



similar repos search  
**motivation.**

FIND, COMPARE & EXPLORE

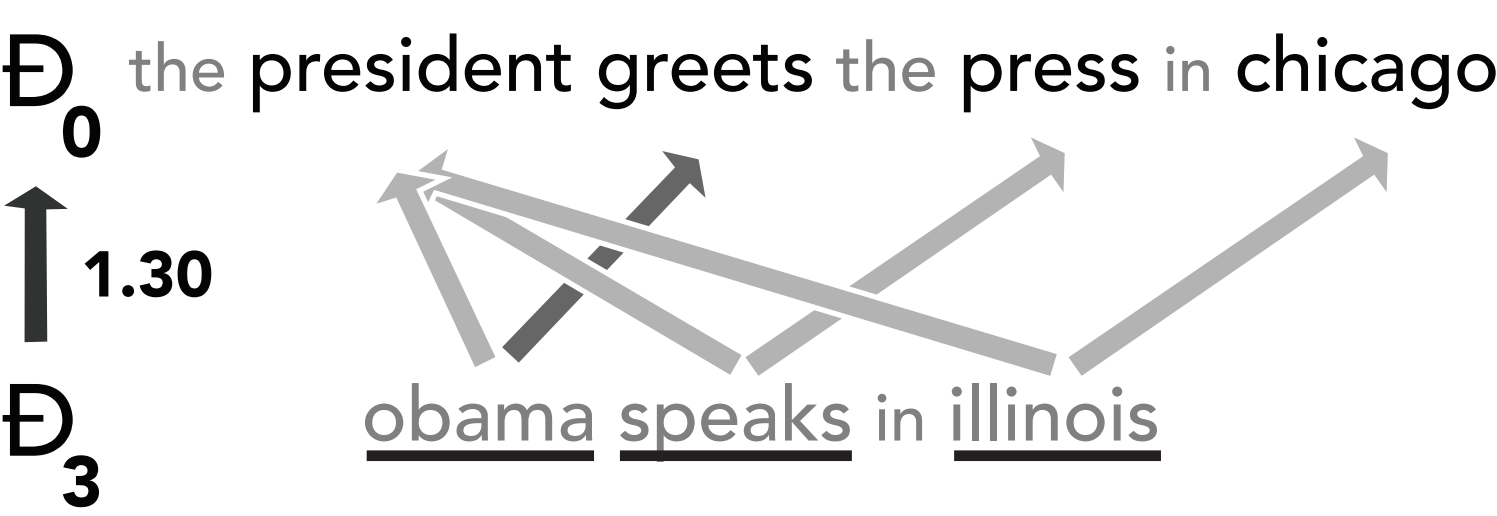
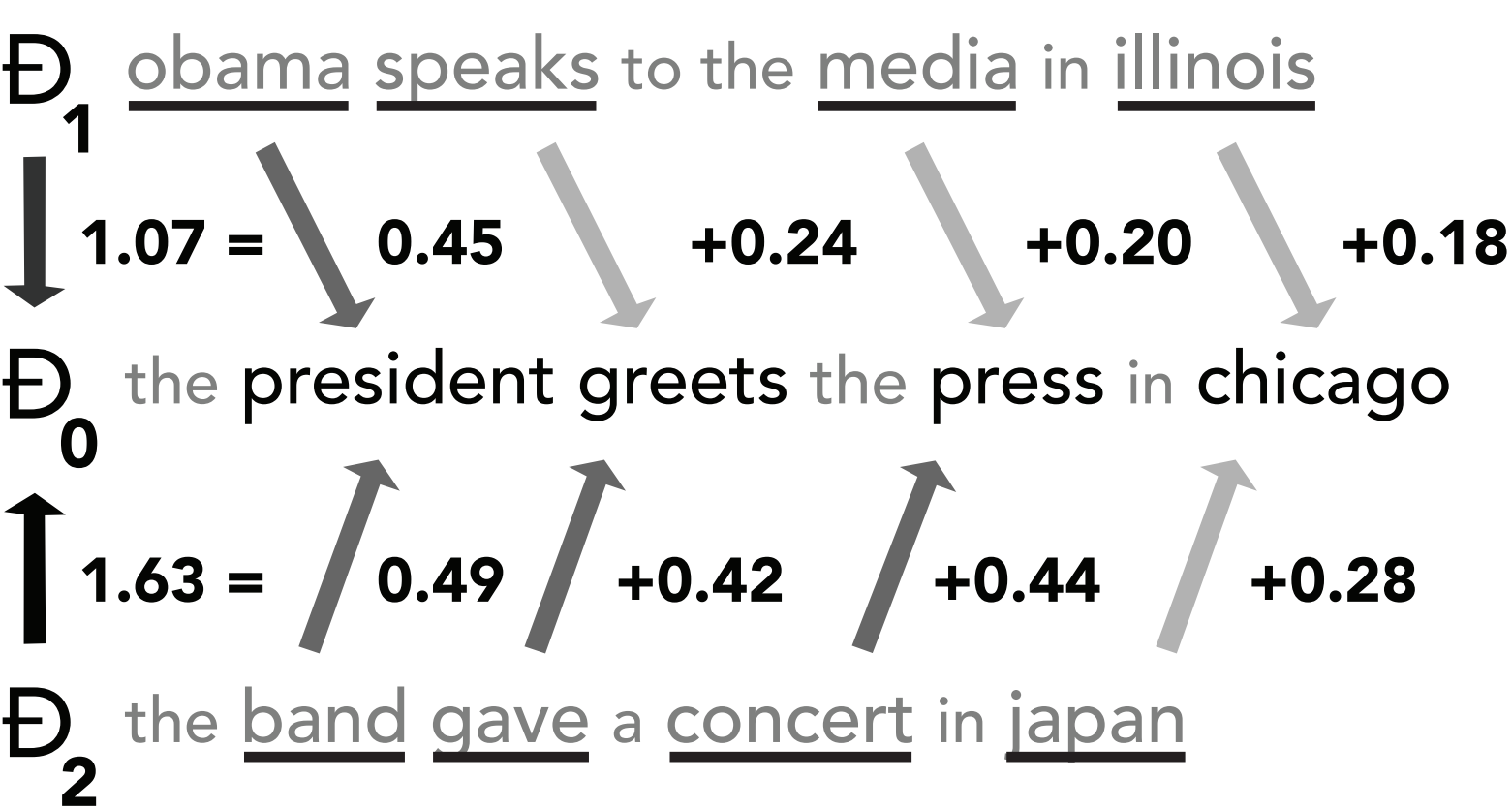
- ‡ repositories that better fit needs
- ‡ new ideas, inspiration
- ‡ people with similar ideas
- ‡ compare libraries with the same purpose
- ‡ explore new areas of knowledge

**how-to.**

PREPARATION

- ‡ clone
- ‡ parse source code
- ‡ tokenization
- ‡ prepare Swivel input & TF-IDF
- ‡ train Swivel embeddings
- ‡ repo weighted nbow

SEARCH ALGO



- ‡ centroid distance
- ‡ relaxed LP  
(which gives an upper boundary)
- ‡ WMD only for best options

This allows to avoid 95% WMD evaluations

**usage.**

```
>>> import vecino
>>> engine = vecino.SimilarRepositories()
>>> root = "https://github.com/"
>>> repo = root + "tensorflow/tensorflow"
>>> print(engine.query(repo))
```

topic modeling  
**motivation.**

WHY

- ‡ programmers use natural language in source code
- ‡ names are a rich source of information
- ‡ names help to understand project at high level
- ‡ names help to summarize content
- ‡ group projects that deal with the same topic

**how-to.**

PIPELINE

- ‡ repositories
- ‡ deduplicate repos (src-d/minhashcuda)
- ‡ select source code files only (src-d/enry)
- ‡ extract names (doc.bblf.sh)
- ‡ name processing (tokenization)
- ‡ filtering by DF
- ‡ bag-of-words per repository
- ‡ additive regularization of topic models (bigARTM)
- ‡ manual topic labeling

SHORT MATH

$$p(w|d) = \sum_{t \in T} p(w|t)p(t|d)$$

where w - term, d - document, t - topic

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_t \phi_{wt} \theta_{td} + R(\Phi, \Theta) \rightarrow \max_{\Phi, \Theta}$$

the idea of ARTM - MLE + regularization R

$$R(\Phi, \Theta)_{Dirichlet} = \sum_{t,w} (\beta_w - 1) \ln \phi_{wt} + \sum_{d,t} (\alpha_t - 1) \ln \theta_{t,d}$$

LDA in terms of ARTM

**usage.**

```
# docker run srcd/github_topics apache/spark
```

- ‡ 18.52 spark
- ‡ 16.58 hadoop
- ‡ 13.86 java web servers
- ‡ 13.03 matplotlib; python machine learning
- ‡ 12.13 transport, gps

These numbers express the relative importance of a given topic for this repository

future  
**motivation.**

WHY ML ON SOURCE CODE

- ‡ source code has strong patterns
- ‡ software is everywhere → tons of data
- ‡ complexity of source code is increasing
- ‡ transfer code patterns from best projects
- ‡ ~50M people who know how to program

ML APPLIED ON SOURCE CODE

- ‡ refactoring, suggestion, snippet search, ...  
(coding assistance)
- ‡ automatic test generation, code review
- ‡ bug detection and fixing
- ‡ source code generation in narrow area  
(like pixel2code)

SUGGESTIONS NOW

from django.db i

```
> if if
> idea idea
> im import
> ifmain if __name__ == '__main__':
> pdb iPDB set trace
```

FUTURE SUGGESTIONS

from django.db

```
import models
from django.utils.encoding import \
    python_2_unicode_compatible

@python_2_unicode_compatible
class test_from_model(models.Model):
    title = models.CharField(max_length=...)
```

**tree-based nn.**

MOTIVATION

- ‡ code has structure - AST
- ‡ AST differs from human lang structure
- ‡ AST contains rich information about code
- ‡ distance between items is shorter in tree
- ‡ AST will provide better context for items
- ‡ NN on tree will find great patterns in code

**bblfsh.**

USAGE

```
>>> python3 -m bblfsh -f source_file
```

MOTIVATION

- ‡ programming languages have AST
- ‡ leverage standard parser from each language
- ‡ normalize as a post-processing step: AST → UAST
- ‡ result: same UAST for different languages

HOW-TO

- ‡ lang drivers are packaged as docker containers
- ‡ server contains a lightweight container runtime
- ‡ no docker, no external runtime dependencies
- ‡ official drivers published at docker hub