Getting to know python packages with package2vec

Exploring the space of python packages with ML

Devin de Hueck
AI Data Engineering Intern
ddehueck@redhat.com
January 25th, 2020

Red Hat

In this Presentation

1. Introduction
2. Data and Data Collection
3. Building Python Package Representations
4. Evaluating Representations
5. What comes next?

Introduction

Red Hat

The Goal

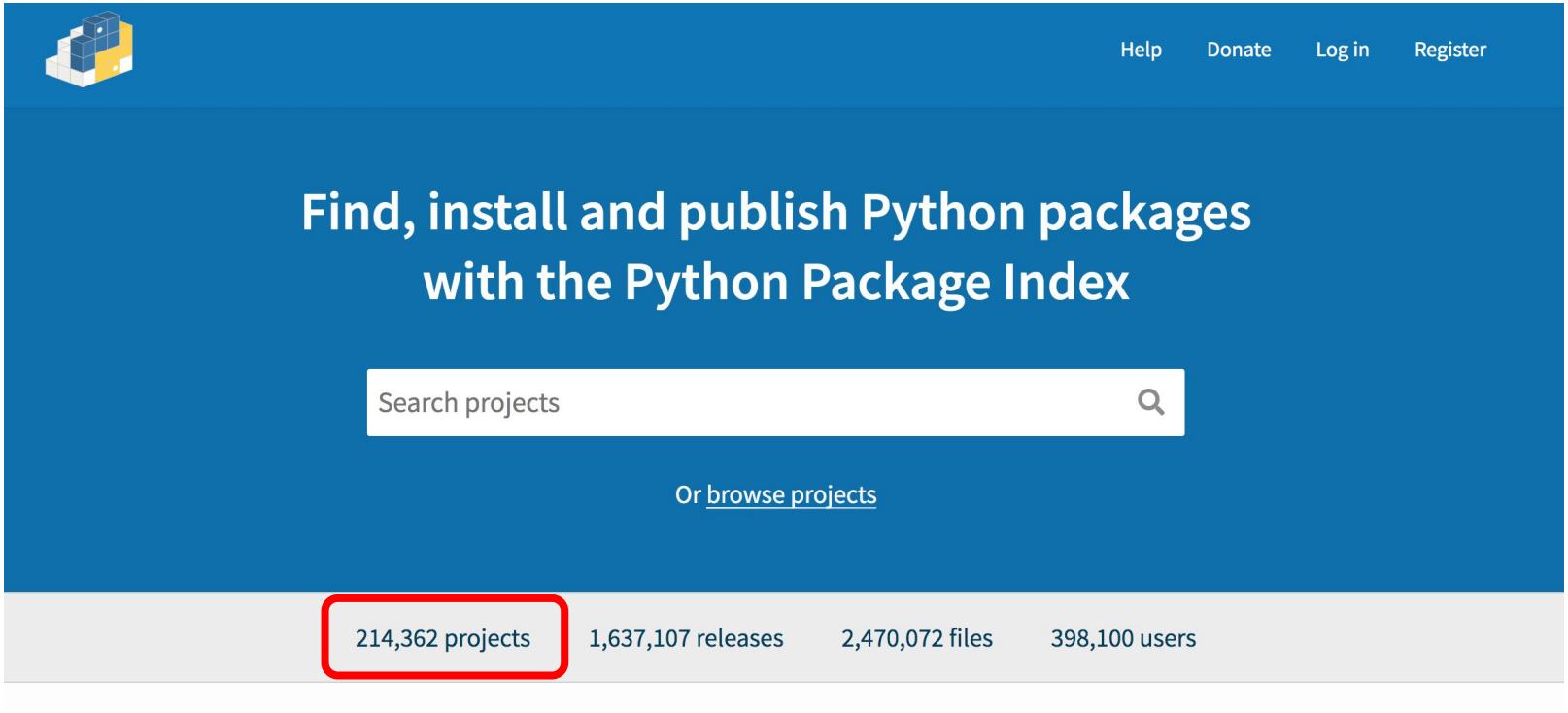
Create a python package vector space

Find packages with better features, documentation, performance, etc.

Representation Learning

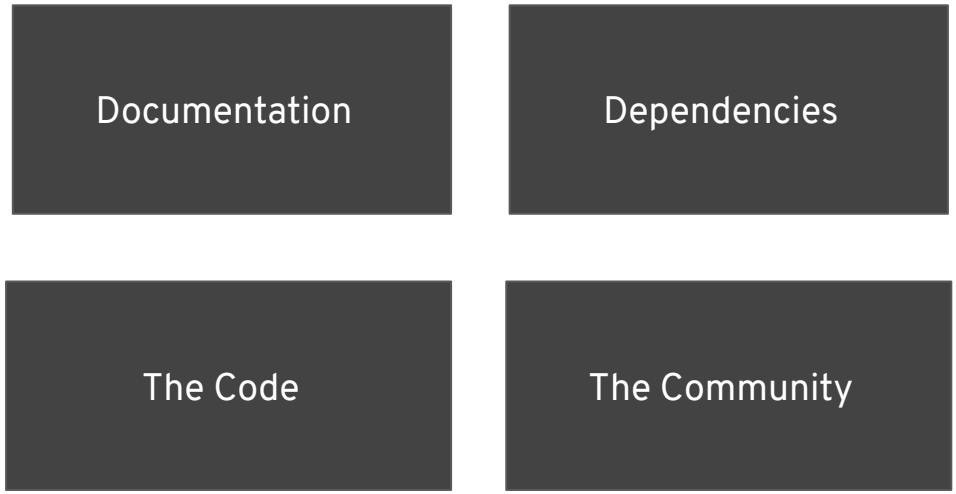
“The success of machine learning algorithms generally depends on data representation” - Bengio et al. 2012

Data and Data Collection

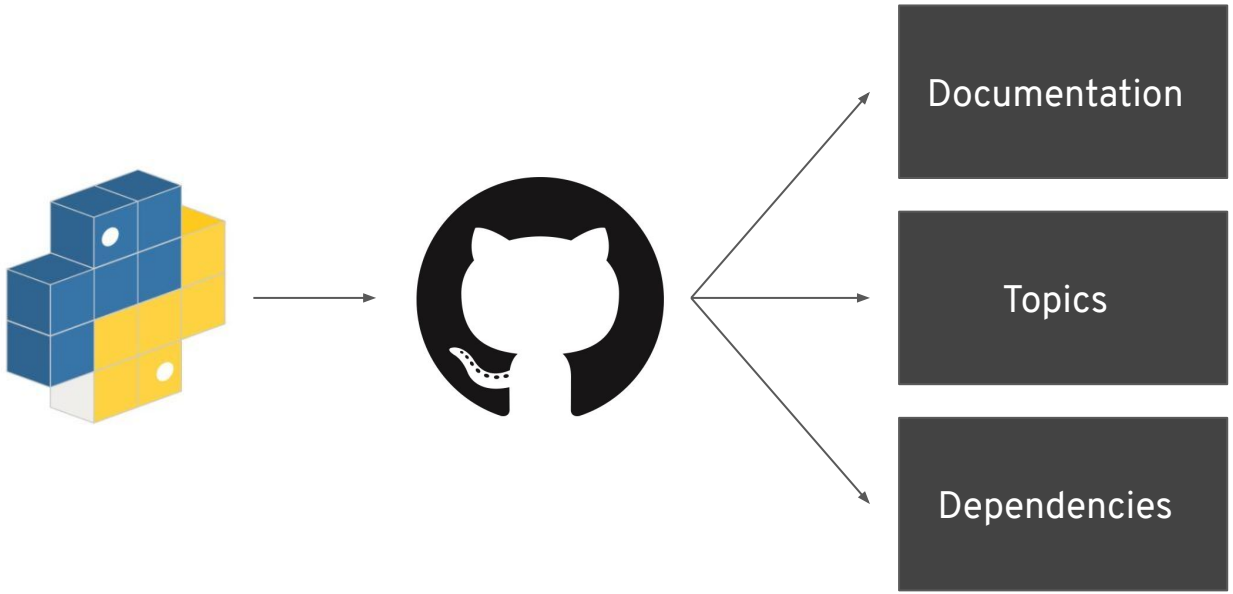

Help
Donate
Log in
Register
Find, install and publish Python packages with the Python Package Index
Search projects
Or browse projects
214,362 projects
1,637,107 releases
2,470,072 files
398,100 users

Red Hat

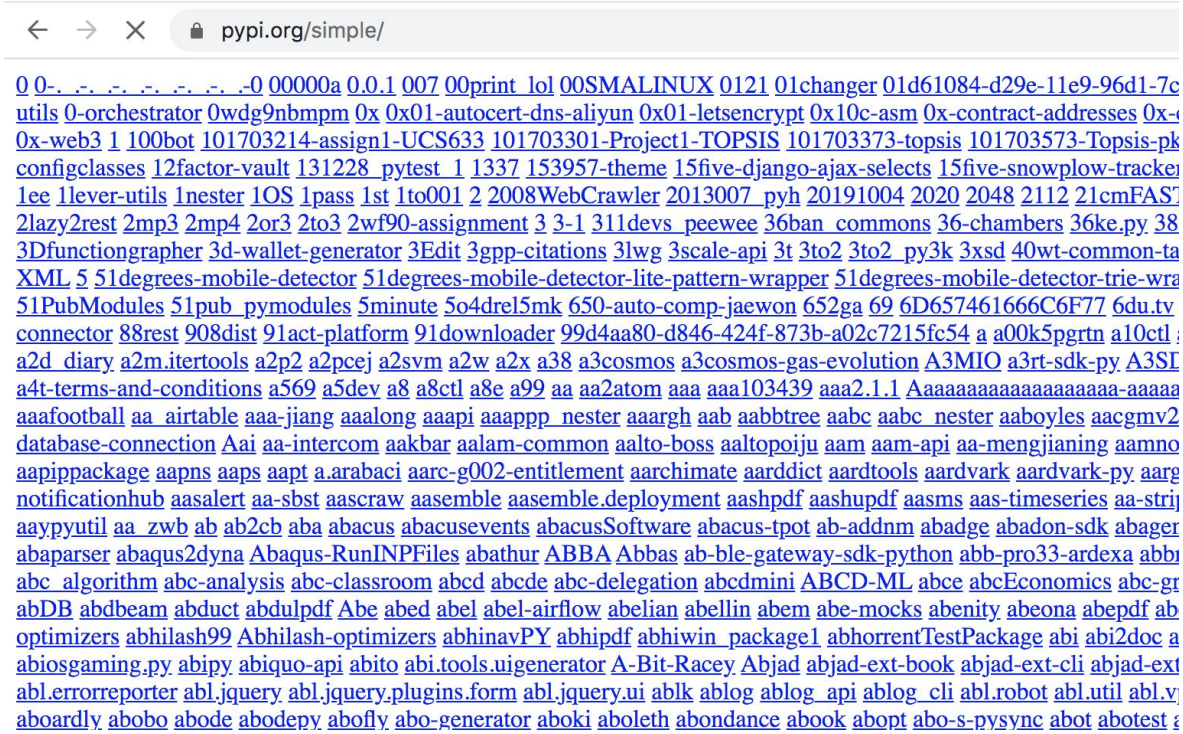
What Makes up a Python Package?



Collecting the Data

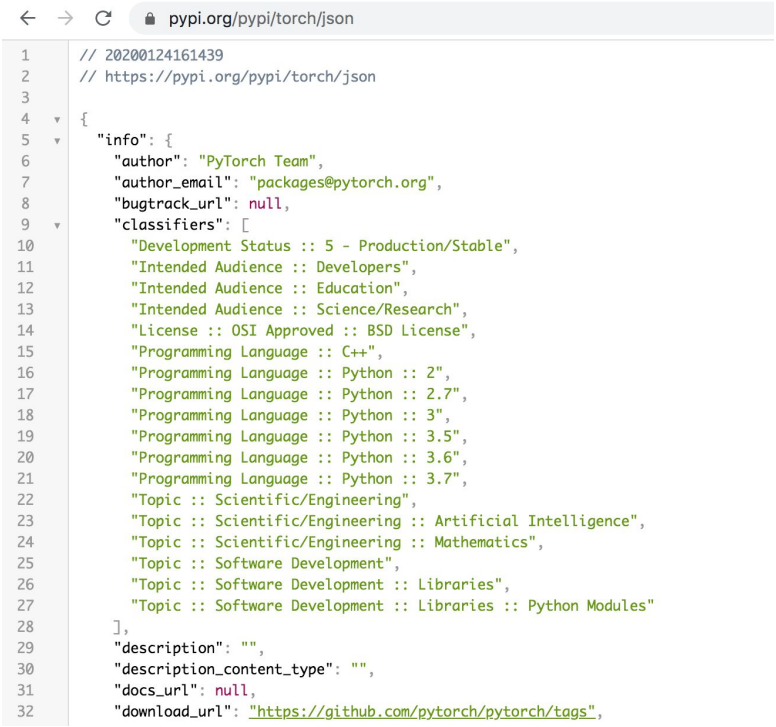


Collecting the Data



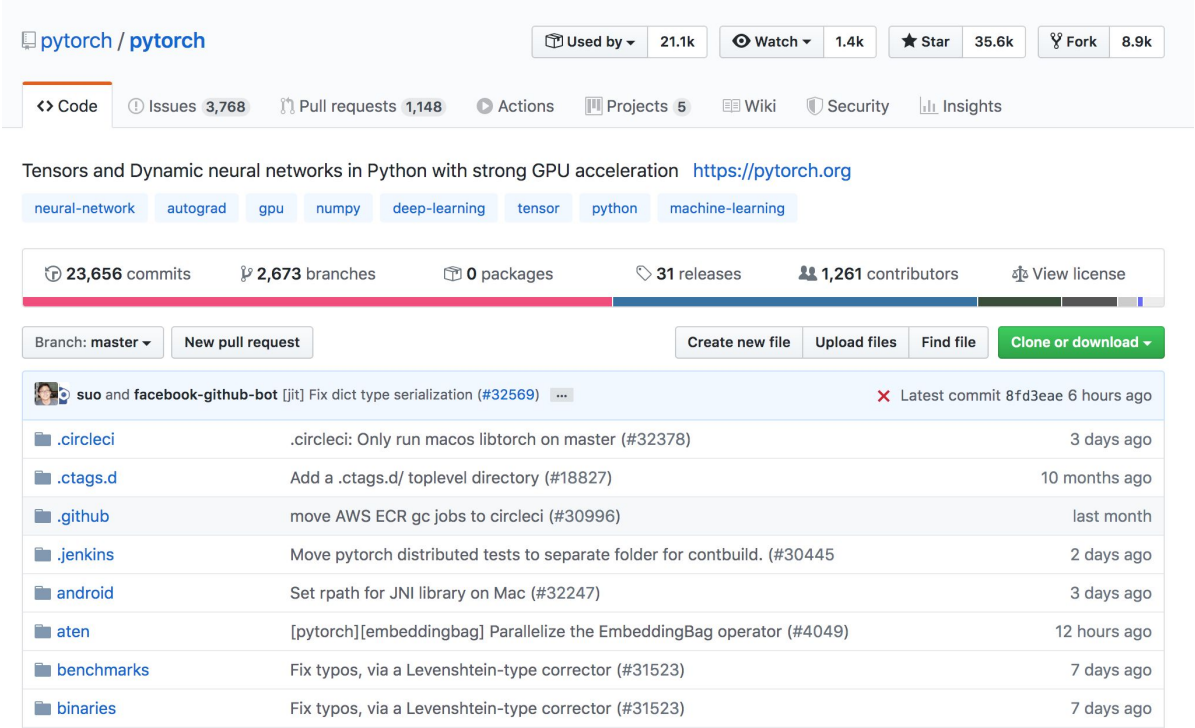
Red Hat

Collecting the Data



Red Hat

Collecting the Data



Red Hat

Collecting the Data

README.md

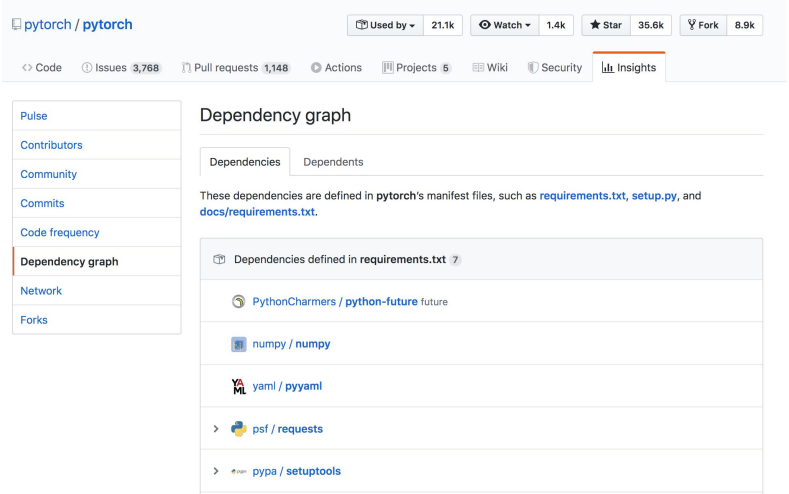
PyTorch

PyTorch is a Python package that provides two high-level features:

- Tensor computation (like NumPy) with strong GPU acceleration
- Deep neural networks built on a tape-based autograd system

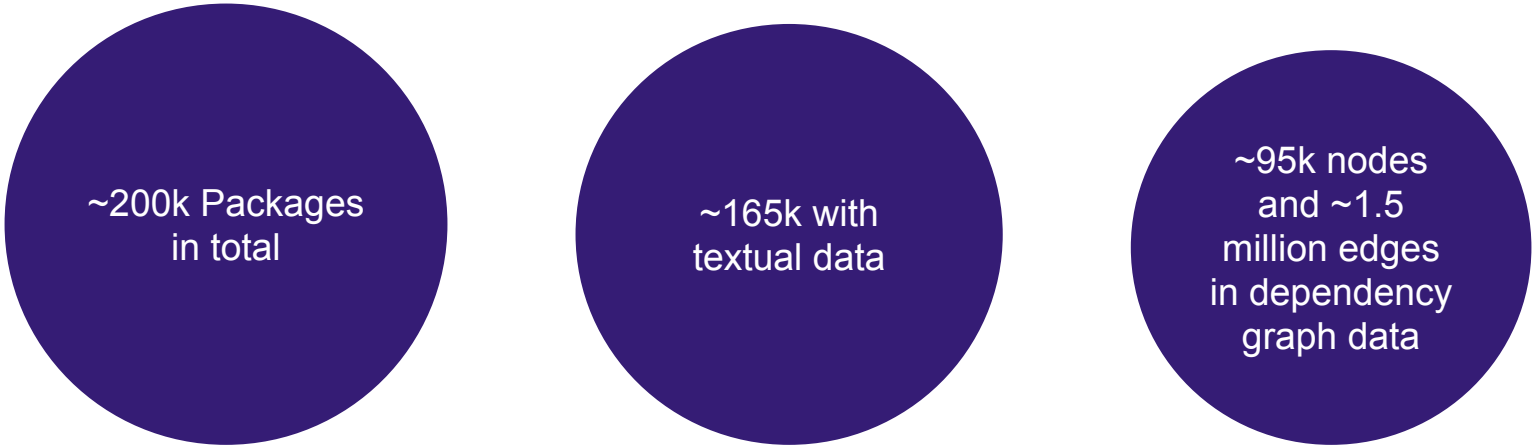
You can reuse your favorite Python packages such as NumPy, SciPy and Cython to extend PyTorch when needed.

- More about PyTorch
- Installation
  - Binaries
  - From Source
  - Docker Image
  - Building the Documentation
  - Previous Versions
- Getting Started
- Communication
- Releases and Contributing
- The Team



Red Hat

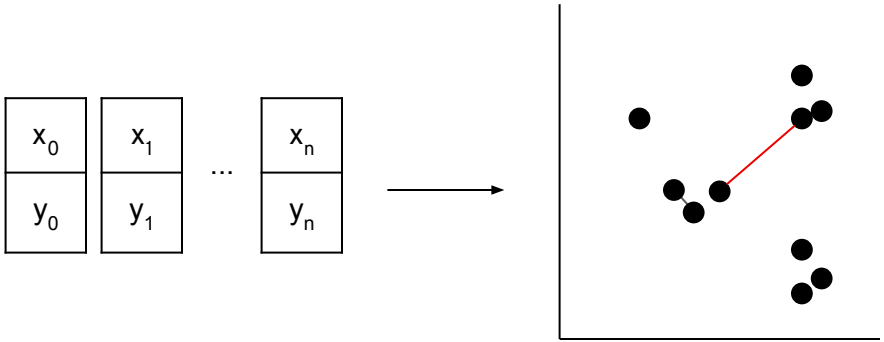
At a Glance



~90k Have both language and graph data

Developing Python Package Representations

Vector Spaces for Similarity Metrics



If we can encode python packages as vectors than we have a quantitative similarity metric

TF-IDF

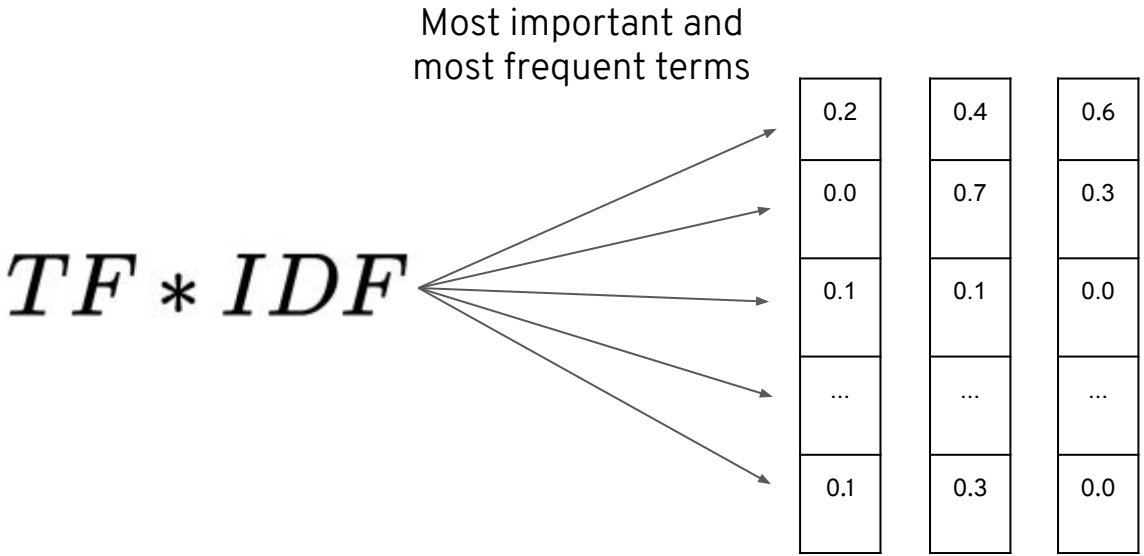
Term Frequency - Inverse Document Frequency

TF-IDF Intuition

$$TF_{i,j} = \frac{\text{Frequency of term } i \text{ in document } j}{\#\text{ Words in document } j}$$

$$IDF_i = log(\frac{\#\text{ of documents}}{\#\text{ documents with term } i})$$

TF-IDF Intuition



TF-IDF - What it tells is

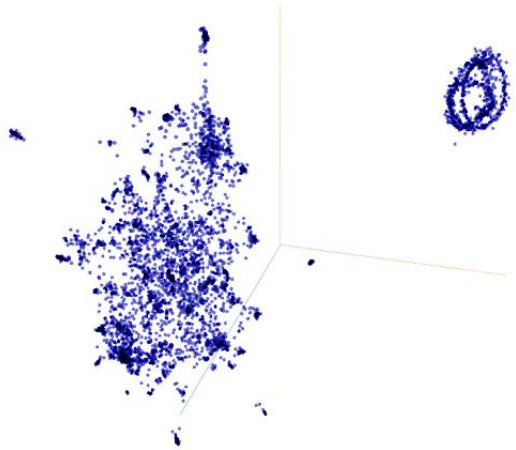
```
Top mean features across ALL documents
          feature     tfidf
0           image  0.074913
1          target  0.053084
2          django  0.046839
3         install  0.041536
4             api  0.040747
5             alt  0.040086
6   documentation  0.033638
7             add  0.032782
8            code  0.030553
9          module  0.027012
10           file  0.026988
11           pypi  0.025442
12      interface  0.024354
13        package  0.024143
14        license  0.024042
15         plugin  0.023913
16         travis  0.023182
17          datum  0.022962
18            run  0.022431
19         github  0.021816
20           test  0.020548
21        example  0.019522
22        library  0.019413
23            pip  0.017485
24            oca  0.017446
Saving Embeddings...
Saved!
```

In [7]:
```
1 feature_names = vectorizer.get_feature_names()
2 print(feature_names)
```

```
['00', '10', '2017', '2018', 'access', 'account', 'add', 'address', 'allow', 'alt', 'api', 'app', 'application', 'arg
ument', 'attribute', 'author', 'automatically', 'available', 'badge', 'base', 'bash', 'bin', 'block', 'branch', 'bu
g', 'build', 'cache', 'case', 'change', 'check', 'class', 'client', 'clone', 'code', 'column', 'command', 'config',
'configuration', 'configure', 'connection', 'contain', 'content', 'context', 'copy', 'coverage', 'create', 'current',
'custom', 'data', 'database', 'date', 'datum', 'def', 'default', 'define', 'delete', 'dependency', 'description', 'de
v', 'development', 'different', 'directory', 'display', 'django', 'docker', 'docs', 'document', 'documentation', 'dow
nload', 'easy', 'email', 'enable', 'end', 'environment', 'error', 'event', 'example', 'execute', 'exist', 'extensio
n', 'false', 'feature', 'field', 'file', 'filter', 'fix', 'folder', 'follow', 'foo', 'form', 'format', 'function', 'g
enerate', 'git', 'github', 'group', 'help', 'host', 'html', 'http', 'https', 'image', 'implement', 'implementation',
'import', 'include', 'index', 'info', 'information', 'input', 'instal', 'install', 'installation', 'instance', 'inste
ad', 'interface', 'issue', 'item', 'json', 'key', 'language', 'late', 'level', 'library', 'license', 'like', 'line',
'link', 'list', 'load', 'local', 'log', 'look', 'main', 'master', 'match', 'message', 'method', 'mode', 'model', 'mod
ule', 'multiple', 'need', 'new', 'node', 'note', 'number', 'object', 'odoo', 'open', 'option', 'optional', 'order',
'output', 'package', 'page', 'parameter', 'pass', 'password', 'path', 'pip', 'plone', 'plugin', 'png', 'point', 'por
t', 'post', 'print', 'process', 'program', 'project', 'provide', 'pypi', 'python3', 'query', 'read', 'reference', 're
lease', 'remove', 'report', 'repository', 'request', 'require', 'requirement', 'resource', 'response', 'result', 'ret
urn', 'root', 'run', 'sample', 'save', 'script', 'search', 'second', 'section', 'select', 'self', 'send', 'server',
'service', 'set', 'setting', 'setup', 'shell', 'simple', 'single', 'site', 'size', 'software', 'source', 'specific',
'specify', 'standard', 'start', 'state', 'status', 'step', 'store', 'str', 'string', 'style', 'sudo', 'support', 'sv
g', 'table', 'tag', 'target', 'task', 'td', 'template', 'test', 'text', 'time', 'title', 'token', 'tool', 'travis',
'true', 'try', 'txt', 'type', 'update', 'url', 'usage', 'use', 'user', 'value', 'variable', 'version', 'view', 'wan
t', 'way', 'web', 'work', 'write']
```

Advantages and Disadvantages of TF-IDF

- *Advantages*
  - Interpretability
  - Ease of computation
- *Disadvantages*
  - How to decide what words across the entire vocabulary of python packages?
  - Not memory efficient
  - Poor similarity metric when sparse



Doc2Vec

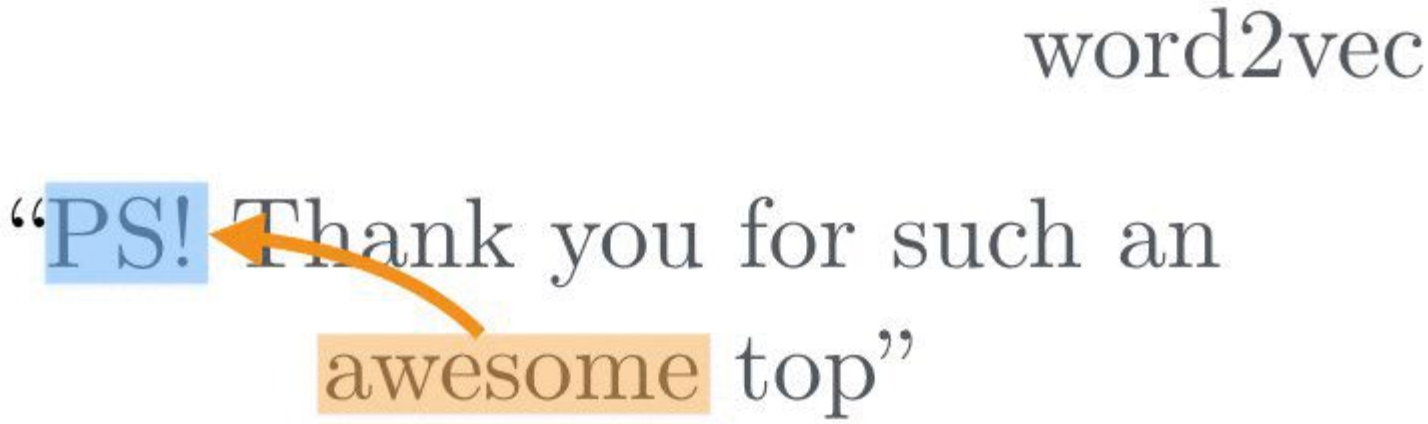
Language

Red Hat

The Distributional Hypothesis

Words occurring in similar contexts tend to be semantically similar

Word2Vec Intuition

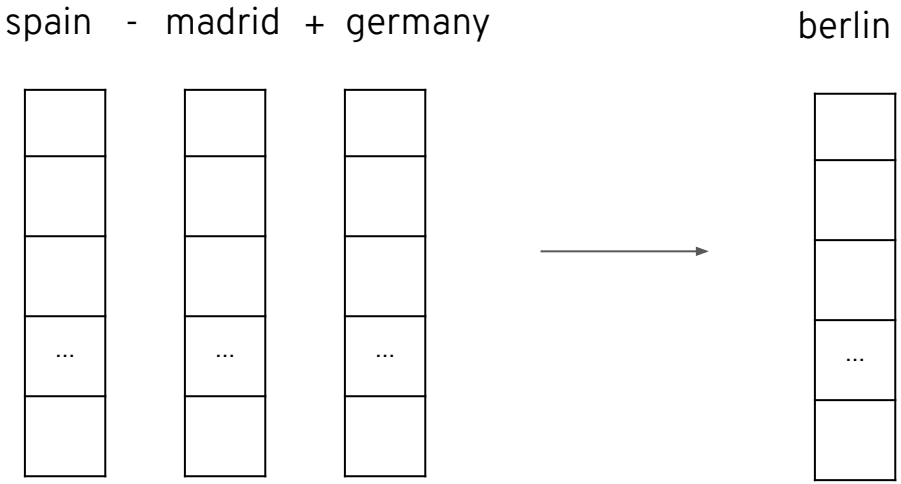


Imagine doing this over billions of sentences.

source: *https://multithreaded.stitchfix.com/blog/2016/05/27/lda2vec/*

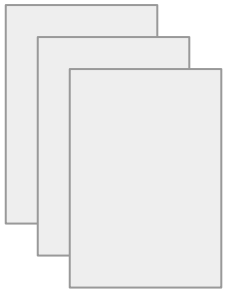
Interesting Properties of Word2Vec

What is (spain - madrid) + germany?

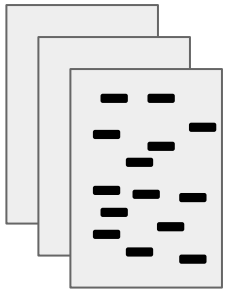


Training Examples

Raw Documents



Tokenize



Generate Windows

word2vec: **(Center, Context)**

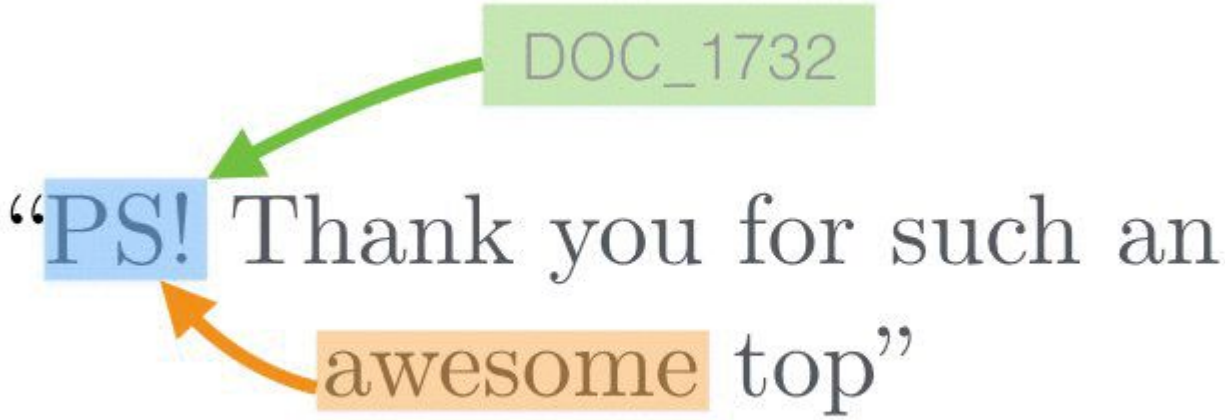
Word2Vec Negative Sampling Intuition

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$

(Center Word * Context Word) → σ → 1.0/Real

(Center Word * Fake Context Word) → σ → -1.0/False

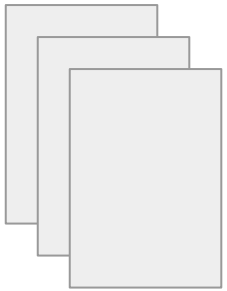
Doc2Vec



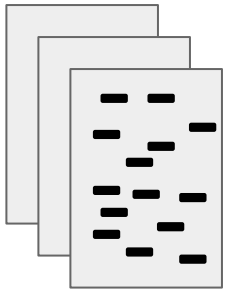
source: https://multithreaded.stitchfix.com/blog/2016/05/27/lda2vec/

Training Examples

Raw Documents



Tokenize



Generate Windows

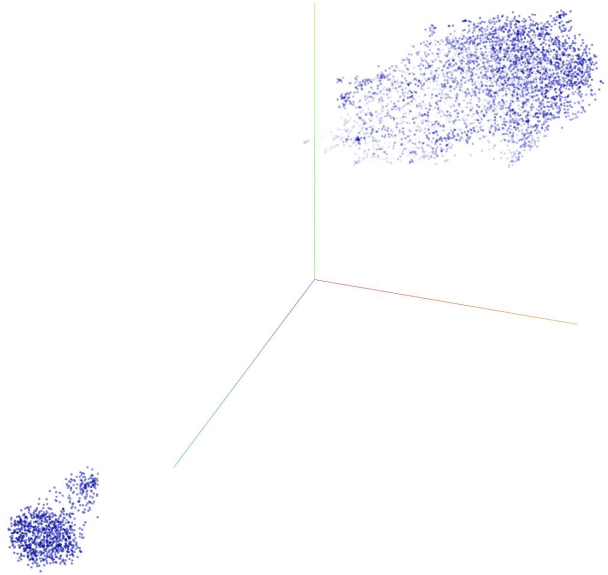
word2vec: **(Center, Context)**

doc2vec: **(Center, Document, Context)**

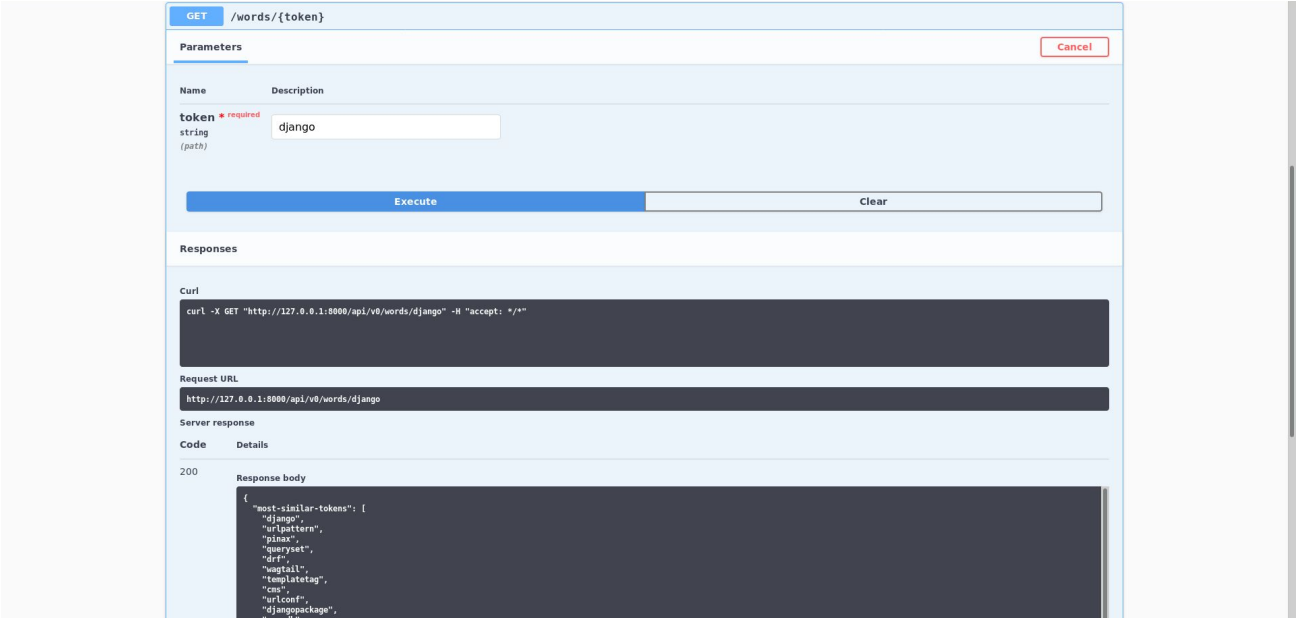
doc2vec: **(Center + Document, Context)**

Advantages and Disadvantages of Doc2Vec

- *Advantages*
  - Semantically meaningful distance metric
  - A dense representation let's us use less memory than TF-IDF vectors
- *Disadvantages*
  - A dense representation is not interpretable



Ash API Demo



Red Hat

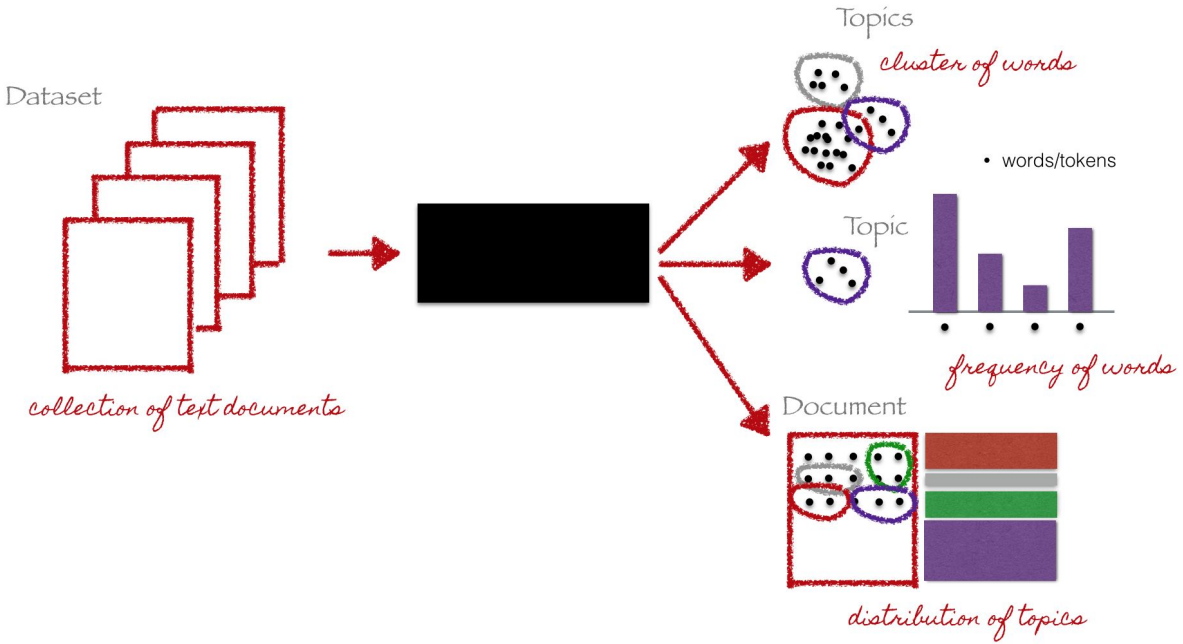
ETM

Topic Modeling in Embedding Spaces

Language

Red Hat

Latent Dirichlet Allocation



source: http://chdoig.github.io/pygotham-topic-modeling/#/3/1

Latent Dirichlet Allocation

Doesn't work well with a massive vocabulary

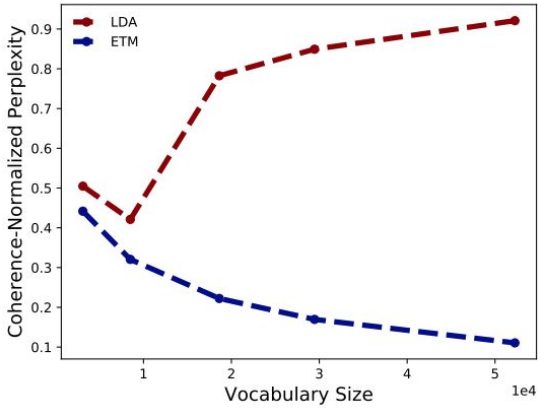


Figure 1. Ratio of the normalized held-out perplexity for document completion and the topic coherence as as a function of the vocabulary size for the ETM and LDA. While the performance of LDA deteriorates for large vocabularies, the ETM maintains good performance.

(2019, Dieng et al.)

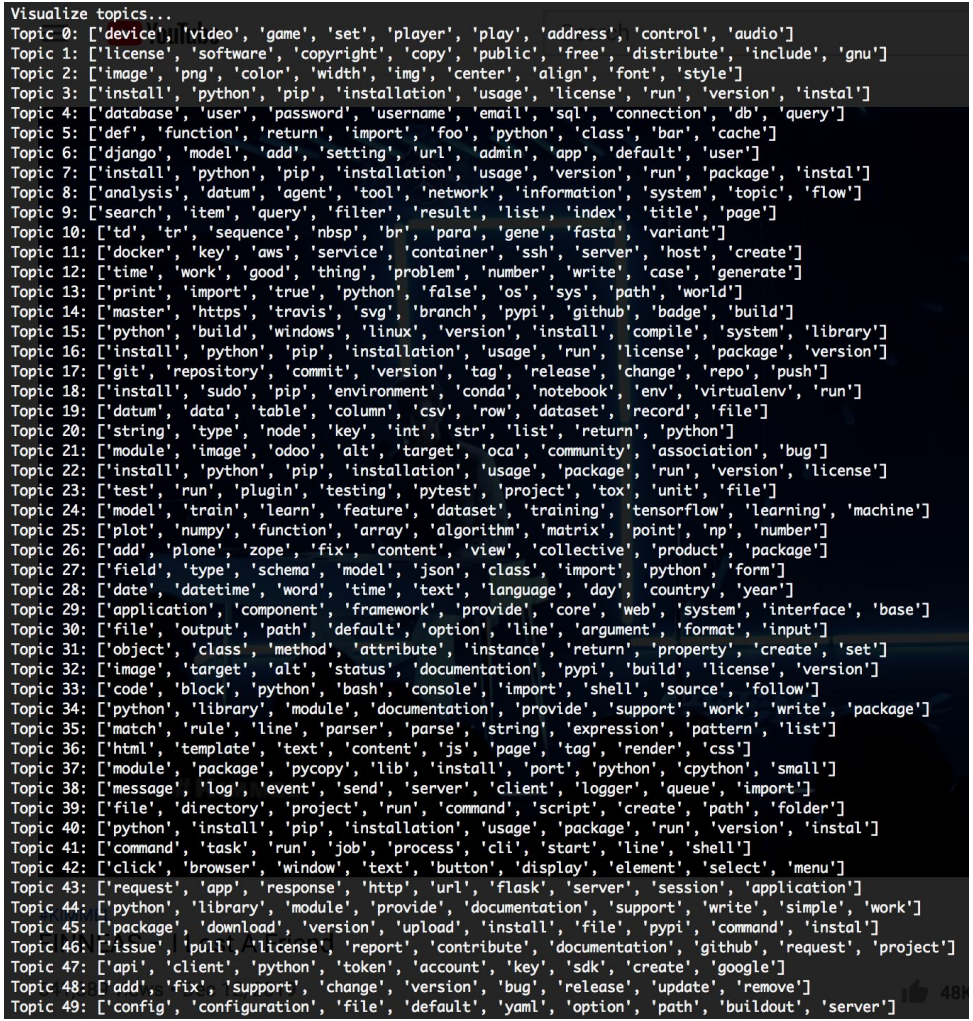
“Games”

“Cloud Computing”

“Data”

“Machine Learning”

“Web Applications”



Red Hat

“Games” Topic	“Cloud Computing” Topic	“Machine Learning” Topic	“Web Applications” Topic
Device	Docker	Model	Application
Video	Key	Train	Component
Game	AWS	Learn	Framework
Set	Service	Feature	Provide
Player	Container	Dataset	Core
Play	SSH	Training	Web
Address	Server	Tensorflow	System
Control	Host	Learning	Interfase
Audio	Create	Machine	Base

Advantages and Disadvantages of ETM

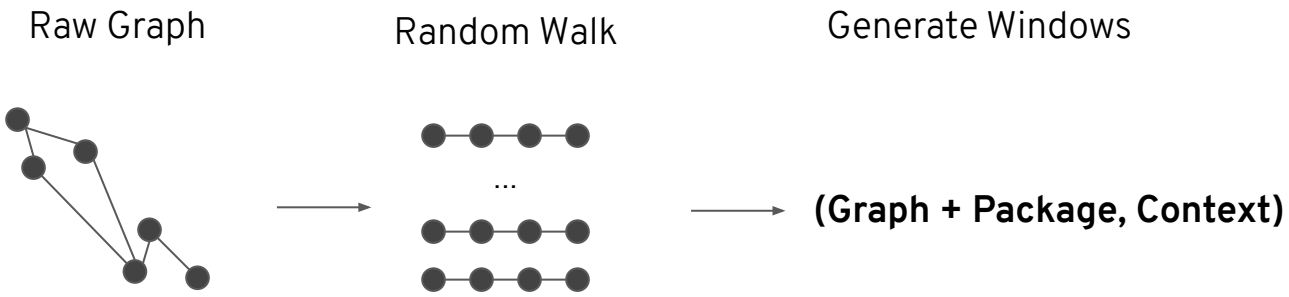
- *Advantages*
  - Learns accurate word embeddings alongside topics
  - Topics allow for some interpretability
  - Beats out LDA in large vocabulary spaces
- *Disadvantages*
  - Doesn't directly learn document embeddings*
  - Have to set number of topics

DeepWalk

Graphical

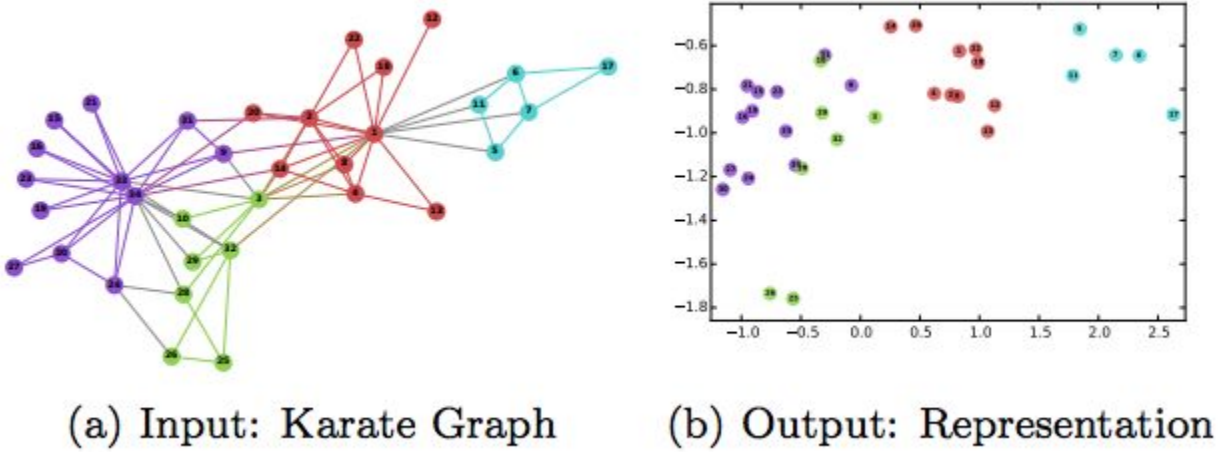
Red Hat

Training Examples



A random walk through “the language graph” is a sentence.

Community Graph Structures

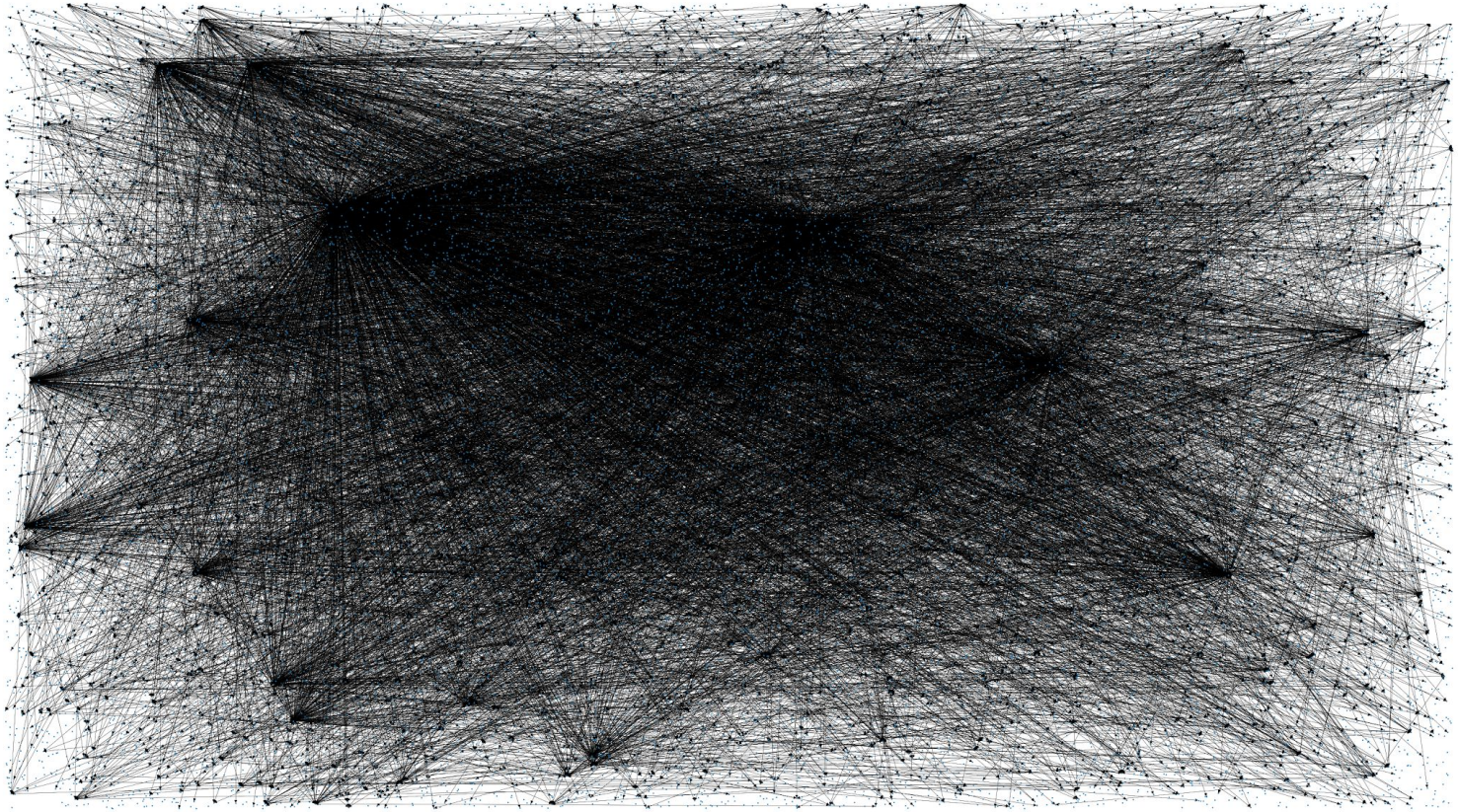


(a) Input: Karate Graph (b) Output: Representation

Perozzi et al. 2014

What about PyPI data?

Red Hat

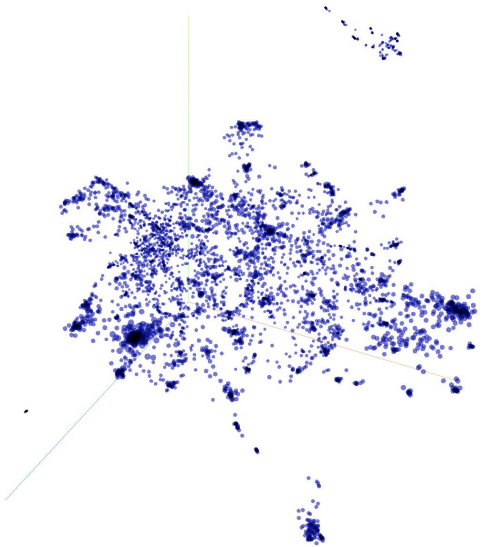


This is 20k packages

Red Hat

Advantages and Disadvantages of DeepWalk

- *Advantages*
  - Opens up a new domain of data!
  - Emphasizes community/social structure
  - **Generalizes language to graph structure**
- *Disadvantages*
  - Not really interpretable



A Joint Approach

CrossWalk - Walking across data domains

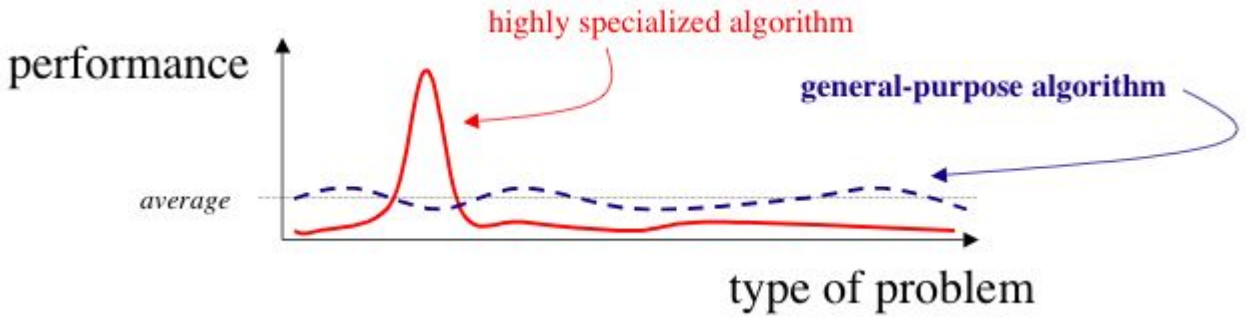
Red Hat

A Joint Approach

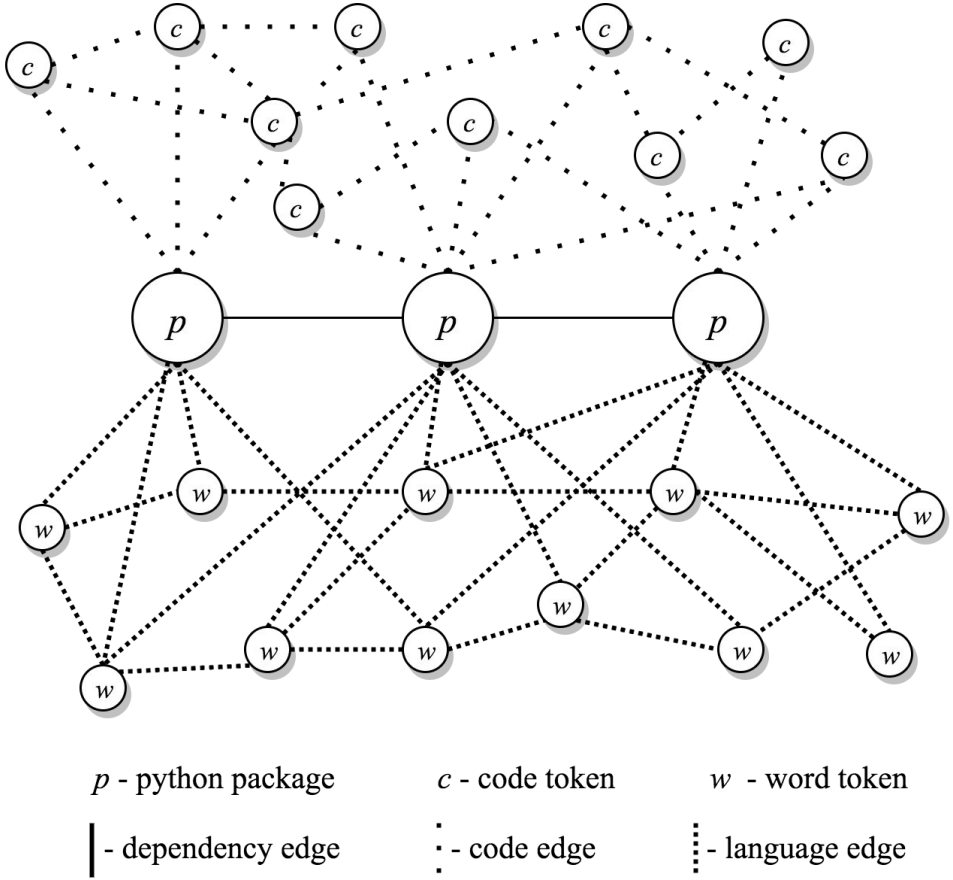
Why not use all data sources in one model?

How to interpret our data?

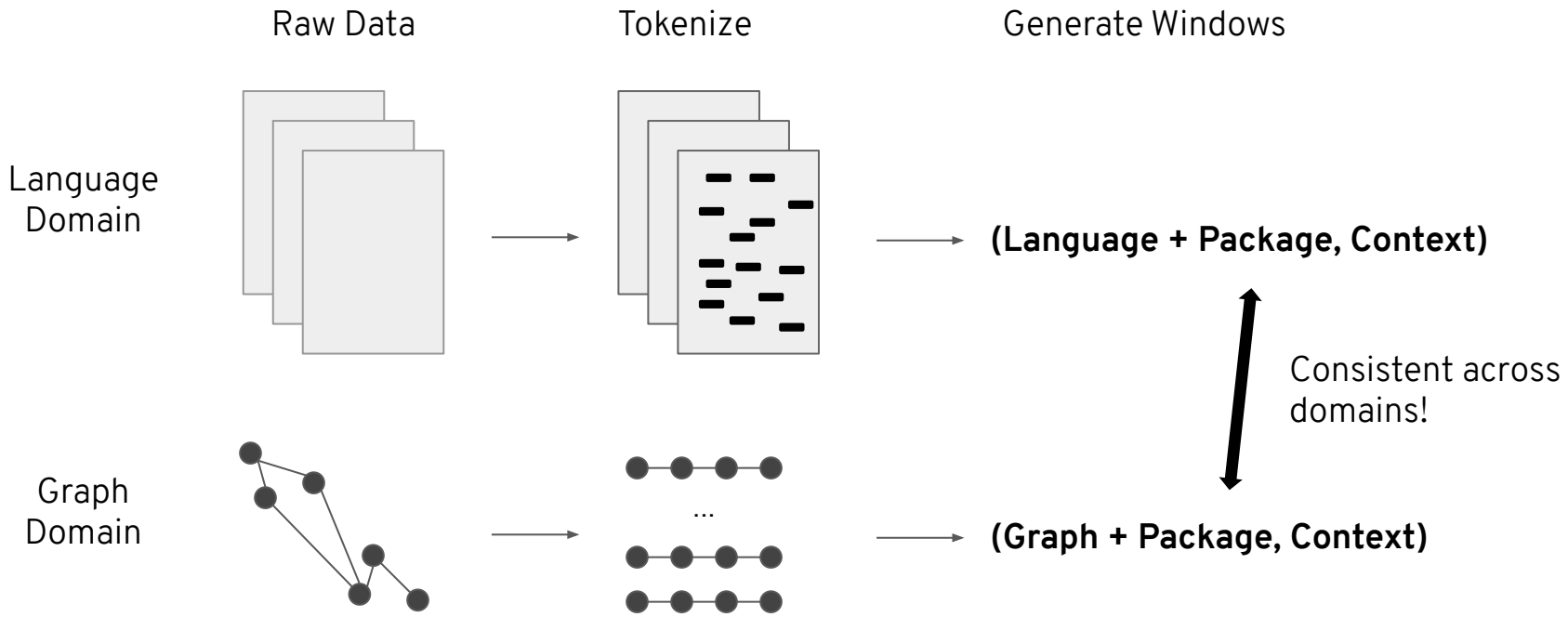
A Joint Approach



source: https://multithreaded.stitchfix.com/blog/2016/05/27/lda2vec/


p - python package
c - code token
w - word token
- dependency edge
- code edge
- language edge

Training Examples



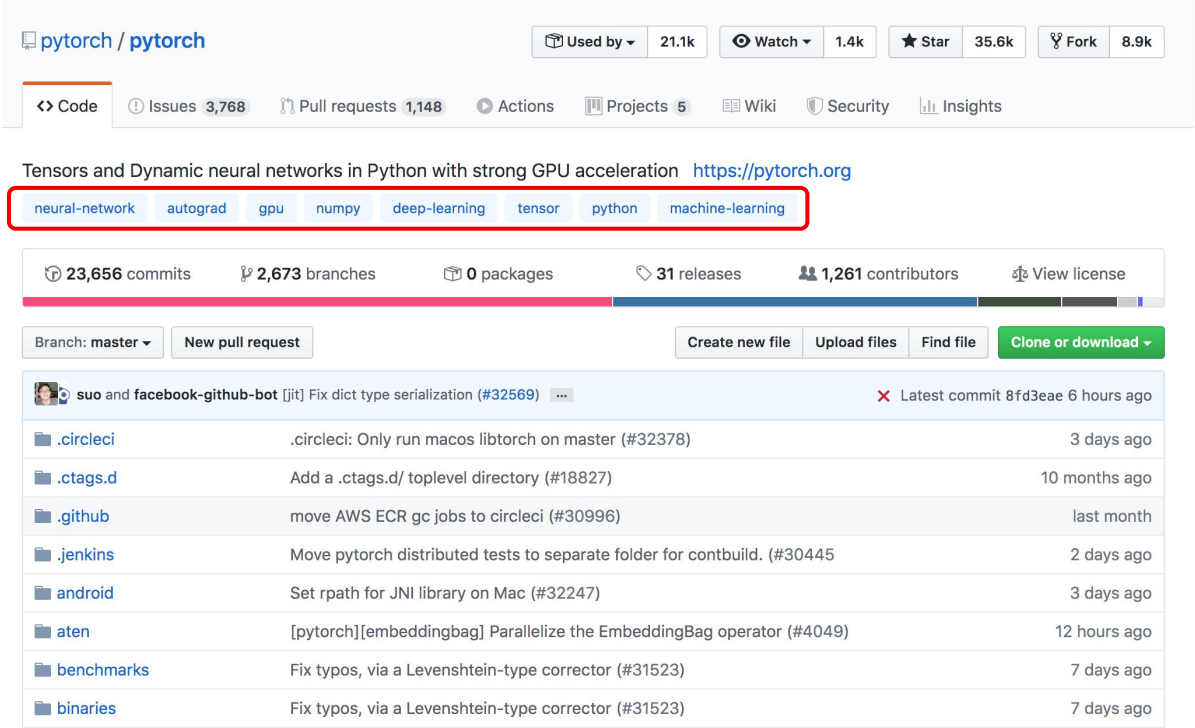
Red Hat

Evaluating Python Package Vectors

How to Evaluate?

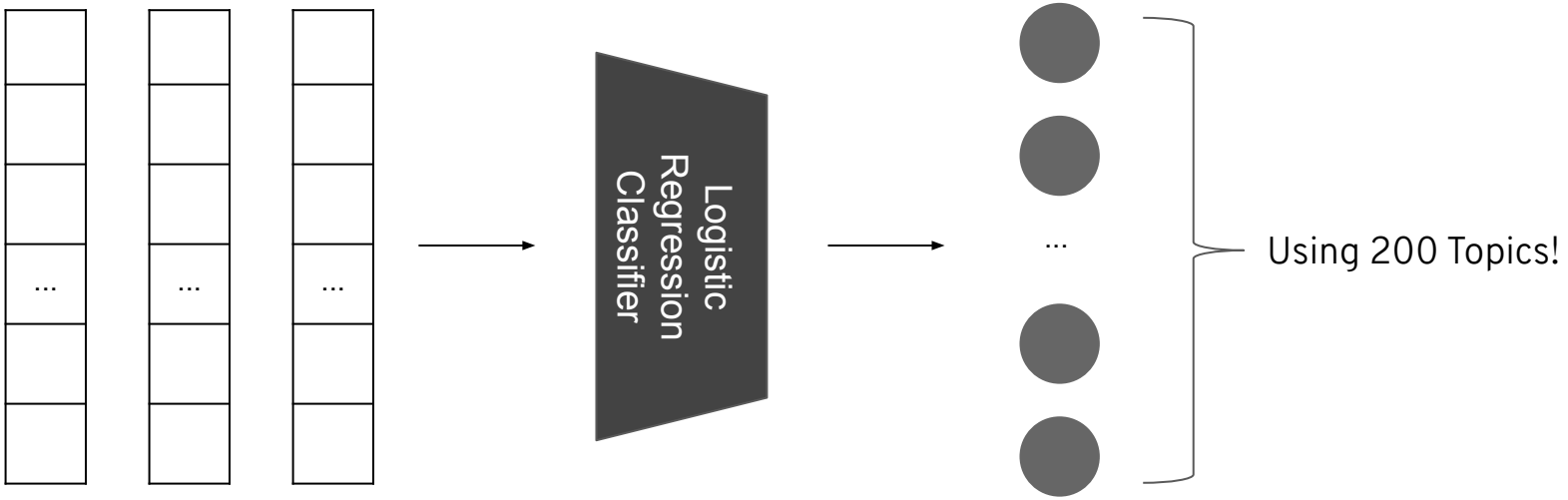
By the discriminatory ability in the embeddings, i.e. The easier it is to classify the better the representation!

Evaluating with GitHub Topics

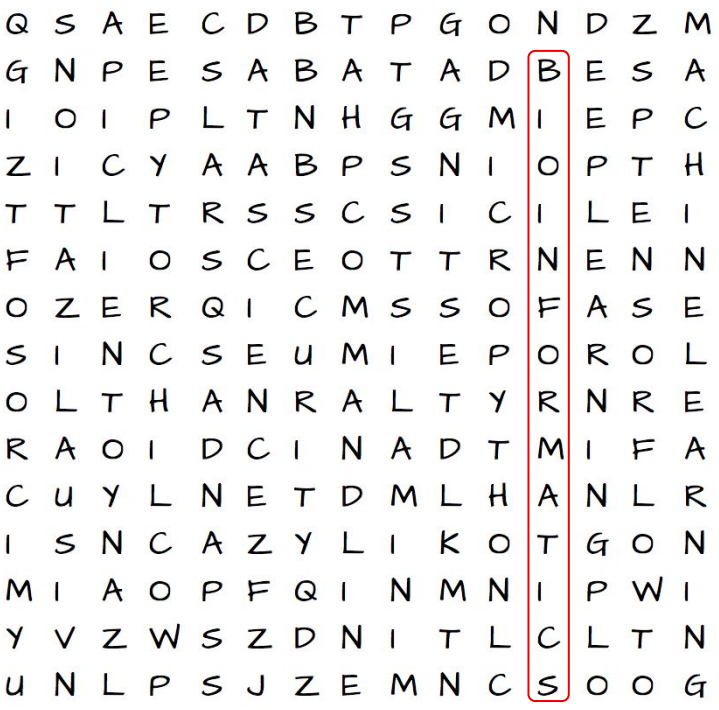


Red Hat

Evaluating with Logistic Regression



Results



MACHINELEARNING
CLI
MICROPYTHON
DEEPLEARNING
AWS
MINIMALIST
NLP
JSON
DATASCIENCE
TENSORFLOW
PYTORCH
BIOINFORMATICS
MICROSOFT
TESTING
COMMANDLINE
PANDAS
DATABASE
SECURITY
VISUALIZATION
APICLIENT

Red Hat

Results

% Labeled	*10%*	*50%*	*90%*
TF-IDF	30.55%	36.84%	38.40%
Doc2Vec	31.82%	**41.03%**	**43.33%**
DeepWalk	**34.6%**	39.11%	40.53%

In Conclusion

- We built a PyPI dataset - partially labeled
- Tried different embedding techniques and learned along the way
- Developing a model to put it all together
- More to do!

References

1. Bengio, Y., A. Courville, and P. Vincent. "Representation Learning: A Review and New Perspectives. ArXiv e-prints." arXiv preprint arXiv:1206.5538 (2012).
2. Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).
3. Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.
4. Dieng, Adji B., Francisco JR Ruiz, and David M. Blei. "Topic modeling in embedding spaces." arXiv preprint arXiv:1907.04907 (2019).