Exploratory code search and snippet suggestion

Article review

Slavnov Konstantin
konstantin@sourced.tech

July 13, 2017

Introduction

What is the **fastest** way to learn a new library?

New framework investigation ways:

- Documentation reading;
- Ask stackoverflow;
- Just start to use it;
- Search code examples;
- etc

Introduction

What is the **fastest** way to learn a new library?

New framework investigation ways:

- Documentation reading;
- Ask stackoverflow;
- Just start to use it;
- Search code examples;
- etc

Insights from Searching and Skimming: An Exploratory Study [5].

Ways to solve

Let’s build a machine learning assistant!

Approaches:

- Topic modeling
- Hierarchical clustering
- Deep learning way
- Probabilistic way

Ways to solve

Let's build a machine learning assistant!

Approaches:

- Topic modeling
- Hierarchical clustering
- Deep learning way
- Probabilistic way

Topic modeling

Scheme

- Get a codebase of library usage
- Build a hierarchical topic modeling for codebase
- Show it for user API query
- ???
- PROFIT!

Flat topic model. Reminder

- Documents $d \in D$
- Tokens (words) $w \in W$
- Topics $t \in T$
- Document-token counters n_{dw}

Flat topic model:

$$P_{wd} = \frac{n_{dw}}{\sum_{w' \in W} n_{dw'}} = p(w \mid d) \approx \sum_{t \in T} p(w \mid t)\, p(t \mid d) = \sum_{tin} \phi_{wt} \theta_{td} = \{\Phi\Theta\}_{wd}$$

or just

$$P \approx \Phi\Theta$$

Flat topic model. Reminder

- Documents $d \in D$
- Tokens (words) $w \in W$
- Topics $t \in T$
- Document-token counters n_{dw}

Flat topic model:

$$P_{wd} = \frac{n_{dw}}{\sum_{w' \in W} n_{dw'}} = p(w \mid d) \approx \sum_{t \in T} p(w \mid t)\, p(t \mid d) = \sum_{tin} \phi_{wt} \theta_{td} = \{\Phi\Theta\}_{wd}$$

or just

$$P \approx \Phi\Theta$$

Applying MLE:

$$L(\Psi, \Theta) = \sum_{d \in D} \sum_{w \in d} n_{dw} \ln \sum_{t} \psi_{wt} \theta_{td} \quad \longrightarrow \max_{\Psi, \Theta \text{ – stochastic}}$$

EM-algorithm is used for training.

Flat topic model.

BigARTM is good tool for it.

What we can do:

- Add regularisers:

$$L(\Psi, \Theta) + R(\Psi, \Theta) \longrightarrow \max$$

- Add modalities $m \in M$.

$$W = \bigsqcup_{m \in M} W_m \text{ and } \Phi = [\Phi_1 | \cdots | \Phi_n]$$

Flat topic model.

BigARTM is good tool for it.

What we can do:

- Add regularisers:

$$L(\Psi, \Theta) + R(\Psi, \Theta) \longrightarrow \max$$

- Add modalities $m \in M$.

$$W = \bigsqcup_{m \in M} W_m \text{ and } \Phi = [\Phi_1 | \cdots | \Phi_n]$$

Let's build **topic hierarchies**.

- Each level is a topic model.
- Next level is learned with **specific regulariser** to find parent topics from previous level.

Check out [6, 7].

Topic hierarchies.

- **Learned** parent level: topics $a \in A$ with $\Phi' \in \mathbb{R}^{|W| \times |A|}$ and $\Theta' \in \mathbb{R}^{|A| \times |D|}$.
- **To learn:**
  New level with topics $t \in T$ and $\Phi \in \mathbb{R}^{|W| \times |T|}$ and $\Theta \in \mathbb{R}^{|T| \times |D|}$.
  Parent-child relations Ψ_{ta} – t is a child of a.

Topic hierarchies.

- **Learned** parent level: topics $a \in A$ with $\Phi' \in \mathbb{R}^{|W| \times |A|}$ and $\Theta' \in \mathbb{R}^{|A| \times |D|}$.
- **To learn:**
  New level with topics $t \in T$ and $\Phi \in \mathbb{R}^{|W| \times |T|}$ and $\Theta \in \mathbb{R}^{|T| \times |D|}$.
  Parent-child relations Ψ_{ta} – t is a child of a.
- **Assumption:** parent topic is a mixture of children's:

$$p(w \mid a) \approx \sum_{t \in T} p(w \mid t) p(t \mid a) \qquad \text{or just} \qquad \Phi^{l} \approx \Phi\Psi$$

Topic hierarchies.

- **Learned** parent level: topics $a \in A$ with $\Phi' \in \mathbb{R}^{|W| \times |A|}$ and $\Theta' \in \mathbb{R}^{|A| \times |D|}$.
- **To learn:**
  New level with topics $t \in T$ and $\Phi \in \mathbb{R}^{|W| \times |T|}$ and $\Theta \in \mathbb{R}^{|T| \times |D|}$.
  Parent-child relations Ψ_{ta} – t is a child of a.
- **Assumption:** parent topic is a mixture of children's:

$$p(w \mid a) \approx \sum_{t \in T} p(w \mid t) p(t \mid a) \qquad \text{or just} \qquad \Phi^{l} \approx \Phi\Psi$$

- We can just add $|A|$ pseudo documents with n_{wa} counters

Topic hierarchies.

- **Learned** parent level: topics $a \in A$ with $\Phi' \in \mathbb{R}^{|W| \times |A|}$ and $\Theta' \in \mathbb{R}^{|A| \times |D|}$.
- **To learn:**
  New level with topics $t \in T$ and $\Phi \in \mathbb{R}^{|W| \times |T|}$ and $\Theta \in \mathbb{R}^{|T| \times |D|}$.
  Parent-child relations Ψ_{ta} – t is a child of a.
- **Assumption:** parent topic is a mixture of children's:

$$p(w \mid a) \approx \sum_{t \in T} p(w \mid t) p(t \mid a) \qquad \text{or just} \qquad \Phi^{l} \approx \Phi\Psi$$

- We can just add $|A|$ pseudo documents with n_{wa} counters
- The same point with Θ regularisation. $\Theta^{l} \approx \tilde{\Psi}\Theta$ It is like add new modality with tokens corresponding to $a \in A$.

Hierarchy sparsing

The goal: Topics should have small number of parents.

$p(a \mid t)$ should be sparse.

Similar to LDA regulariser:

$$R(\Psi) = \frac{1}{|A|} \sum_{a} \sum_{t} \ln p(a \mid t) = \frac{1}{|A|} \sum_{a} \sum_{t} \ln \frac{\psi_{ta}\, p(a)}{\sum_{a'} \psi_{ta'}\, p(a')}$$

To apply we need just to update M-step of EM-algorithm.

The same approach for Θ regularisation.

Hierarchical clustering approach

Scheme

- Get a codebase of library usage
- Somehow get a document representations in $\mathbb{R}^d$
- Build a hierarchical clusterization
- Show it for user API query
- ???
- PROFIT!

Hierarchical clustering approach

Scheme

- Get a codebase of library usage
- Somehow get a document representations in $\mathbb{R}^d$
- Build a hierarchical clusterization
- Show it for user API query
- ???
- PROFIT!

DocNADE

NADE – Neural Autoregressive Distribution Estimator [4].

Based on fact that $$p(v) = \prod_{d=1}^{D} p(v_d \mid v_{<d})$$

We need to **parametrise** $p(v_d \mid v_{<d})$.

DocNADE

NADE – Neural Autoregressive Distribution Estimator [4].

Based on fact that $$p(v) = \prod_{d=1}^{D} p(v_d \mid v_{<d})$$

We need to **parametrise** $p(v_d \mid v_{<d})$.

$$p(v_d \mid v_{<d}) = \mathrm{sigm}(b_d + V_{d,:}h_d)$$

$$h_d = \mathrm{sigm}(c + W_{:,<d}v_{<d}).$$

W, V, b, c – learnable parameters by LME.

source{d}

DocNADE

NADE – Neural Autoregressive Distribution Estimator [4].

Based on fact that $$p(v) = \prod_{d=1}^{D} p(v_d \mid v_{<d})$$

We need to **parametrise** $p(v_d \mid v_{<d})$.

$$p(v_d \mid v_{<d}) = \mathrm{sigm}(b_d + V_{d,:} h_d)$$

$$h_d = \mathrm{sigm}(c + W_{:,<d} v_{<d}).$$

W, V, b, c – learnable parameters by LME. Softmax is used for vectors modeling:

$$p(v_d \mid v) = \frac{\exp(b_{w_b} + V_{w_d,:} h_d)}{\sum_w \exp(b_w + V_{w,:} h_d)}$$

DocNADE

NADE – Neural Autoregressive Distribution Estimator [4].

Based on fact that $$p(v) = \prod_{d=1}^{D} p(v_d \mid v_{<d})$$

We need to **parametrise** $p(v_d \mid v_{<d})$.

$$p(v_d \mid v_{<d}) = \mathrm{sigm}(b_d + V_{d,:}h_d)$$

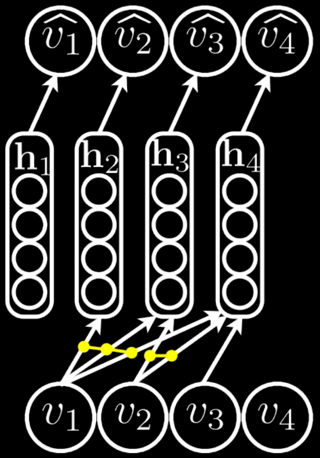
$$h_d = \mathrm{sigm}(c + W_{:,<d}v_{<d}).$$

W, V, b, c – learnable parameters by LME. Softmax is used for vectors modeling:

$$p(v_d \mid v) = \frac{\exp(b_{w_b} + V_{w_d,:}h_d)}{\sum_w \exp(b_w + V_{w,:}h_d)}$$

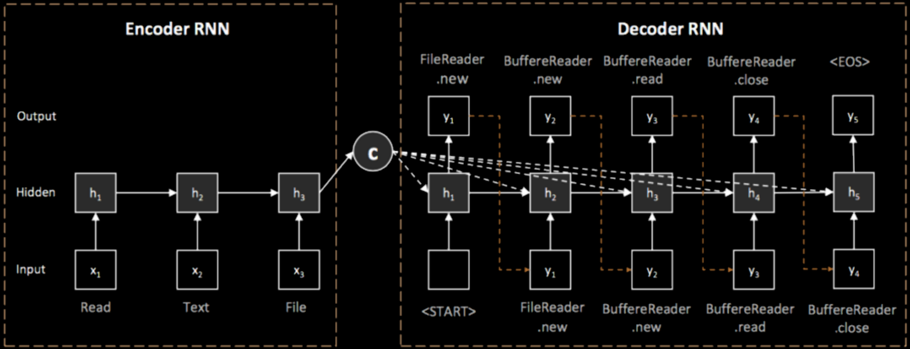
Trains on random permutations of the words in a document.

Document representation is h_T at the final timestep T.



Deep learning way

- **Aim:** Generate API sequences for a natural language query [3].
- **Method:** RNN encoder-decoder model for API learning.
- **Data:** annotated code snippets collected from GitHub.



source{d}

Deep learning way

Details:

- Run on sequences of API methods only.
- RNN is **GRU**, Encoder is bidirectional with attention, 1000 hidden units, 120 dimension of word embeddings.
- Beam Search for generation several API sequences to choose

Deep learning way

Details:

- Run on sequences of API methods only.
- RNN is **GRU**, Encoder is bidirectional with attention, 1000 hidden units, 120 dimension of word embeddings.
- Beam Search for generation several API sequences to choose
- IDF-based weights for API as a penalty term in **loss**:

$$\mathrm{loss}_{it} = -\log p_\theta(y_{it} \mid x_i) - \lambda \log\left(\frac{N}{n_t}\right)$$

where
i is i-th train instance,
t is t-th target word in instance i,
N is the total number of API sequences,
n_t is the number of sequences where the API t appears.

Probabilistic way. Interesting Sequence Mining

Task: get meaningful API patterns $\mathcal{I}$ [1, 2].

Idea: Use API patterns $\mathcal{I}$ to define a code probability in database X.

Pattern is interesting code **subsequence**.

Probabilistic way. Interesting Sequence Mining

Task: get meaningful API patterns $\mathcal{I}$ [1, 2].

Idea: Use API patterns $\mathcal{I}$ to define a code probability in database X.

Pattern is interesting code **subsequence**.

Simplified model:

$$p(X, z \mid \mathcal{I}) \sim \prod_{i \in \mathcal{I} \cap X} p_i^{z_i} (1 - p_i)^{1 - z_i}$$

X – code database,
$\mathcal{I}$ – set of API patterns,
p_i – API pattern $i \in \mathcal{I}$ probability,
z_i – indicator of including pattern into code (hidden).

Probabilistic way. Interesting Sequence Mining

Task: get meaningful API patterns $\mathcal{I}$ [1, 2].

Idea: Use API patterns $\mathcal{I}$ to define a code probability in database X.

Pattern is interesting code **subsequence**.

Simplified model:

$$p(X, z \mid \mathcal{I}) \sim \prod_{i \in \mathcal{I} \cap X} p_i^{z_i} (1 - p_i)^{1 - z_i}$$

X – code database,
$\mathcal{I}$ – set of API patterns,
p_i – API pattern $i \in \mathcal{I}$ probability,
z_i – indicator of including pattern into code (hidden).

Example:

$$X = \{\ d\ b\ c\ e\ d\ f\ f;\ \ e\ e\ d\ f\ f\ f;\ \ d\ f\ d\ e\ f\ f;\ \}$$

$$\mathcal{I} = \{\ [\,b\ c\ e\,]\ [\,d\ f\,]\ [\,d\ f\,]\ [\,e\ f\,]\ \}$$

	[b c e]	[d f]	[d f]	[e f]
	1	1	1	0
z :	0	1	0	1
	0	1	1	1
p_i :	0.33	1	0.66	0.66

Probabilistic way. Interesting Sequence Mining

Solver: EM-algorithm.

Iterate:

1. Structural-EM ($\mathcal{I}$ update)
   1.1 Somehow generate candidate S'
   1.2 See if quality increases
2. Hard-EM (z and p update)
   2.1 Find patterns from $\mathcal{I}$ that was used to sample X with greedy search.

$$z = \arg\max_z \log p(z \mid p, \mathcal{I}; X)$$

   2.2 Update p_i by averaging z.

Example:

$$X = \{\ d\ b\ c\ e\ d\ f\ f;\quad e\ e\ d\ f\ f\ f;\quad d\ f\ d\ e\ f\ f;\ \}$$

$$\mathcal{I} = \{\ [\,b\ c\ e\,]\ [\,d\ f\,]\ [\,d\ f\,]\ [\,e\ f\,]\ \}$$

	[b c e]	[d f]	[d f]	[e f]
z :	1	1	1	0
	0	1	0	1
	0	1	1	1
p_i :	0.33	1	0.66	0.66

References. Links

1. Hierarchical Multimodal Topic Modeling presentation. N. A. Chirkova and K. V. Vorontsov
2. BigARTM for Topic Modeling
3. Post about DocNADE with implementation
4. Probabilistic API Mining code

References. Articles i

[1] J. Fowkes and C. Sutton.
Parameter-free probabilistic api mining across github.
In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 254–265. ACM, 2016.

[2] J. Fowkes and C. Sutton.
A subsequence interleaving model for sequential pattern mining.
arXiv preprint arXiv:1602.05012, 2016.

[3] X. Gu, H. Zhang, D. Zhang, and S. Kim.
Deep api learning.
In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 631–642. ACM, 2016.

[4] H. Larochelle and I. Murray.
The neural autoregressive distribution estimator.
In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.

References. Articles ii

[5] J. Starke, C. Luce, and J. Sillito.
Searching and skimming: An exploratory study.
In *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, pages 157–166. IEEE, 2009.

[6] K. Vorontsov.
Additive regularization for topic models of text collections.
In *Doklady Mathematics*, volume 89, pages 301–304. Springer, 2014.

[7] K. Vorontsov and A. Potapenko.
Tutorial on probabilistic topic modeling: Additive regularization for stochastic matrix factorization.
In *International Conference on Analysis of Images, Social Networks and Texts_x000D_*, pages 29–46. Springer, 2014.