Рекомендательные системы на основе профиля интересов пользователей

Илия Кузьмина, 344
Научные руководители: С.И.Николенко
ПОМИ РАН
Р. Галинский
EMC

Поставленные задачи

- Реализовать алгоритм построения профиля интересов пользователя, основываясь на их 

- Реализовать рекомендательный алгоритм
- Изучить предметную область topic modeling
- Освоить python

Используемые технологии

python™

word2vec

Tool for computing continuous distributed representations of words.

gensim

topic modelling for humans

Word2Vec

- Каждое слово представляется в виде вектора размерности K
- K-мерное семантическое пространство:
  - схожие по контексту слова оказываются векторами на маленьком расстоянии
  -
```
>>> model.most_similar(positive=['woman', 'king'],
                       negative=['man'])
    [('queen', 0.50882536), ...]
```

Кластеризация векторов → разбиение слов на темы?

Vec2Topic

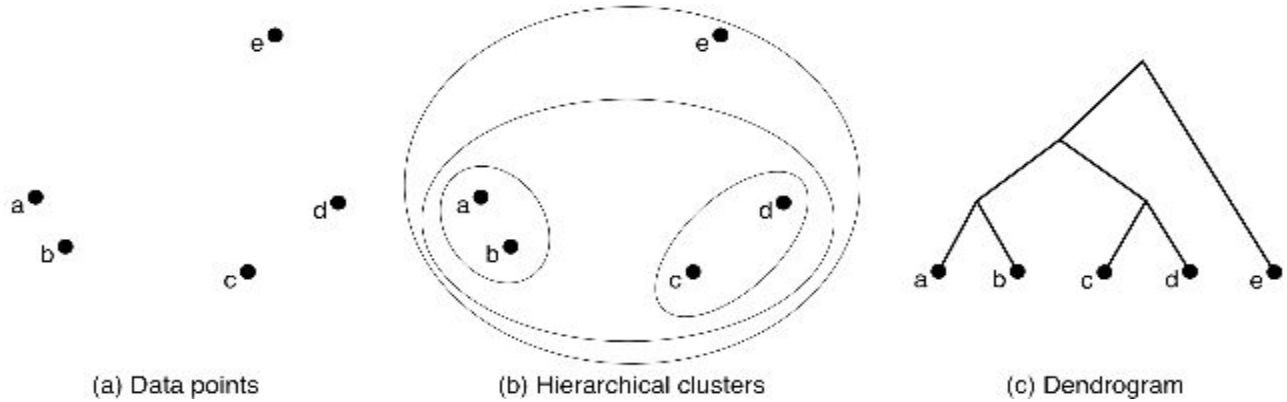
Как по кластеру из >>10 слов понять, о чём он?

Topical words:

- **depth(w)** = глубина слова **w** в дереве кластеров
- **degree(w)** = количество уникальных слов, которые встречаются с **w** в одном предложении

Depth(w) – Agglomerative Clustering

- Разобьём все векторы на N кластеров
- На каждом шаге будем находить два ближайших кластера и объединять их в один большой
- Таким образом, в конце получится дерево вложенных кластеров, в котором корень содержит все векторы



Depth(w) – Agglomerative Clustering

- Сложность наивного алгоритма - $O(N^3 \cdot K)$
- Можно заметить, что необязательно на каждом шаге считать заново все попарные расстояния между кластерами
- Будем пересчитывать расстояния только для тех кластеров, у которых ближайший сосед только что объединился с кем-то в один кластер

$\Rightarrow$ $O(C \cdot N^2 \cdot K)$

Degree(w)

- Слишком большой объём данных, чтобы посчитать степень каждого слова за один проход

Решение проблемы:

- Тексты постов были разбиты на 200 частей по 50 Мб. Для каждого файла посчитаем все уникальные пары слов, которые встречаются в одном предложении, отсортируем, сохраним результаты в файл
- Пройдём по полученным 200 файлам указателями, чтобы посчитать степени вершин

Мера(вес) слова

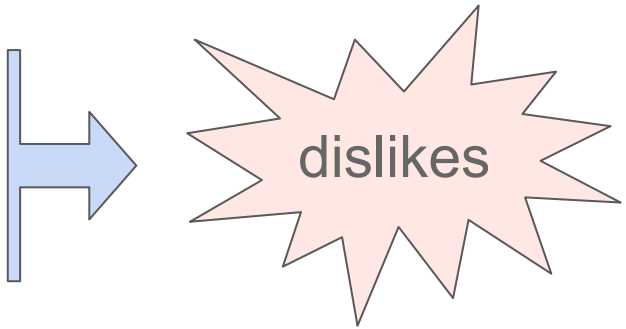
$$score(w) = \frac{depth(w)}{max_{u \in V} depth(u)} \cdot \frac{log(1 + degree(w))}{log(1 + max_{u \in V} degree(u))}$$

$$score(t) = \frac{\sum_{w \in t} score(w)}{|t|}$$

Реальные данные

- likes:
  `(group_id, post_id) {(user_id, timestamp(ms))...}`
- посты:
  `group_id author_id post_id timestamp(iso) text`

- likes:
  `user_id (group_id, post_id)...`
- посты: на каждую группу по файлу, посты отсортированы по времени публикации
  `post_id text`



Примеры

- **Topic #10: 0.57949**
  торт глазурь крем тесто корж противень
  Topic #6: 0.54294
  молоко яйцо мука сахара ложка сметана желатина какао-порошок масло приготовление столовый сахар крахмал ванильный маргарин лимонный ванилин заварной ст щепотка
- **Topic #21: 0.69140**
  пензенский пенза алексий азовский пресвятой
  Topic #16: 0.67656
  богоматерь общецерковный святой божий матерь чудотворный
- **Topic #6: 0.43481**
  арбидол биопарокс композитум неврохель валерианохель траумель канефрон лимфомиозот
  Topic #21: 0.42425
  лекарственный зверобой экстракт боярышник пассифлора страстоцвет бузина

Рекомендательный алгоритм

- 2 тренировочных набора данных:
  likes, dislikes → score_like(t), score_dislike(t)

- p(like|d) <> p(dislike|d)

$$W(t) = \sum_{w \in d} score_{like}(w) \quad p(like|d) = \sum_{t} W(t) * score_{like}(t)$$

- Тестовые наборы данных:
  2000 пользователей, >= 20 likes, >=20 dislikes
  precision 0.68 = и-п / (и-п + л-п)
  recall 0.74 = и-п / (и-п + л-о)

Результаты

- Реализован алгоритм построения профиля интересов пользователя, основывающийся на их лайках
- Реализован рекомендательный алгоритм