Векторное представление слов

Курс «Интеллектуальные информационные системы»

Кафедра управления и интеллектуальных технологий

НИУ «МЭИ»

Весна 2023 г.

Векторное представление слова

или word embedding – как из слова получить вектор?

- Взять вектор размерностью = длине словаря, каждое слово в словаре – вектор, состоящий из 0 и одной 1 на соответствующей позиции (one-hot encoding, OHE)

‘стол’ = [0,0,0,0,0,1,0,0,0,0,0,0,0,0,0]

‘стул’ = [0,0,0,0,0,0,0,0,0,0,0,0,1,0,0] - не обладает свойством семантической близости

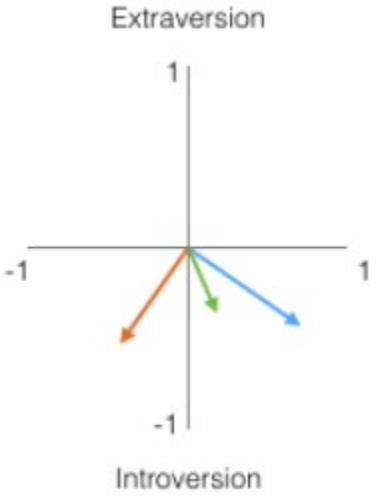
По сути – получили обычный «мешок слов»

- Хорошо бы, чтобы близкие друг к другу с семантической точки зрения слова имели бы близкие вектора.

Нужен вектор в «пространстве смыслов».

Векторное представление слова

Openness to experience	79 out of 100
Agreeableness	75 out of 100
Conscientiousness	42 out of 100
Negative emotionality	50 out of 100
Extraversion	58 out of 100



	Trait #1	Trait #2
Jay	-0.4	0.8
Person #1	-0.3	0.2
Person #2	-0.5	-0.4

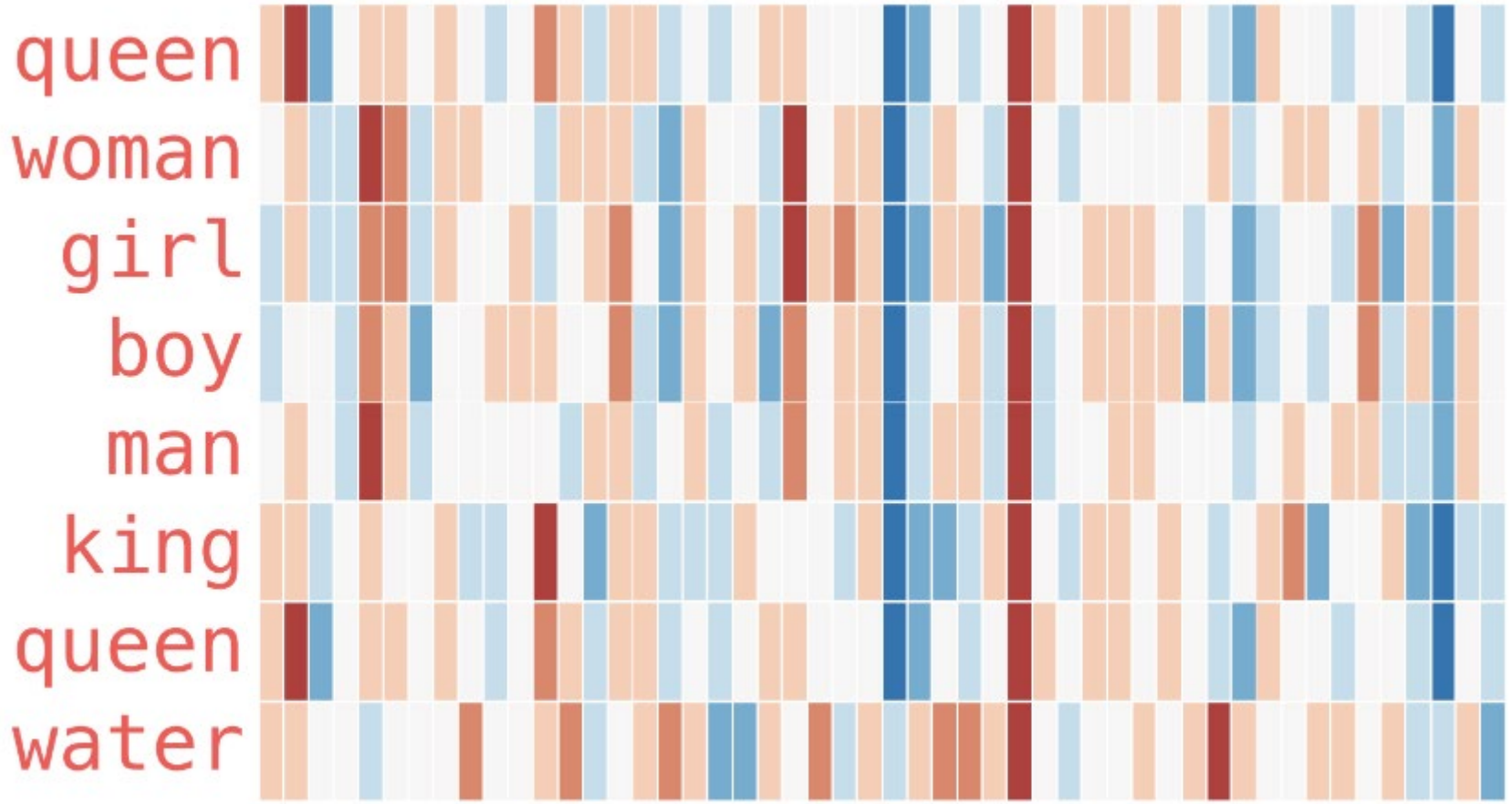
cosine_similarity(Jay [-0.4, 0.8] , Person #1 [-0.3, 0.2]) = 0.87 ✓

cosine_similarity(Jay [-0.4, 0.8] , Person #2 [-0.5, -0.4]) = -0.20

Можем сравнить двух людей по характерам.
Чем больше измерений – тем точнее результат

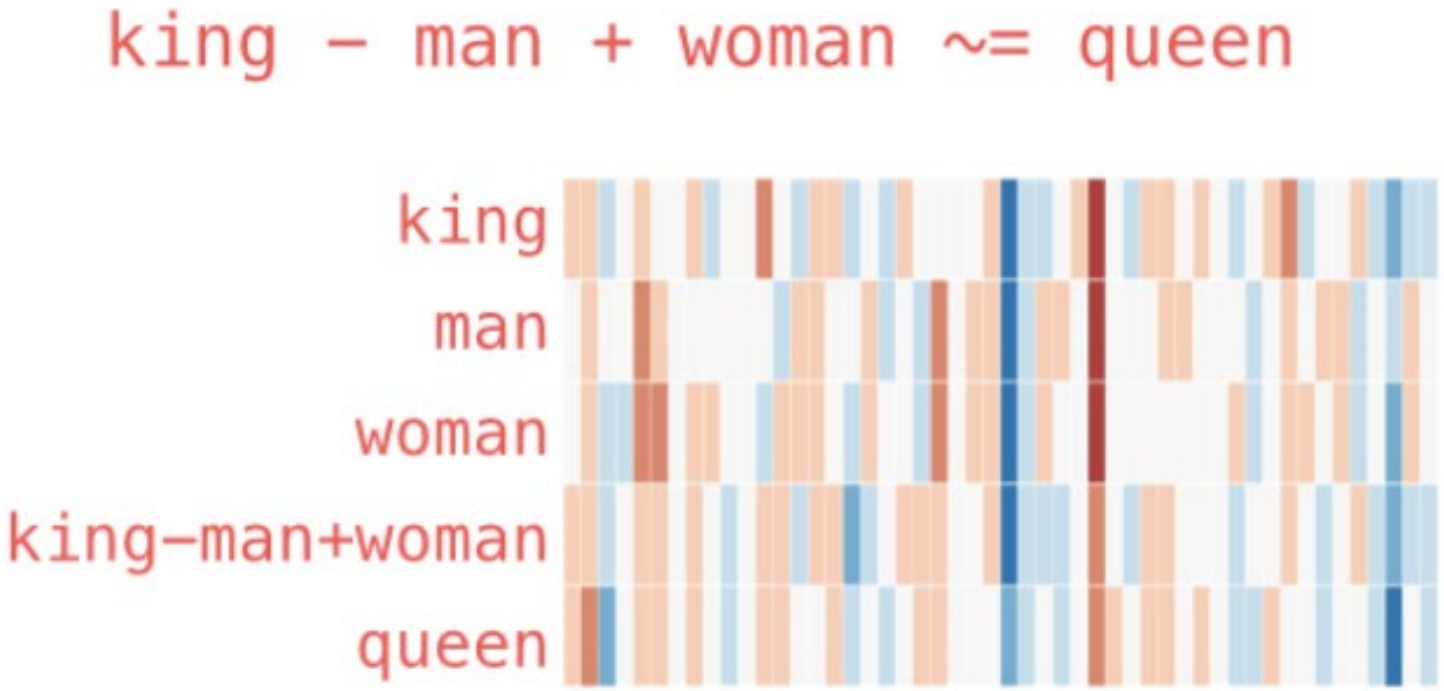
На основе материала https://jalammar.github.io/illustrated-word2vec/

Векторное представление слова



На основе материала https://jalammar.github.io/illustrated-word2vec/

Векторное представление слова



На основе материала https://jalammar.github.io/illustrated-word2vec/

Word2Vec

Word2vec — программный инструмент анализа семантики естественных языков, представляющий собой технологию, которая основана на дистрибутивной семантике и векторном представлении слов. Этот инструмент был разработан группой исследователей Google в 2013 году. Работу над проектом возглавил Томаш Миколов

- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space // In Proceedings of Workshop at ICLR, 2013

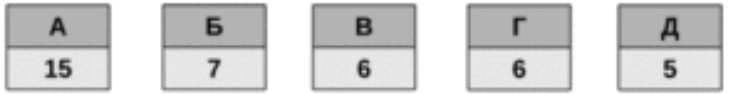
Работа этой технологии осуществляется следующим образом: word2vec принимает большой текстовый корпус в качестве входных данных и сопоставляет каждому слову вектор, выдавая координаты слов на выходе. Сначала он создает словарь, «обучаясь» на входных текстовых данных, а затем вычисляет векторное представление слов. Векторное представление основывается на контекстной близости: слова, встречающиеся в тексте рядом с другими словами (а следовательно, имеющие схожий смысл), в векторном представлении будут иметь близкие координаты векторов-слов.
Более формально задача стоит так: максимизация косинусной близости между векторами слов (скалярное произведение векторов), которые появляются рядом друг с другом, и минимизация косинусной близости между векторами слов, которые не появляются друг рядом с другом. Рядом друг с другом в данном случае значит в близких контекстах.

- Контекст - в широком смысле – среда, в которой существует объект.

Word2Vec – как работает?

- Читается корпус, и рассчитывается встречаемость каждого слова в корпусе (т.е. количество раз, когда слово встретилось в корпусе — и так для каждого слова)
- Массив слов сортируется по частоте (слова сохраняются в хэш-таблице), и удаляются редкие слова
- Строится дерево Хаффмана. Дерево Хаффмана (Huffman Binary Tree) часто применяется для кодирования словаря — это значительно снижает вычислительную и временную сложность алгоритма.
- Из корпуса читается т.н. субпредложение (sub-sentence) и проводится субсэмплирование наиболее частотных слов (sub-sampling). Субпредложение — это некий базовый элемент корпуса, обычно — просто предложение, но это может быть и абзац, например, или даже целая статья. Субсэмплирование — это процесс изъятия наиболее частотных слов из анализа, что ускоряет процесс обучения алгоритма и способствует значительному увеличению качества получающейся модели.

Дерево Хаффмана



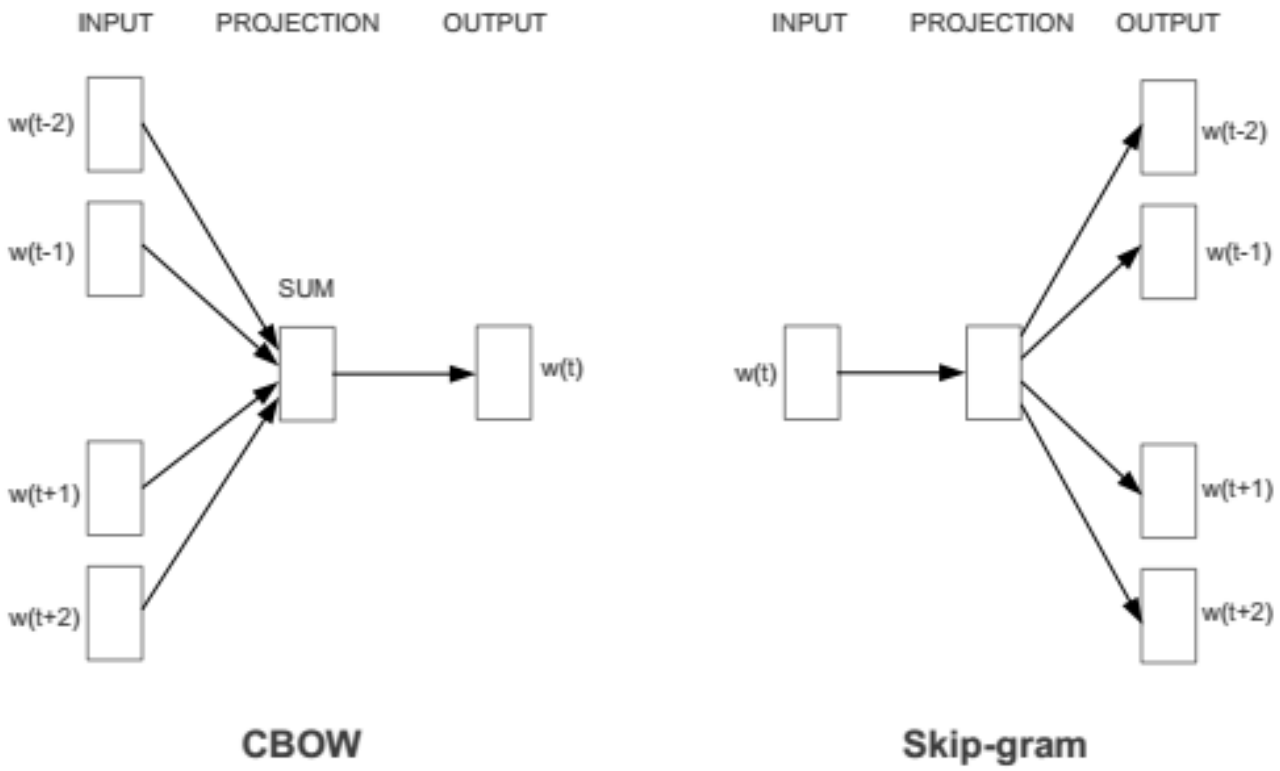
Word2Vec – как работает?

- По субпредложению проходим окном (размер окна задается алгоритму в качестве параметра). В данном случае под окном подразумевается максимальная дистанция между текущим и предсказываемым словом в предложении. То есть, если окно равно трем, то для предложения «The quick brown fox jumps over the lazy dog» анализ будет проходит внутри блока в три слова — для «The quick brown», «quick brown fox», «brown fox jumps» и т.д. Окно по умолчанию равно пяти, рекомендуемым значением является десять.

- Применяется нейросеть прямого распространения (Feedforward Neural Network) для получения векторных представлений слов.

Word2Vec (2)

Существует две архитектуры нейросети - CBOW (Continuous Bag Of Words) и Skip-gram, которые описывают, как именно нейросеть «учится» на данных и «запоминает» представления слов. Принципы у обоих архитектур разные. Принцип работы CBOW — предсказывание слова при данном контексте, а Skip-gram наоборот — предсказывается контекст при данном слове.



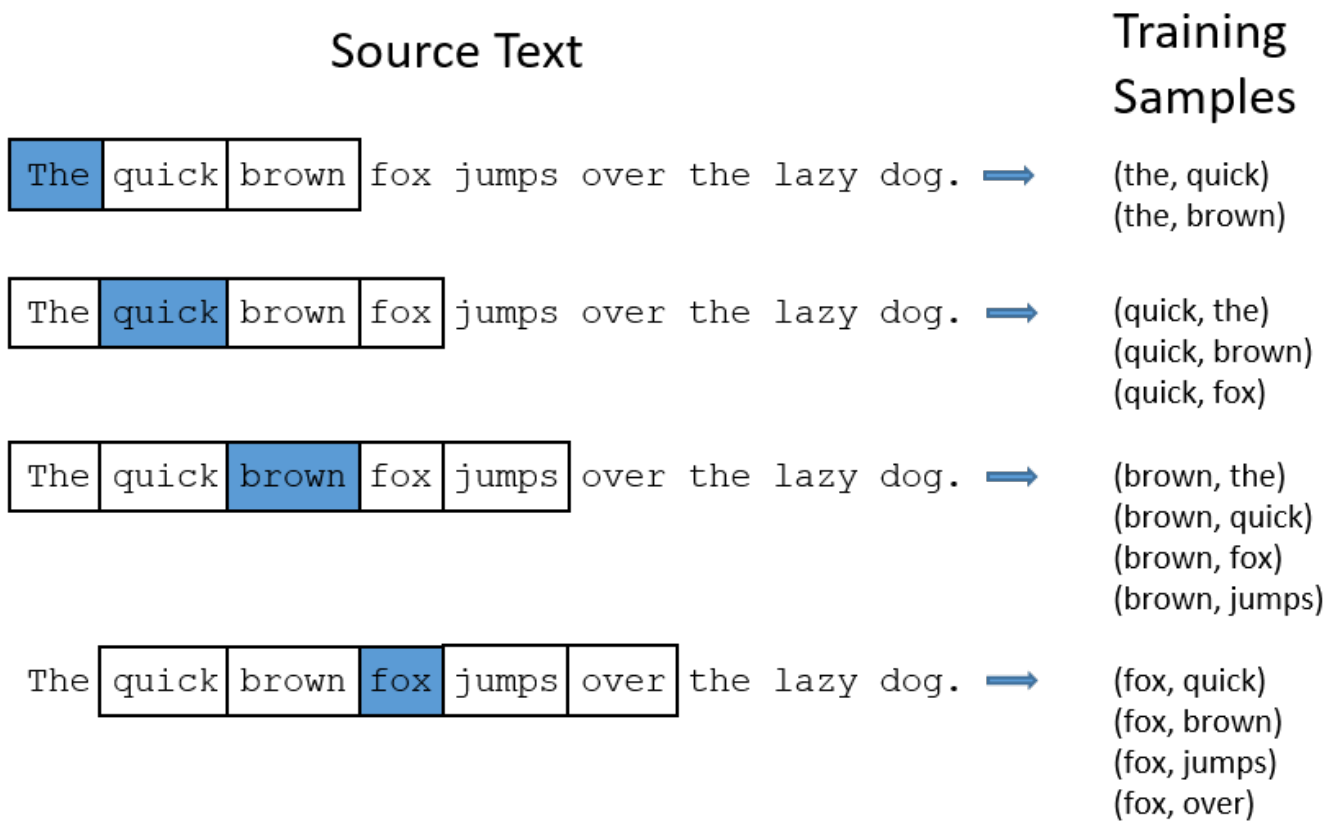
Skip-gram

Мы обучим нейронную сеть следующим действиям. Возьмем определенное слово в середине предложения (вводное слово, input word). Настроенная сеть должна выдать для каждого слова в нашем словаре вероятности того, что оно будет находиться «рядом» с вводным.

Полученная вероятность показывает то, насколько вероятно каждое слово из словаря будет находится в пределах размера окна с вводным словом.

Например, если вы задали обученной сети входное слово “моторное", то для таких слов, как “масло” и “двигатель”, полученная вероятность будет намного выше, чем для несвязанных слов, например, “яблоко” и “стул”.

Обучать сеть мы будем, подавая пары слов из обучающей выборки.
В приведенном примере показаны некоторые примеры таких обучающих пар. Слово, выделенной синим является вводных словом, размер окна = 2



GloVe – Global Vectors for Word Representation

1) Обучение модели на отдельном локальном контексте плохо использует глобальную статистику корпуса. Первый шаг для преодоления этого ограничения — создание глобальной матрицы **X**, где каждый элемент **i,j** подсчитывает количество упоминаний слова **j** в контексте слова **i**.
2) Требуется матрица совместного вхождения [M x M], откуда можно напрямую извлекать определённые аспекты значений.

I love NLP
I love training models

	I	Love	NLP	Training	models
I	-	2	0	0	0
Love	2	-	1	1	0
NLP	0	1	-	0	0
Training	0	1	0	-	1
models	0	0	0	1	-

https://nlp.stanford.edu/projects/glove/

GloVe – Global Vectors for Word Representation

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Хотим найти функцию, которая бы для 3 векторов слов сохраняла отношение: $F(w_i, w_j, \tilde{w}_k) = \frac{P(k|i)}{P(k|j)}$

Преобразуется к функции минимизации ошибки по МНК: $J = \sum_{i,j=1}^{M} f(X_{ij})[w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij})]^2$

https://nlp.stanford.edu/projects/glove/

FastText

FastText исправляет совершенно другой недостаток Word2Vec: если обучение модели начинается с прямого кодирования одного D-мерного вектора, то игнорируется внутренняя структура слов.
Вместо прямого кодирования кодирования слов, изучающих словесные представления, fastText предлагает изучать N-граммы символов и представлять слова как сумму векторов N-грамм.
Например, при N=3:
«flower» -> [<fl, flo, low, owe, wer, er>] плюс специальная последовательность <flower>.

Таким образом, слово представлено его индексом в словаре слов и набором N-грамм, которые оно содержит.
Это простое улучшение позволяет разделить N-граммовые представления между словами и вычислить векторные представления слов, которых не было в корпусе обучения

https://github.com/facebookresearch/fastText

Примеры использования библиотеки gensim

```
# build vocabulary and train model
    model = gensim.models.Word2Vec(
        documents,
        size=150,
        window=10,
        min_count=2,
        workers=10)
    model.train(documents, total_examples=len(documents), epochs=10)
```

```
# similarity between two different words
model.wv.similarity(w1="dirty",w2="smelly")
```

```
0.76181122646029453
```

```
# similarity between two identical words
model.wv.similarity(w1="dirty",w2="dirty")
```

```
1.0000000000000002
```

```
# similarity between two unrelated words
model.wv.similarity(w1="dirty",w2="clean")
```

```
0.25355593501920781
```

```
# look up top 6 words similar to 'france'
w1 = ["france"]
model.wv.most_similar (positive=w1,topn=6)
```

```
[('canada', 0.6603403091430664),
 ('germany', 0.6510637998580933),
 ('spain', 0.6431018114089966),
 ('barcelona', 0.61174076795578),
 ('mexico', 0.6070996522903442),
 ('rome', 0.6065913438796997)]
```

```
w1 = "dirty"
model.wv.most_similar (positive=w1)
```

```
[('filthy', 0.871721625328064),
 ('stained', 0.7922376990318298),
 ('unclean', 0.7915753126144409),
 ('dusty', 0.7772612571716309),
 ('smelly', 0.7618112564086914),
 ('grubby', 0.7483716011047363),
 ('dingy', 0.7330487966537476),
 ('gross', 0.7239381074905396),
 ('grimy', 0.7228356599807739),
 ('disgusting', 0.7213647365570068)]
```