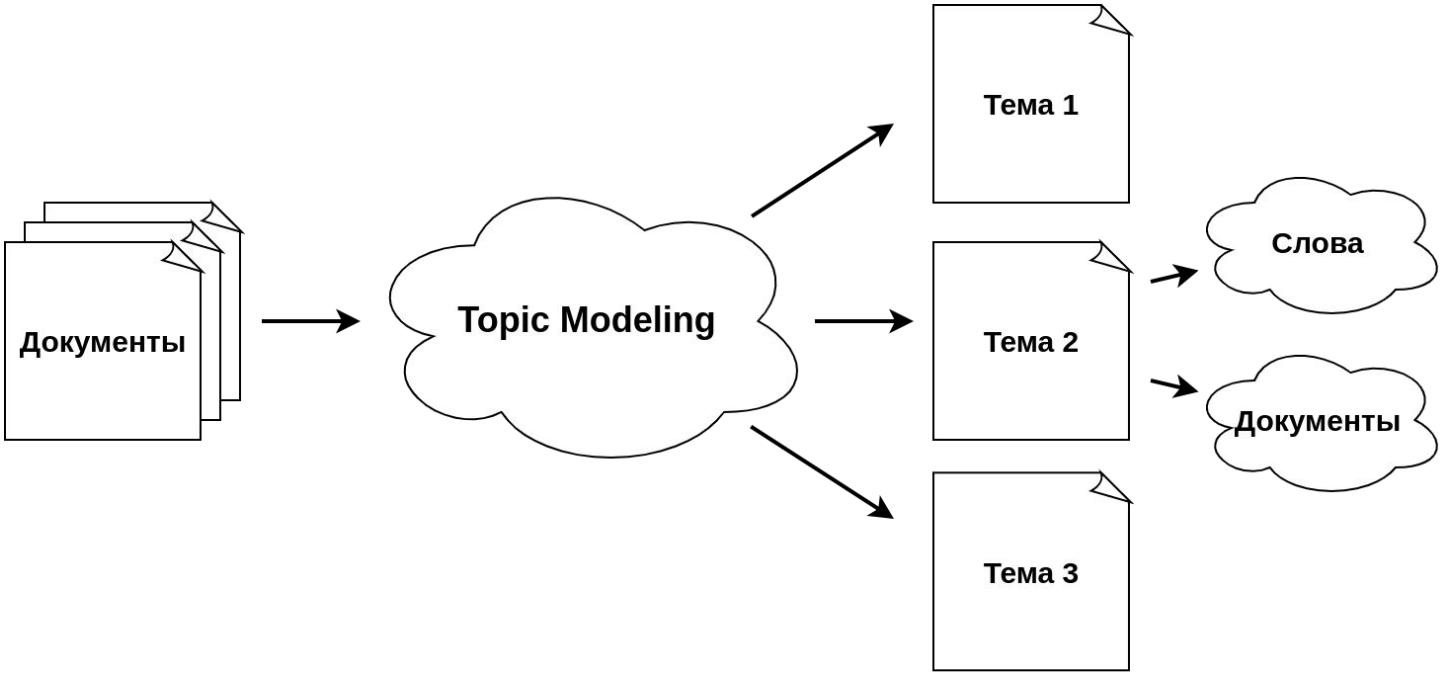
Использование моделирования тем в задачах SE

Богомолов Егор Олегович

Научный руководитель: Брыксин Тимофей Александрович

НИУ ВШЭ СПбШФМКН, 2021

Тематическое моделирование (TM)



Мотивация

- Несмотря на длительное развитие области машинного обучения в задачах SE, применимость методов на практике остается ограниченной
- Разработчики могут игнорировать предсказания, даже если в целом считают их релевантными[1]
- Возможным решением является объяснение предсказаний моделей
- Тематическое моделирование является одним из способов получить информацию из большого массива данных в *интерпретируемом* виде

[1] Kovalenko et al., “Does Reviewer Recommendation Help Developers?”, 2018

Тематическое моделирование в SE

- Суммаризация кода[1, 2, 3]
- Локализация дефектов[4, 5]
- Приоритезация выполнения тестов[6]
- Рекомендация рефакторингов[7]
- Поиск дубликатов сообщений об ошибках[8]
- Назначение разработчиков для исправления ошибок (issue)[9, 10, 11]

Используемые методы: LSA, LDA, ARTM и их модификации

[1] Gelman et al., “A language-agnostic model for semantic source code labeling”, 2019
[2] Saeidi et al., “ITMViz: Interactive Topic Modeling for Source Code Analysis”, 2015
[3] McBurney et al., “Improving topic model source code summarization”, 2014
[4] Chen et al., “Explaining software defects using topic models”, 2012
[5] Wang et al., “Bug Localization via Supervised Topic Modeling”, 2018
[6] Thomas et al., “Static test case prioritization using topic models”, 2014
[7] Bavota et.al, “Methodbook: Recommending Move Method Refactorings via Relational Topic Models”, 2014
[8] Nguyen et al., “Duplicate bug report detection with a combination of information retrieval and topic modeling”, 2012,
[9] Xia et al., “Accurate developer recommendation for bug resolution”, 2013
[10] Zhang et al., “A Novel Developer Ranking Algorithm for Automatic Bug Triage Using Topic Model and Developer Relations”, 2014
[11] Alenezi et al., “Using Categorical Features in Mining Bug Tracking Systems to Assign Bug Reports”, 2018

Особенности существующих подходов

Используемые подходы к ТМ:

- Не учитывают специфику работы с исходным кодом
- Используют ТМ как самостоятельный инструмент для
  - Представления информации из кода в более простом виде
  - Анализа состояния и трендов в области
  - Построения явных правил для решения задач (рефакторинги, приоритезация тестов)

Тематическое моделирование не используется как инструмент для извлечения факторов для других моделей

Цель и задачи

Цель: предложить подход для извлечения интерпретируемых признаков из исходного кода на основе тематического моделирования и проверить его применимость в задачах SE

Задачи:

- Разработать подход для извлечения тем из кода произвольной гранулярности, учитывающий специфику работы с исходным кодом
- Разработать способ получения интерпретируемых факторов на основе построенной тематической модели
- Проверить применимость извлеченных факторов в следующих задачах:
  - Поиск похожих проектов
  - Рекомендация разработчиков для исправления ошибок

Особенности ТМ на исходном коде

- Размер словаря в коде значительно больше, чем в естественном языке[1]
  - Через CamelCase или snake_case объединены целые фразы
  - Используются слова из разных языков
  - Вводятся новые термины
- Для уменьшения размера словаря требуется разбиение токенов на части по CamelCase/snake_case или применение Byte Pair Encoding (BPE)
- Разбиение токенов на части усложняет интерпретацию тем по частотным словам, так как части слов могут не иметь понятного человеку смысла

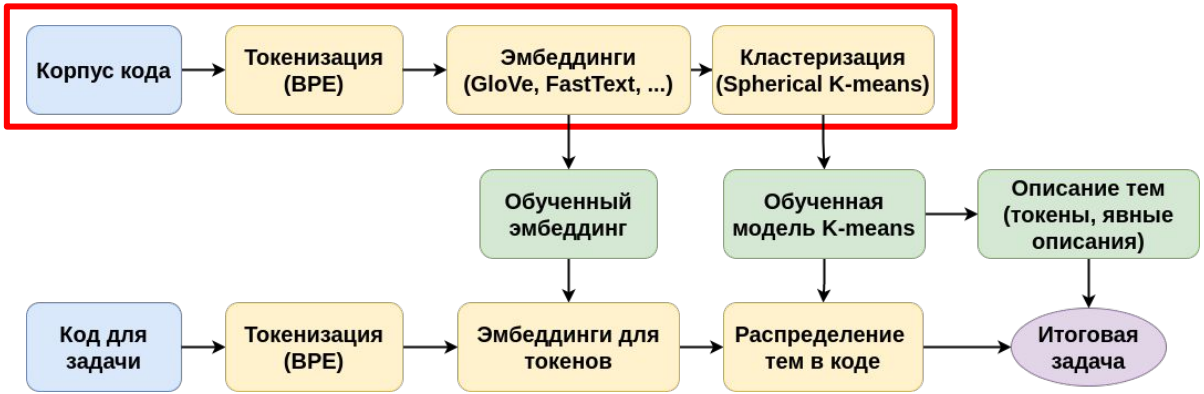
[1] Karampatsis et al., “Big code != big vocabulary: open-vocabulary models for source code”, 2020

Code2Topic: TM с учетом специфики кода

Предложен алгоритм *Code2Topic*, учитывающий специфику словаря в коде:

1. Извлекает информацию о *частях* токенов при помощи эмбеддингов (FastText/GloVe)
2. Строит векторы для токенов путем усреднения векторов частей[1]
3. Кластеризует векторы, выделяя группы семантически схожих токенов (Spherical K-Means)

Кластеры токенов соответствуют темам, смысл целых токенов понятен человеку



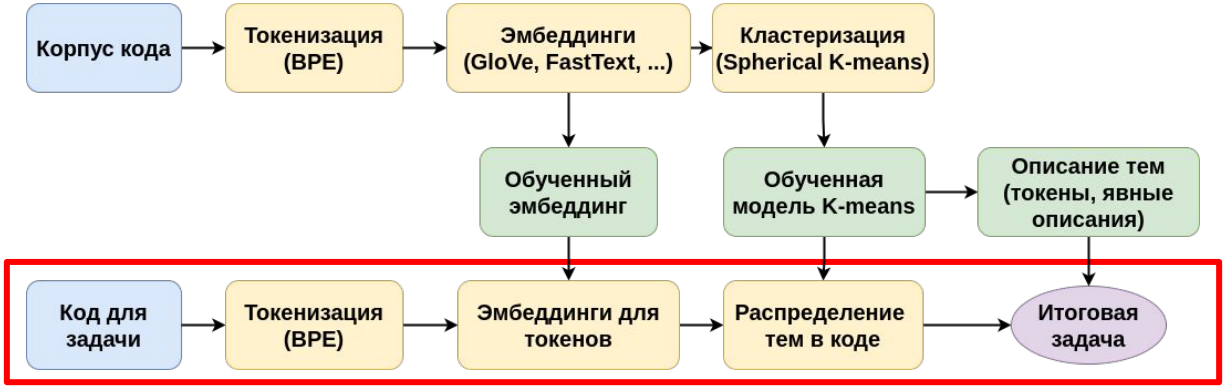
[1] Heinzerling et al., “BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages”, 2018

Code2Topic: получение факторов из кода

Код представляется в виде распределения тем среди его идентификаторов

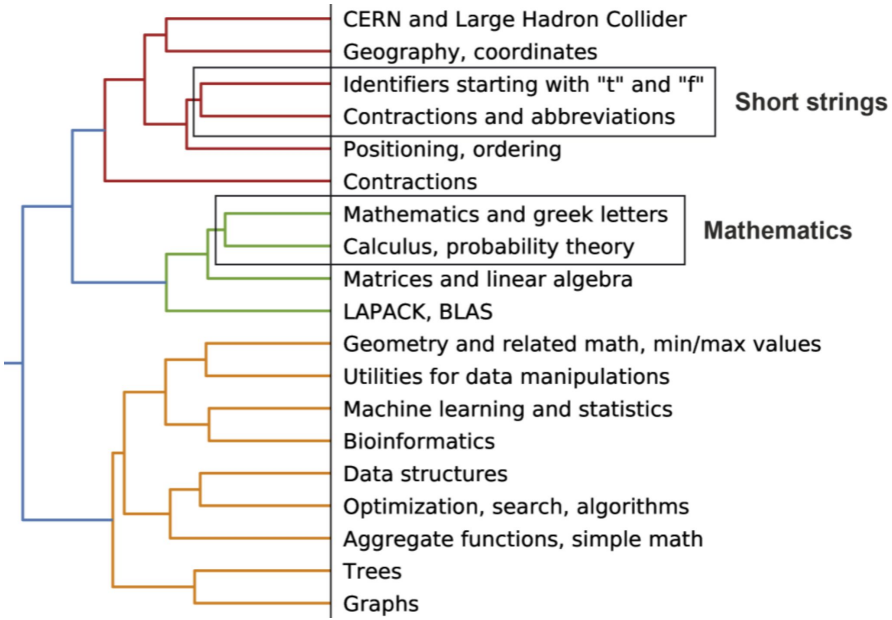
- Для проектов/файлов/методов извлечение факторов требует токенизации кода и применения обученной модели *Code2Topic*
- Для разработчиков из Git извлекаются все изменения, которые они совершали в доступных проектах, и *Code2Topic* применяется к добавленным токенам

Полученные факторы можно интерпретировать, если темы имеют понятный смысл



Code2Topic: интерпретация тем

- Для интерпретации тем обычно используют набор токенов, относящихся к ним
- Интерпретация проще, если темам дать короткие текстовые описания, но это требует экспертизы и времени
- Для более наглядной визуальной интерпретации и упрощения разметки предложено построить дендрограмму объединения центров кластеров
- Эксперты размечают бинарные деления в дендрограмме, а не отдельные кластеры



Поиск похожих репозиториев на GitHub

Реализован *Sosed*, инструмент для поиска похожих репозиториев среди 9 миллионов проектов из GitHub на 15 языках:

- Репозитории представляются в виде распределения тем среди их токенов
- Похожие репозитории определяются по косинусной близости или KL-дивергенции распределений тем
- Для оценки качества были размечены top-5 рекомендаций для 94 репозиториев:
  - Средняя релевантность *Sosed* – top-5: **4,2**, top-1: **4,7**
  - Средняя релевантность *Gazer* – top-5: **3,7**, top-1: **4,3**
- Предсказания интерпретируемы благодаря текстовым описаниям тем

Статья об инструменте[1] представлена на конференции ASE'2020

[1] Bogomolov et al., "*Sosed*: a tool for finding similar software projects", 2020

Рекомендация разработчиков для решения issue

Задача: по сообщению об ошибке определить программиста, который должен её исправить

- Для тестирования использован исторический датасет из 9700 issue, собранный внутри системы YouTrack
- За основу взята модель JetBrains Baseline: XGBoost + факторы о связи автора issue, программиста и текста сообщения
- К факторам добавлены распределения тем в коде программистов, полученные при помощи *Code2Topic*, и частоты слов, релевантных для каждой темы
- Точность предсказаний улучшена с 68% до 75%

Модель	Точность (%)
Time TF-IDF [Ramin, 2015]	42,3
Word2Vec+FCN [Guo, 2020]	44,7
Word2Vec+CNN [Guo, 2020]	52,6
JetBrains Baseline	68,4
JetBrains Baseline + *Code2Topic* (Эта работа)	**75,1**

Выводы

- Предложен подход для извлечения тем из кода при помощи эмбеддингов
- Предложен способ для представления и разметки полученных тем при помощи дендрограммы
- Подход верифицирован на задаче поиска похожих репозиториев
  - Средняя релевантность в top-5 улучшена с 3.7 до 4.2
- Извлеченные факторы для разработчиков применены в задаче рекомендации разработчиков для исправления ошибок
  - Улучшена точность с 68% до 75%
- Статья про инструмент для поиска похожих репозиториев представлена на конференции Automated Software Engineering 2020 (Core A)

Размеченные кластеры для GitHub

Drawing	CERN and Large Hadron Collider	Warcraft	CPU sensors	Identifiers starting with "j"	General Java	Random strings	Random short strings
draw	rpt	wailing	level	jnicall	exception	tmm	sld
background	muon	scourgebane	temp	ljava	java	qtprf	xin
fill	rocp	berserker	power	jenv	lang	fmi	gggg
rect	ecal	terrifyingroar	intval	jniexport	util	tvc	xyval
graphics	calo	garrote	debounce	jresult	org	cme	npt

Sosed: пример вывода

Запрос: IntelliJ IDEA Community (open-source version of the IDEA IDE)

Результаты:

- go-lang-plugin-org/go-lang-idea-plugin – Go language **IDE** built on the IntelliJ Platform
- eclipse/che – Kubernetes-Native **IDE** for developer teams
- consulo/consulo – multi-language **IDE**
- carymrobbins/intellij-haskforce – Haskell plugin for IntelliJ IDEA
- aptana/studio3 – Web development **IDE**

Sosed: интерпретация предсказаний

$$cosine\_similarity(P_Q, P_R) = \sum_{c \in Clusters} \frac{P_Q(c)}{||P_Q||} \cdot \frac{P_R(c)}{||P_R||}$$

$$D_{KL}(P_Q||P_R) = \sum_{c \in Clusters} P_Q(c) \log \frac{P_Q(c)}{P_R(c)} \longrightarrow$$

$$\min_{P_R} \sum_{c \in Clusters} P_Q(c) \log \frac{P_Q(c)}{P_R(c)} =$$

$$\min_{P_R} \sum_{c \in Clusters} P_Q(c) \log P_Q(c) - P_Q(C) \log P_R(c) =$$

$$\min_{P_R} \sum_{c \in Clusters} -P_Q(C) \log P_R(c) =$$

$$\max_{P_R} \sum_{c \in Clusters} P_Q(C) \log P_R(c)$$

Sosed: интерпретация предсказаний

Query project: github.com/tensorflow/tensorflow

Output: github.com/dmlc/xgboost | similarity = 0.8710

Intersecting topics:

intersection = 0.21 | Data, data types, read-write |
input (139925), output (110979), data (81043), std (71129), from (17618)

intersection = 0.21 | Machine learning and statistics |
tensor (129095), tensorflow (45720), dataset (23191), outputs (15761), weights (15733)

intersection = 0.11 | Common programming words |
get (96002), type (86532), value (58794), string (57437), set (38336)

intersection = 0.04 | Data structures |
array (49665), index (34666), values (21846), all (21687), list (21408)

intersection = 0.04 | Memory management |
size (77894), ptr (67072), new (19412), memory (17057), allocator (6217)