Лабораторна робота №5

Моделювання тем

Мета роботи: Ознайомитись з вирішенням задач пошуку ключових слів та моделювання тем.

Короткі теоретичні відомості

Загалом існує кілька операцій, які можна виконати над текстовими документами, а саме:

• Виділення ключових фраз: фокусується на вилученні ключових фраз із документів.

• Моделювання теми: виділення різноманітних концепцій або теми, присутні в документах, зберігаючи основні теми в цих документах.

• Узагальнення документів: резюмування цілих текстових документів, щоб виділити суть, яка зберігає важливі аспекти всього корпусу.

Виділення ключових слів, також відоме як вилучення термінології, — це процес вилучення ключових термінів або фраз із основної частини неструктурованого тексту, щоб охопити основні теми.

Існують різні підходи до виділення ключових фраз, з яких розглянемо наступні:

• Сполучення

• Вилучення фраз на основі зважених тегів

Словосполучення можна визначити як послідовність або групу слів, які, як правило, трапляються часто, і ця частота, як правило, є більшою, ніж те, що можна назвати випадковою появою.

Різні типи словосполучень можуть бути утворені на основі частин мови, таких як іменники, дієслова тощо. Існують різні способи витягти сполучення, і один із найкращих способів зробити це — використовувати підхід групування чи сегментації n-грам.

Розглянемо засоби пошуку сполучень бібліотеки NLTK, які дозволяють знаходити сполучення за допомогою різних показників, таких як необроблені частоти, поточкова взаємна інформація тощо.

Поточкову взаємну інформацію можна обчислити для двох подій або термів як логарифм відношення ймовірності їх спільного виникнення до добутку їхніх індивідуальних ймовірностей, припускаючи, що вони незалежні одна від одної.

Завантажимо функції для пошуку сполучень та визначення тих, що зустрічаються найчастіше, або тих, що мають найвищі значення інших показників, наприклад, поточкової взаємної інформації:

from nltk.collocations import BigramCollocationFinder

from nltk.collocations import BigramAssocMeasures

bigram_measures = BigramAssocMeasures()

finder = BigramCollocationFinder.from_documents([item.split() for item in emma])

Біграми з найвищою поточковою взаємною інформацією

finder.nbest(bigram_measures.pmi, 10)

Те ж саме можна порахувати для триграм та чотириграм.

from nltk.collocations import TrigramCollocationFinder

from nltk.collocations import TrigramAssocMeasures

finder = TrigramCollocationFinder.from_documents([item.split() for item in emma])

trigram_measures = TrigramAssocMeasures()

Моделювання теми передбачає виділення ознак із термів документа та використання математичних структур і алгоритмів, таких як матрична факторізація та сингулярний розклад матриці, для створення кластерів або груп термів, які відрізняються один від одного, і ці кластери термів утворюють теми чи концепції. Існують різні підходи та алгоритми для побудови моделей тем. Розглянемо наступні методи:

• Приховане семантичне індексування

• Прихований розподіл Діріхле

• Розклад невід'ємних матриць

Приховане семантичне індексування (LSI). Основний принцип LSI полягає в тому, що подібні терми, як правило, використовуються в тому самому контексті, а отже, частіше зустрічаються разом. Назва приховане семантичне індексування походить від того, що ця техніка має здатність розкривати приховані терми, які семантично корелюють, для формування тем. Вона використовує сингулярний розклад матриці.

Бібліотека Gensim має реалізацію моделі прихованого семантичного індексування.

Total_topics = 10

lsi_bow = gensim.models.LsiModel(bow_corpus, id2word=dictionary, num_topics=Total_topics, onepass=True, chunksize=1740, power_iters=1000)

Приховане семантичне індексування в scikit-learn.

from sklearn.decomposition import TruncatedSVD

Total_topics = 10

lsi_model = TruncatedSVD(n_components=Total_topics, n_iter=500, random_state=0)

document_topics = lsi_model.fit_transform(cv_features)

Кожна тема буде мати числове вираження з різними знаками. По суті, ми зменшуємо вимірність до 10-вимірного простору на основі кількості тем, то ж знак на кожному термі вказує на напрям або орієнтацію у векторному просторі для конкретної теми. Чим вища вага, тим важливіший внесок. Отже, схожі корельовані терміни мають однаковий знак або напрям.

Прихований розподіл Діріхле - це генеративна ймовірнісна модель, у якій передбачається, що кожен документ має комбінацію тем, подібну до імовірнісної моделі прихованого семантичного індексування.

Цю модель також реалізовано у бібліотеці Gensim.

lda_model = gensim.models.LdaModel(corpus=bow_corpus, id2word=dictionary, chunksize=1740, alpha='auto', eta='auto', random_state=0, iterations=500, num_topics=Total_topics, passes=20, eval_every=None)

Та у scikit-learn:

from sklearn.decomposition import LatentDirichletAllocation

lda_model = LatentDirichletAllocation(n_components = Total_topics,

max_iter=500, max_doc_update_iter=50, learning_method='online',

batch_size=1740, learning_offset=50., random_state=0, n_jobs=16)

document_topics = lda_model.fit_transform(cv_features)

Розклад невід'ємних матриць.

from sklearn.decomposition import NMF

nmf_model = NMF(n_components= Total_topics, solver='cd', max_iter=500, random_state=0, alpha=.1, l1_ratio=.85, init="nndsvd")

document_topics = nmf_model.fit_transform(cv_features)

Узгодженість тем є складною темою сама по собі, і її можна певною мірою використовувати для вимірювання якості моделей тем. Як правило, набір тверджень вважається узгодженим, якщо вони підтримують одне одного.

topics_coherences = lda_model.top_topics(bow_corpus, topn=20)

avg_coherence_score = np.mean([item[1] for item in topics_coherences])

Як правило, чим менше заплутаність, тим краща модель. Подібним чином, чим нижча оцінка UMass і чим вище оцінка Cv в узгодженості, тим краща

модель. Також можна перебрати моделі з різною кількістю тем і визначити оптимальну кількість.

cv_coherence_model_lda = gensim.models.CoherenceModel(model=lda_model, corpus=bow_corpus, texts=norm_corpus_bigrams, dictionary=dictionary, coherence='c_v')

avg_coherence_cv = cv_coherence_model_lda.get_coherence()

umass_coherence_model_lda = gensim.models.CoherenceModel(model=lda_model, corpus=bow_corpus, texts=norm_corpus_bigrams, dictionary=dictionary, coherence='u_mass')

avg_coherence_umass = umass_coherence_model_lda.get_coherence()

perplexity = lda_model.log_perplexity(bow_corpus)

В scikit-learn можна оцінити модель прихованого розподілу Діріхле за допомогою заплутаності:

lda_model.perplexity

Та логарифмічної подібності (чим більша, тим краще):

lda_model. score

Завдання до лабораторної роботи

Створити програму, яка зчитує заданий набір даних, виділяє текстову частину даних (вони розглядаються як документи), виконує попередню обробку та завдання відповідно до варіанту. Якщо недостатньо ресурсів для роботи з повним набором даних, можна виділити частину, але таким чином, щоб були присутні усі класи.

Оформити звіт. Звіт повинен містити:

— титульний лист;
— код програми;
— результати виконання коду;

Продемонструвати роботу програми та відповісти на питання стосовно теоретичних відомостей та роботи програми.