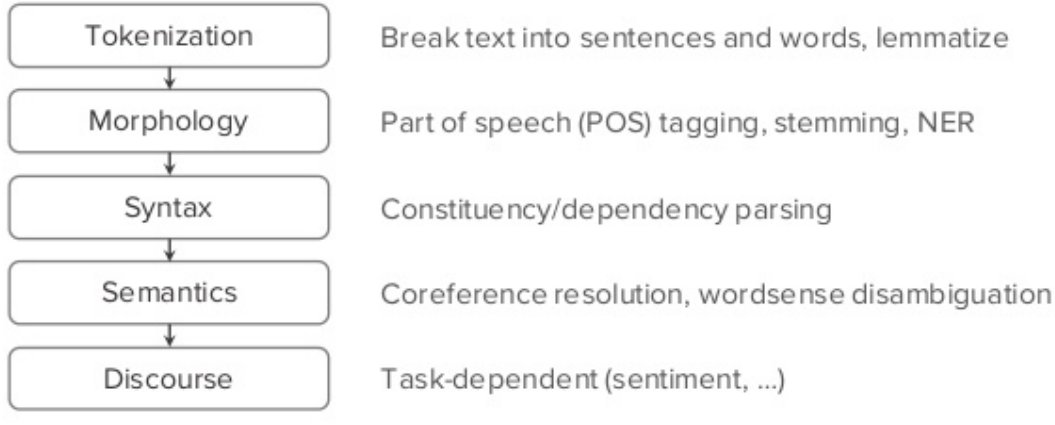
Введение в NLP

Общее направление искусственного интеллекта и математической лингвистики. Оно изучает проблемы компьютерного анализа и синтеза естественных языков. Применительно к искусственному интеллекту анализ означает понимание языка, а синтез — генерацию грамотного текста. Решение этих проблем будет означать создание более удобной формы взаимодействия компьютера и человека.

- Распознавание речи
- Анализ текста
  - Извлечение информации
  - Информационный поиск
  - Анализ высказываний
  - Анализ тональности текста
  - Вопросно-ответные системы
- Генерирование текста
- Синтез речи
- Машинный перевод
- Автоматическое реферирование, аннотирование или упрощение текста

“Classical” NLP Pipeline



Переход из символьного представления в непрерывное(embedding). Семантически близкие должны быть ближе.

embedding

Bag of words

Цель: поставить в соответствие слову вектор.

Самый простой подход -- Bag of words.

1. собираем и нумеруем все слова из корпуса, получаем словарь длины n.
2. i-му слову из словаря соответствует вектор длины n с 1 на i-й позиции и 0 на остальных.

Плюсы:

- всё понятно и легко реализовать

Минусы:

- на больших объёмах данных получаем векторы огромной размерности
- не позволяет сравнивать слова

Матрица терм-терм

1. собираем и нумеруем все слова из корпуса, получаем словарь длины n.
2. i-му слову из словаря сопоставляем вектор длины n, на j-й позиции которого стоит число появлений i-го и j-го слов вместе в окне некоторой ширины w.

Плюсы:

- всё ещё довольно просто
- позволяет как-то "сравнивать" слова (например, с помощью косинусного расстояния)

Минусы:

- всё ещё имеем на больших объёмах данных векторы огромной размерности с кучей нулей
- есть слова, которые встречаются постоянно повсюду (предлоги, частицы и пр.)

TF-IDF

Векторизация слов на основании документов, которые их содержат.

1. Для слова w и текста T считаем его частоту в данном тексте (term frequency):
   $tf(w, T) = \begin{cases} 1 + \log_{10} count(w, T) & count(w, T) > 0 \\ 0 & otherwise \end{cases}$, где $count(w, T)$ -- число появлений слова w в тексте T.
2. Для слова w считаем его обратную частоту по всем текстам (inverse document frequency):
   $idf(w) = \log_{10} \frac{N}{df(w)}$, где N -- число текстов в корпусе, $df(w)$ -- число текстов корпуса, в которых встречается слово w.
3. $tfidf(w, T) = tf(w, T) \cdot idf(w)$.

TF-IDF

Слова, которые встречаются часто везде, получают меньший вес, чем другие.

С помощью такого подхода можно, например, улучшить эвристику с матрицей терм-терм: слову w сопоставляем вектор, на j-й позиции которого стоит не число случаев, когда j-е слово встречается в некотором окне со словом w, а сумма значений $tfidf(w_j, T)$ по всем таким случаям (T -- документ, в котором эти слова встретились).

Плюсы:

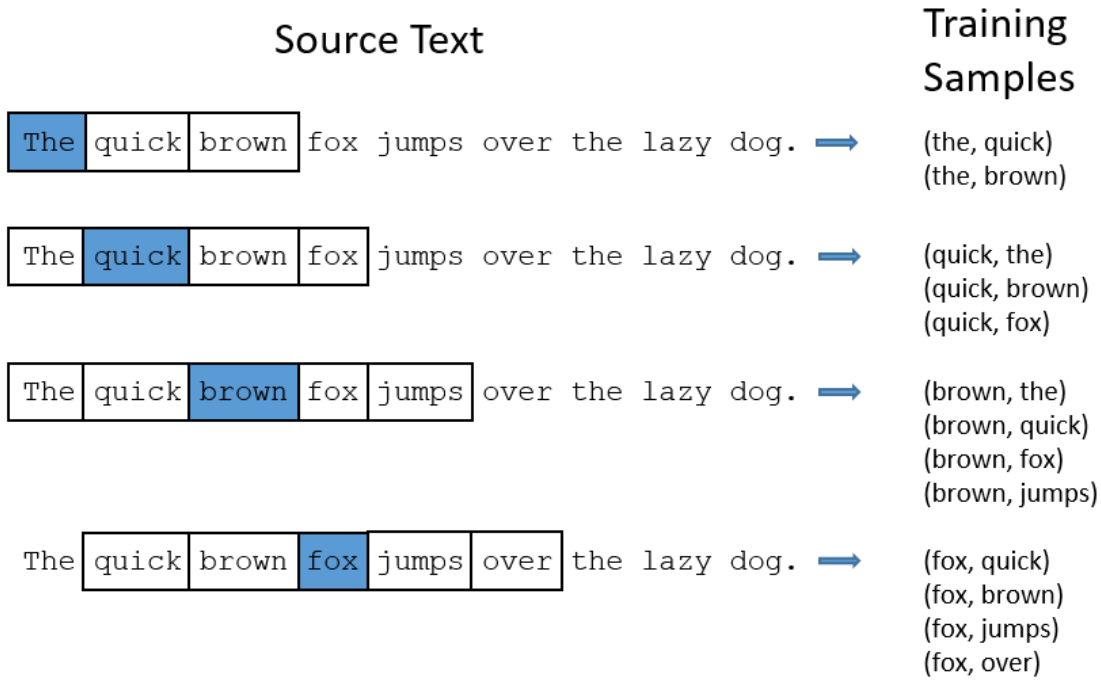
- интуитивно кажется, что слова при данном подходе сравниваются лучше, чем в прошлых вариантах (схожие слова имеют схожие контексты и схожую $tfidf$-важность в документах)

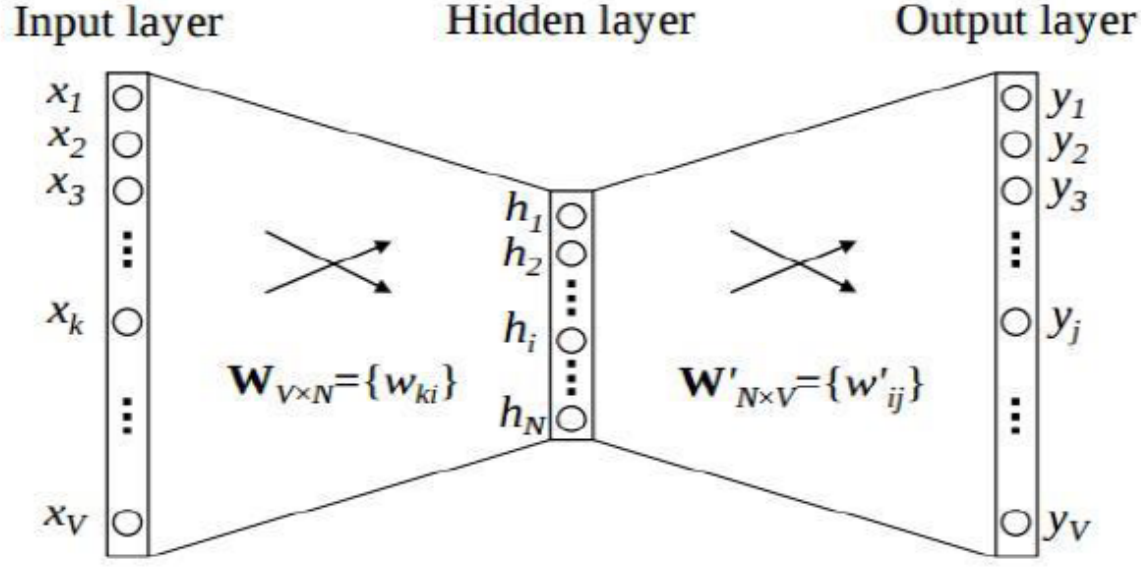
Минусы:

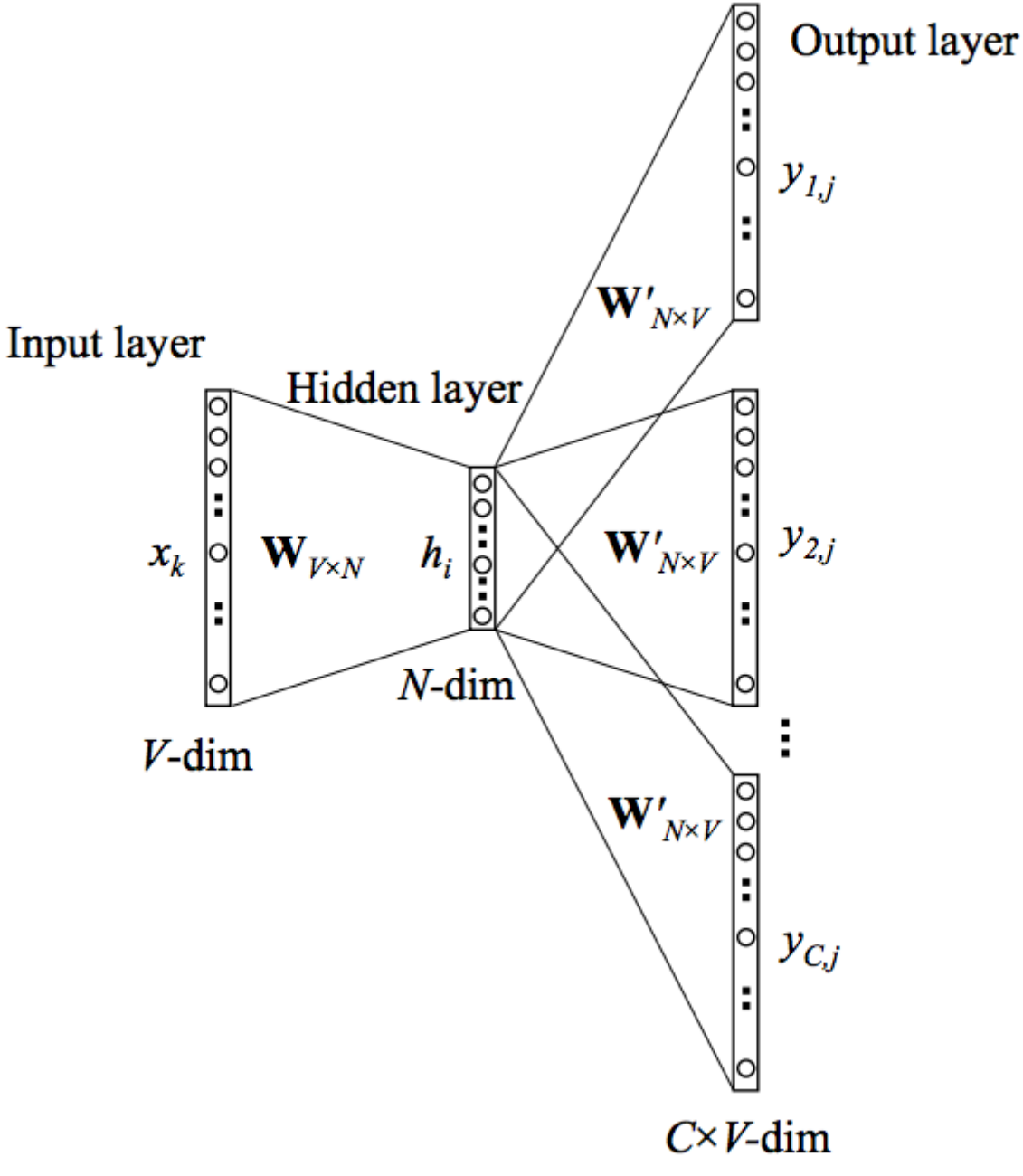
- всё ещё имеем на больших объёмах данных векторы огромной размерности с кучей нулей

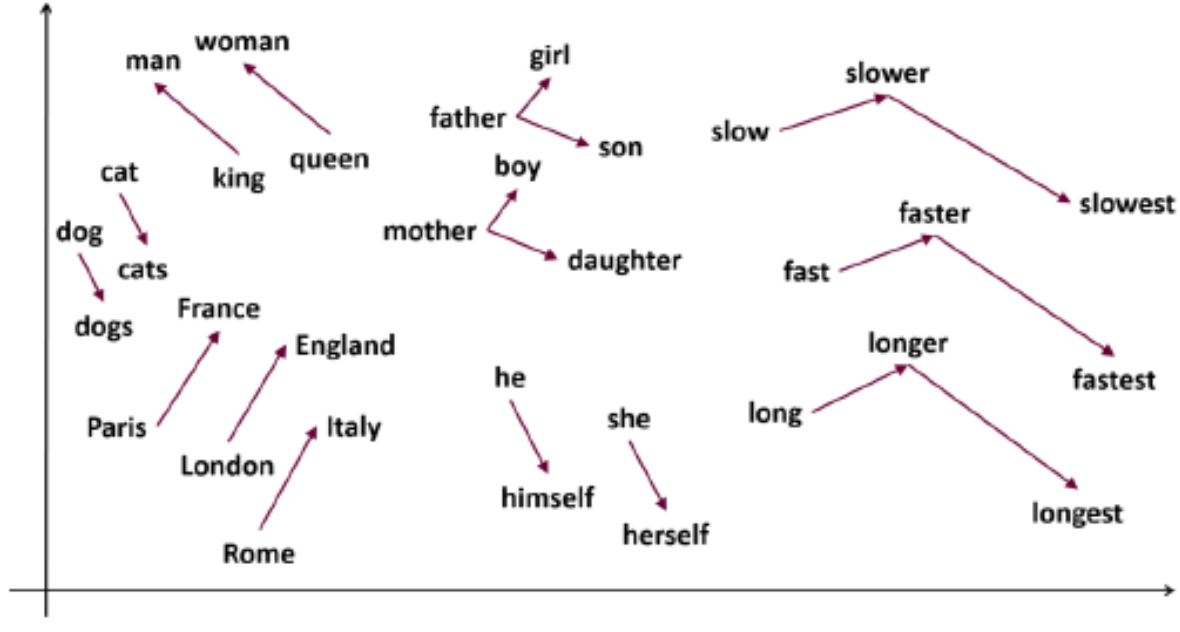
skip-gram word2vec

Все эти подходы страдают от одной и той же проблемы: при увеличении размера корпуса, растёт размерность. В худшем случае получаем десятки тысяч текстов и сотни тысяч слов, и по каждой паре <слово-текст> или <слово-слово> надо что-то считать.


Source Text
Training Samples
The quick brown fox jumps over the lazy dog.
(the, quick)
(the, brown)
The quick brown fox jumps over the lazy dog.
(quick, the)
(quick, brown)
(quick, fox)
The quick brown fox jumps over the lazy dog.
(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)
The quick brown fox jumps over the lazy dog.
(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)


Input layer
Hidden layer
Output layer
x_1
x_2
x_3
x_k
x_V
h_1
h_2
h_i
h_N
y_1
y_2
y_3
y_j
y_V
$\mathbf{W}_{V\times N}=\{w_{ki}\}$
$\mathbf{W}'_{N\times V}=\{w'_{ij}\}$


Output layer
$y_{1,j}$
$\mathbf{W}'_{N\times V}$
Input layer
Hidden layer
x_k
$\mathbf{W}_{V\times N}$
h_i
$\mathbf{W}'_{N\times V}$
$y_{2,j}$
N-dim
V-dim
$\mathbf{W}'_{N\times V}$
$y_{C,j}$
$C\times V$-dim


man
woman
king
queen
cat
cats
dog
dogs
France
England
Paris
London
Italy
Rome
girl
father
son
boy
mother
daughter
he
himself
she
herself
slow
slower
slowest
fast
faster
fastest
long
longer
longest

Векторизация текстов

Если есть векторизатор для слов, то самая простая идея --- посчитать, например, среднее всех слов в предложении.

Чуть менее тривиальная эвристика --- составлять векторы текстов из $tfidf$-весов слов из этих текстов: тексту T сопоставлять вектор

$$((tfidf(w_1, T), \ldots, tfidf(w_n, T)),$$

где $\{w_1, \ldots, w_n\}$ --- весь словарь корпуса. Такой подход лучше, чем Bag of words, потому что выделяет важные слова.

Такая идея проста в реализации, но не очевидно, что полученный таким образом вектор всегда будет подчёркивать *смысл* текста. До какой-то степени, кажется, это работает неплохо для Bag of words и TF-IDF, но тоже ясно, что смысл текста не всегда можно передать списком использующихся в нём слов. Эффективность такого подхода гораздо менее очевидна для skip-gram.