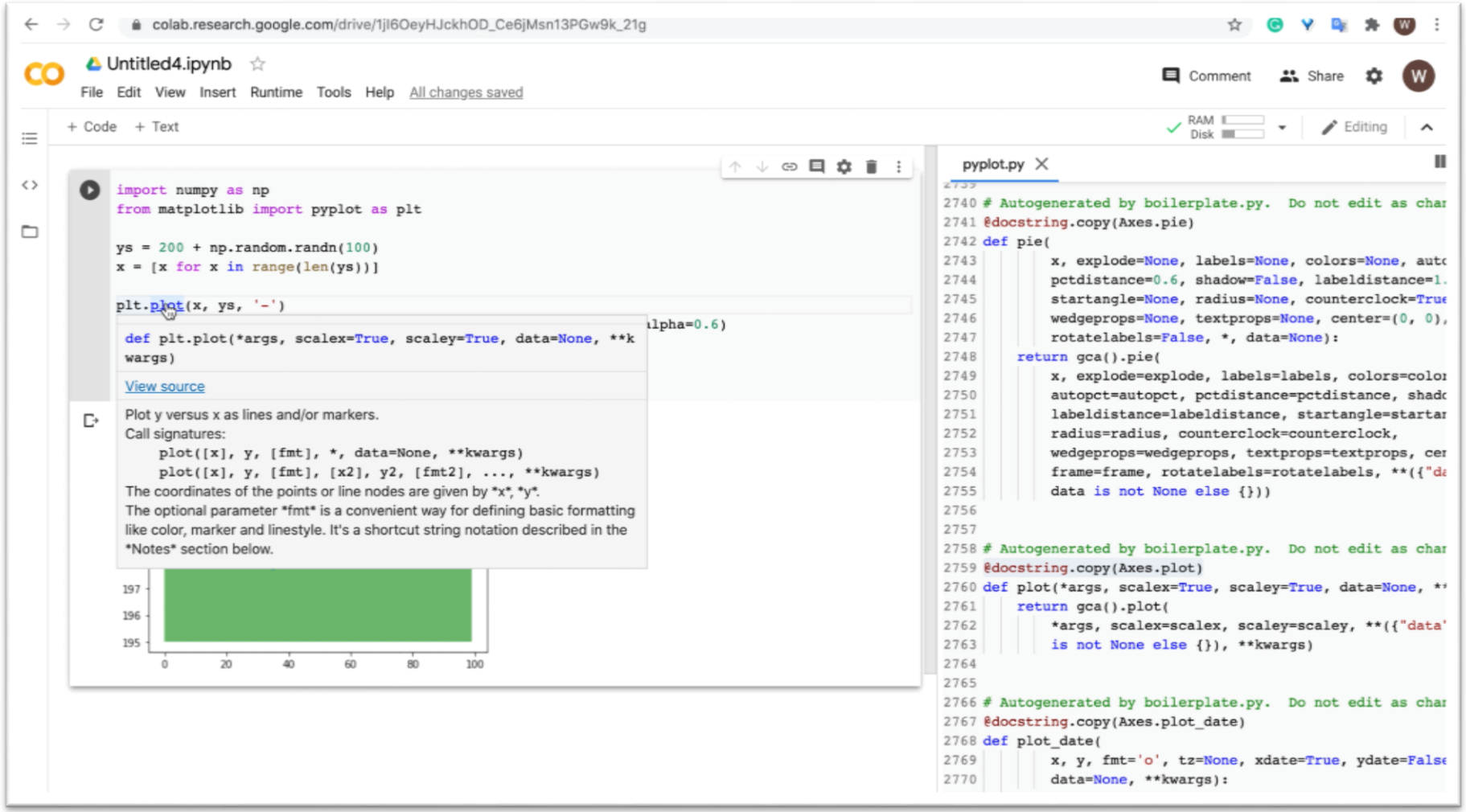
- Connect with Google Drive
- With GPU
- Syntax tips, highlighting
- Share with others
- Easy to debug







Examples

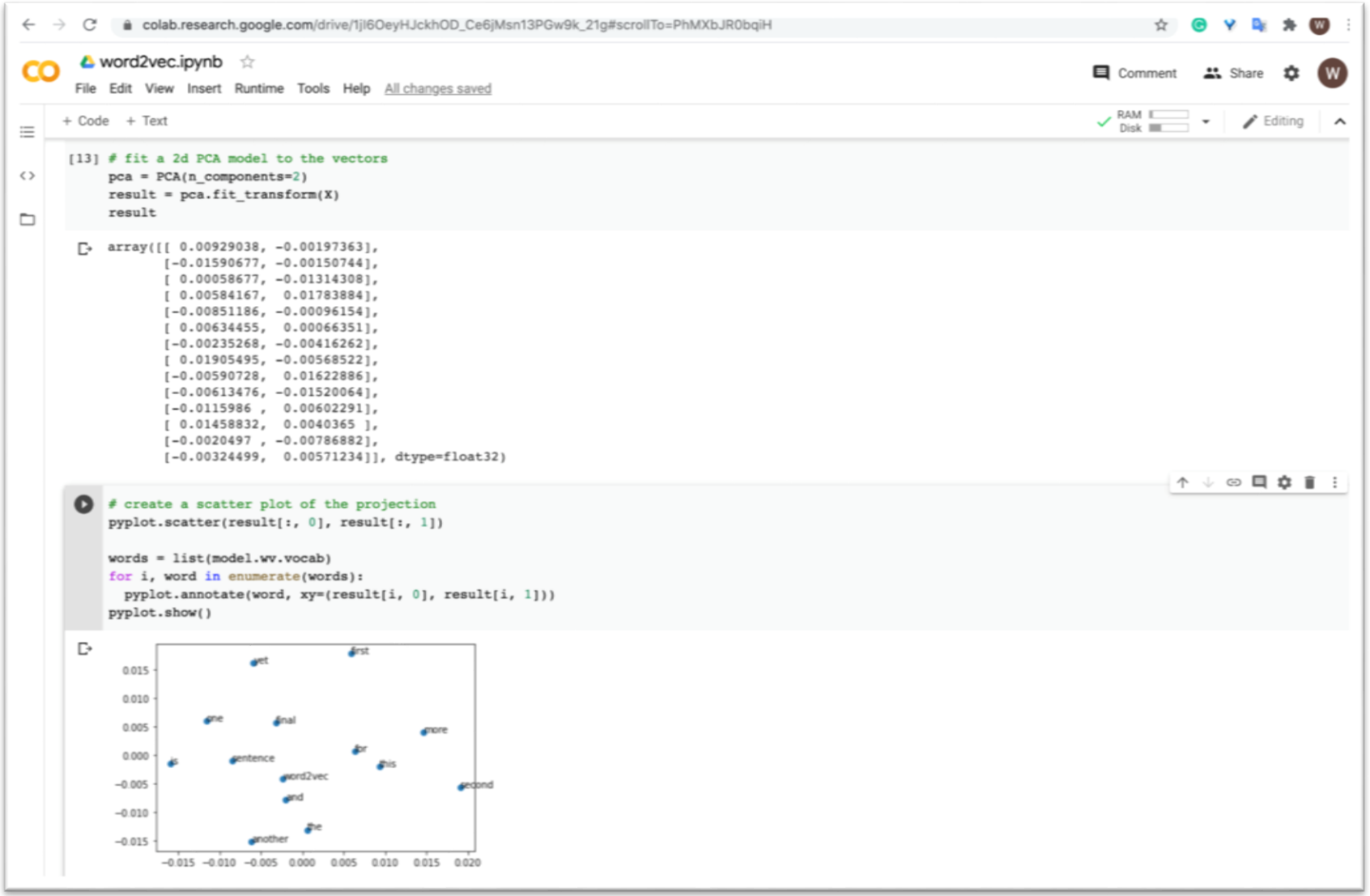

colab.research.google.com/drive/1jI6OeyHJckhOD_Ce6jMsn13PGw9k_21g
Untitled4.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
import numpy as np
from matplotlib import pyplot as plt
ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]
plt.plot(x, ys, '-')
def plt.plot(*args, scalex=True, scaley=True, data=None, **kwargs)
View source
Plot y versus x as lines and/or markers.
Call signatures:
plot([x], y, [fmt], *, data=None, **kwargs)
plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)
The coordinates of the points or line nodes are given by *x*, *y*.
The optional parameter *fmt* is a convenient way for defining basic formatting like color, marker and linestyle. It's a shortcut string notation described in the *Notes* section below.
197
196
195
0
20
40
60
80
100
pyplot.py
2740 # Autogenerated by boilerplate.py. Do not edit as char
2741 @docstring.copy(Axes.pie)
2742 def pie(
2743 x, explode=None, labels=None, colors=None, auto
2744 pctdistance=0.6, shadow=False, labeldistance=1.
2745 startangle=None, radius=None, counterclock=True
2746 wedgeprops=None, textprops=None, center=(0, 0),
2747 rotatelabels=False, *, data=None):
2748 return gca().pie(
2749 x, explode=explode, labels=labels, colors=color
2750 autopct=autopct, pctdistance=pctdistance, shadc
2751 labeldistance=labeldistance, startangle=startar
2752 radius=radius, counterclock=counterclock,
2753 wedgeprops=wedgeprops, textprops=textprops, cer
2754 frame=frame, rotatelabels=rotatelabels, **({"da
2755 data is not None else {}))
2756
2757
2758 # Autogenerated by boilerplate.py. Do not edit as char
2759 @docstring.copy(Axes.plot)
2760 def plot(*args, scalex=True, scaley=True, data=None, **
2761 return gca().plot(
2762 *args, scalex=scalex, scaley=scaley, **({"data"
2763 is not None else {}), **kwargs)
2764
2765
2766 # Autogenerated by boilerplate.py. Do not edit as char
2767 @docstring.copy(Axes.plot_date)
2768 def plot_date(
2769 x, y, fmt='o', tz=None, xdate=True, ydate=False
2770 data=None, **kwargs):

Steps to Replicate

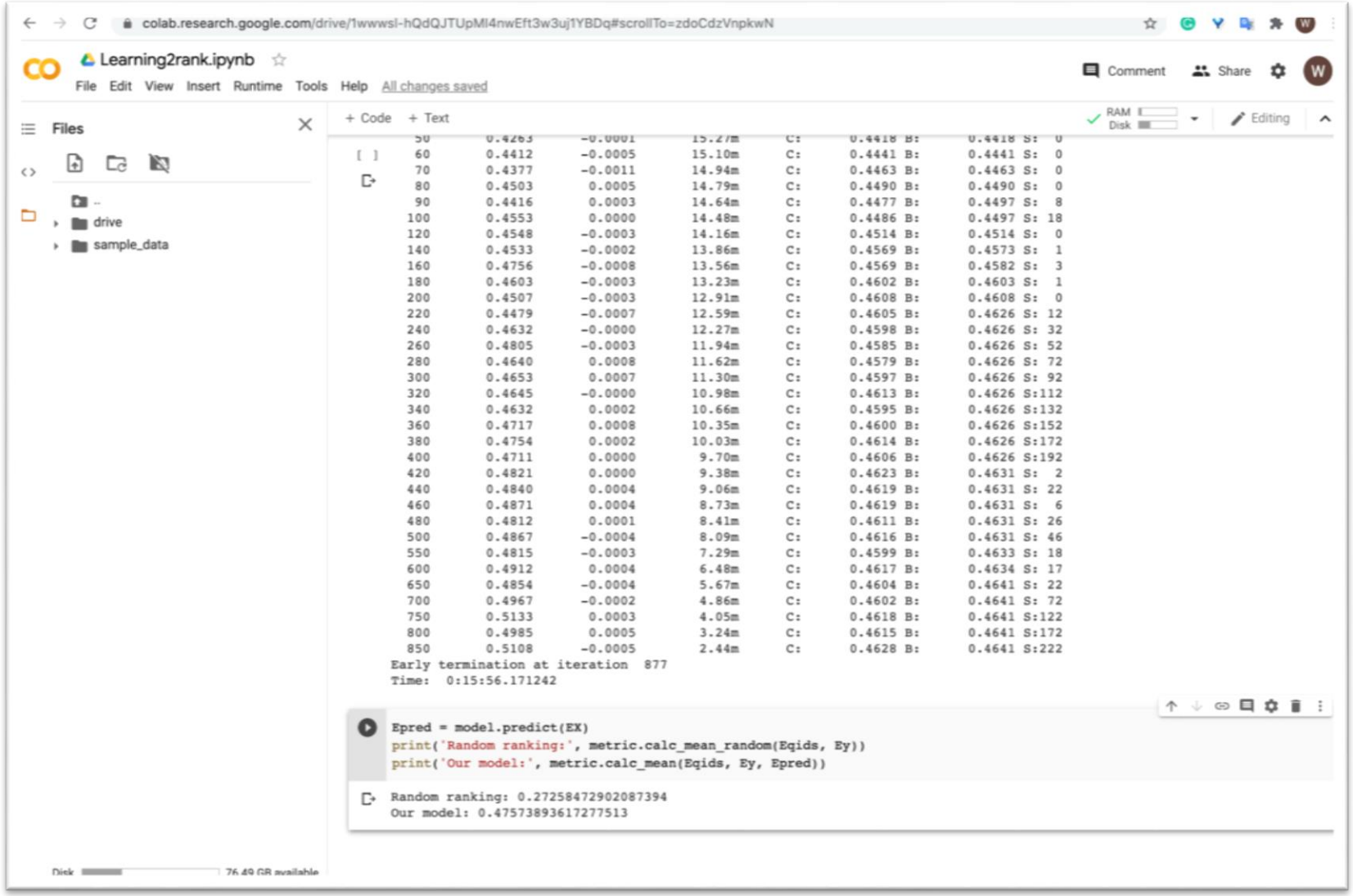
Steps to replicate

1. Read and understand the paper
2. Find existing solutions/tools/code
3. Test code and data (with Google Colab)
4. Complete the final code (with local develop environment)

Word2vec

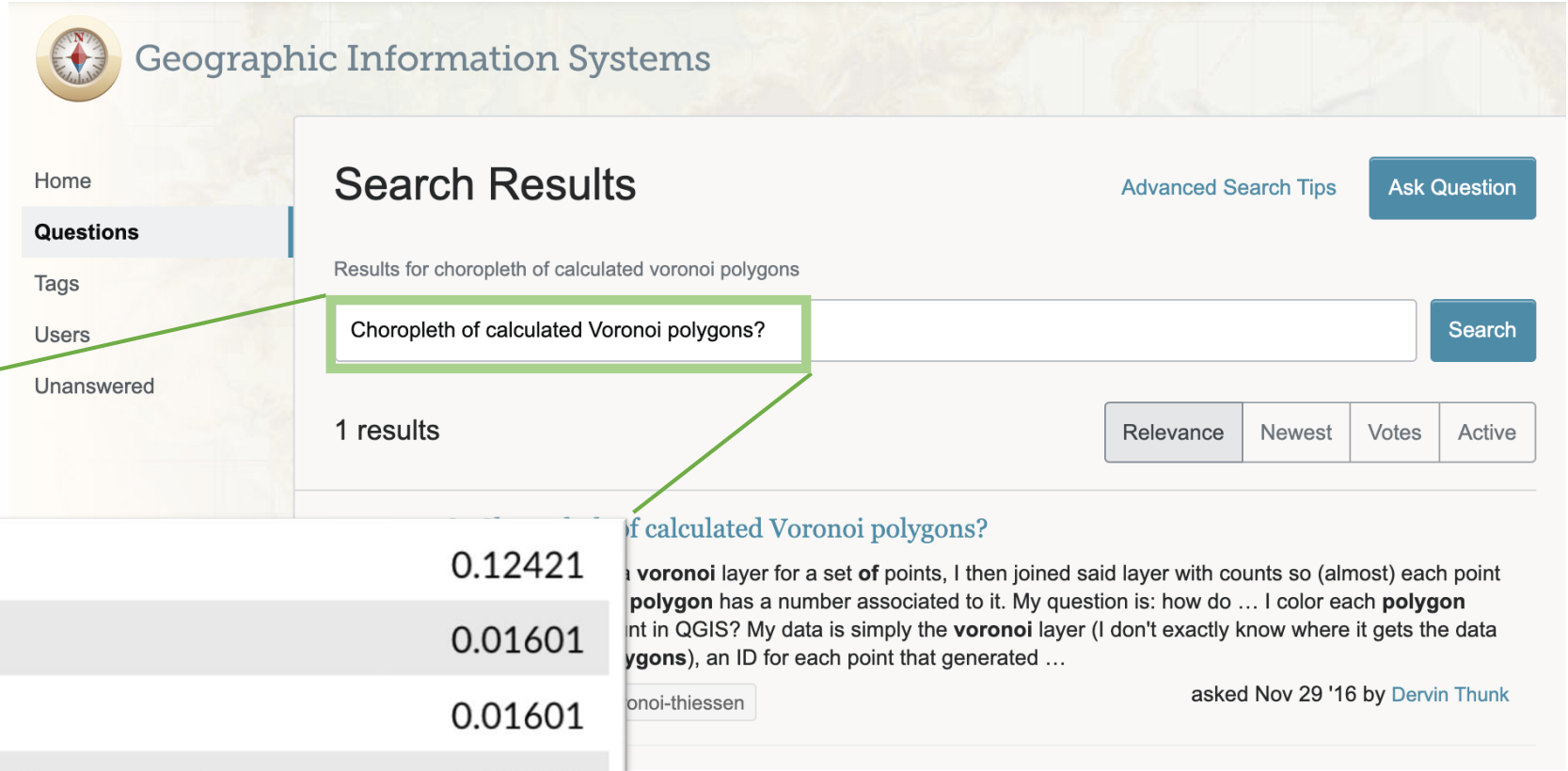

colab.research.google.com/drive/1jl6OeyHJckhOD_Ce6jMsn13PGw9k_21g#scrollTo=PhMXbJR0bqiH
word2vec.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[13] # fit a 2d PCA model to the vectors
pca = PCA(n_components=2)
result = pca.fit_transform(X)
result
array([[0.00929038, -0.00197363],
[-0.01590677, -0.00150744],
[0.00058677, -0.01314308],
[0.00584167, 0.01783884],
[-0.00851186, -0.00096154],
[0.00634455, 0.00066351],
[-0.00235268, -0.00416262],
[0.01905495, -0.00568522],
[-0.00590728, 0.01622886],
[-0.00613476, -0.01520064],
[-0.0115986 , 0.00602291],
[0.01458832, 0.0040365],
[-0.0020497 , -0.00786882],
[-0.00324499, 0.00571234]], dtype=float32)
# create a scatter plot of the projection
pyplot.scatter(result[:, 0], result[:, 1])
words = list(model.wv.vocab)
for i, word in enumerate(words):
pyplot.annotate(word, xy=(result[i, 0], result[i, 1]))
pyplot.show()
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
−0.015 −0.010 −0.005 0.000 0.005 0.010 0.015 0.020
first
get
one
final
more
for
is
sentence
this
word2vec
second
and
the
another

Learning to rank



Future work

- A Web interface
- A Chrome extension



1	**qgis_docs_print_composer**	0.12421
2	qgis_docs_geometry_tools	0.01601
3	qgis_docs_geometry	0.01601
4	qgis_docs_clipping_and_merging_raster_layers	0.01601
5	qgis_docs_modifying_vector_layers	0.01601

Code

- Word2vec(colab)
- Learning2rank(colab)
- DNN(colab)
- https://github.com/jma127/pyltr
- https://sourceforge.net/p/lemur/code/HEAD/tree/RankLib/

Reference

- li, Jing & Sun, Aixin & Xing, Zhenchang. (2018). Learning to answer programming questions with software documentation through social context embedding. Information Sciences. 448. 10.1016/j.ins.2018.03.014.
- Mikolov, Tomas & Sutskever, Ilya & Chen, Kai & Corrado, G.s & Dean, Jeffrey. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems. 26.
- Q. Wu, C. J. Burges, K. M. Svore, J. Gao, Adapting boosting for information retrieval measures, Information Retrieval 13 (3) (2010) 254–270.
- S. Gao and M. F. Goodchild, "Asking Spatial Questions to Identify GIS Functionality," 2013 Fourth International Conference on Computing for Geospatial Research and Application, San Jose, CA, 2013, pp. 106-110, doi: 10.1109/COMGEO.2013.18.
- Severyn, Aliaksei & Moschitti, Alessandro. (2015). UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. 10.18653/v1/S15-2079.
- A. Severyn, A. Moschitti, Learning to rank short text pairs with convolutional deep neural networks, in: SIGIR, 2015, pp. 373–382.
- Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. arXiv preprint 1243 arXiv:1604.01178 (2016).