Embeddings

Word2Vec, Glove, FastText

Shapkin Anton

Что такое эмбеддинг?

Эмбеддинг - представление какой-то сущности (картинки, текста, слова) в виде вектора. Впервые данный термин появился в работах специалистов в области NLP (обработка естественного языка).

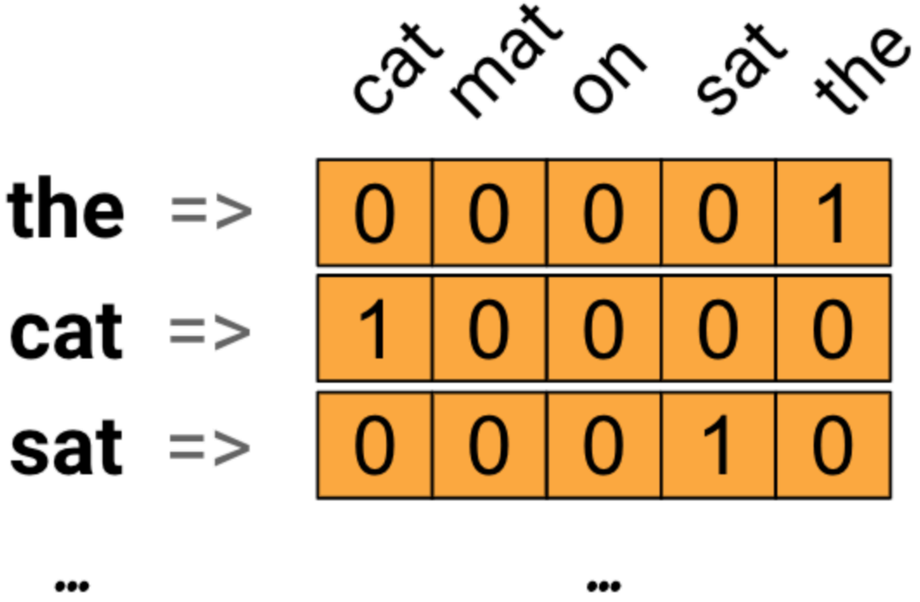
Natural Language Processing

Направление искусственного интеллекта и математической лингвистики, изучающая проблемы компьютерного анализа и синтеза текстов на естественных языках.

- Информационный поиск
- Машинный перевод
- Анализ текстов
- Распознавание речи
- ...

One-hot encoding

Пронумеруем слова в некотором словаре. Тогда, для i-го слова словаря: нулевой вектор размерности словаря с единицей на i-ом месте.



One-hot encoding. Недостатки

- Размер словаря (длинна вектора)
- Данный метод не отображает значения слов

Например, слово Москва так же «близко» к слову Екатеринбург, как к слову математика.

Я купил бутылку _____.

_____ - самый полезный напиток.

Рекомендовано пить 2.5 литра _____ в день.

Гипотеза локальности

Гипотеза: слово определяется по контексту. Слова часто встречающиеся в одном контексте будут похожи друг на друга.

Идея: вложить информацию о контексте слова в его векторное представление

Я купил бутылку _____.

_____ - самый полезный напиток.

Рекомендовано пить 2.5 литра _____ в день.

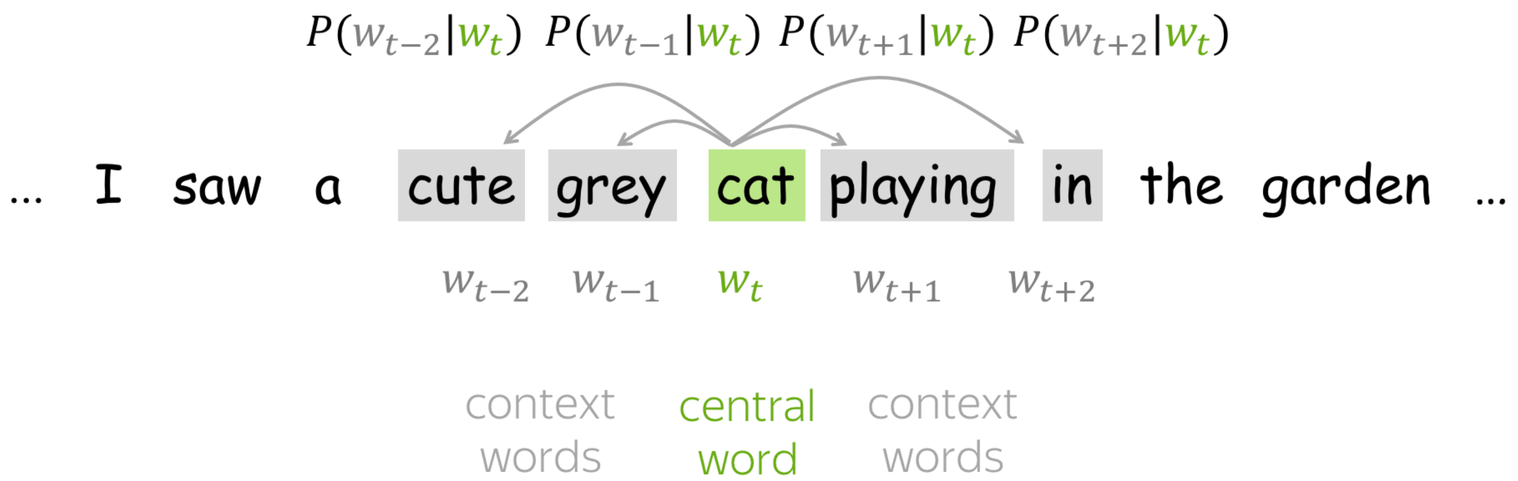
Word2Vec. Общий подход

Идея: вложить информацию о контексте слова в его векторное представление

Выбираем N - размерность наших эмбеддингов

Скользящим окном проходим по корпусу. На каждом шаге есть центральное слово и контекстные слова

Обновляем веса с целью увеличить правдоподобие.



Word2Vec

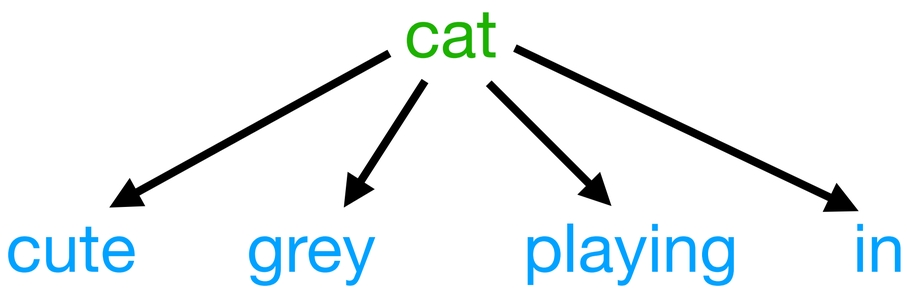
Существует 2 варианта word2vec:

- Continuous Bag Of Word (CBOW) - максимизируем правдоподобие центрального слова в зависимости от контекста
- Skip-gram - максимизируем правдоподобие контекста на основе центрального слова

Word2Vec. Skip-gram

Максимизируем правдоподобие контекста на основе центрального слова

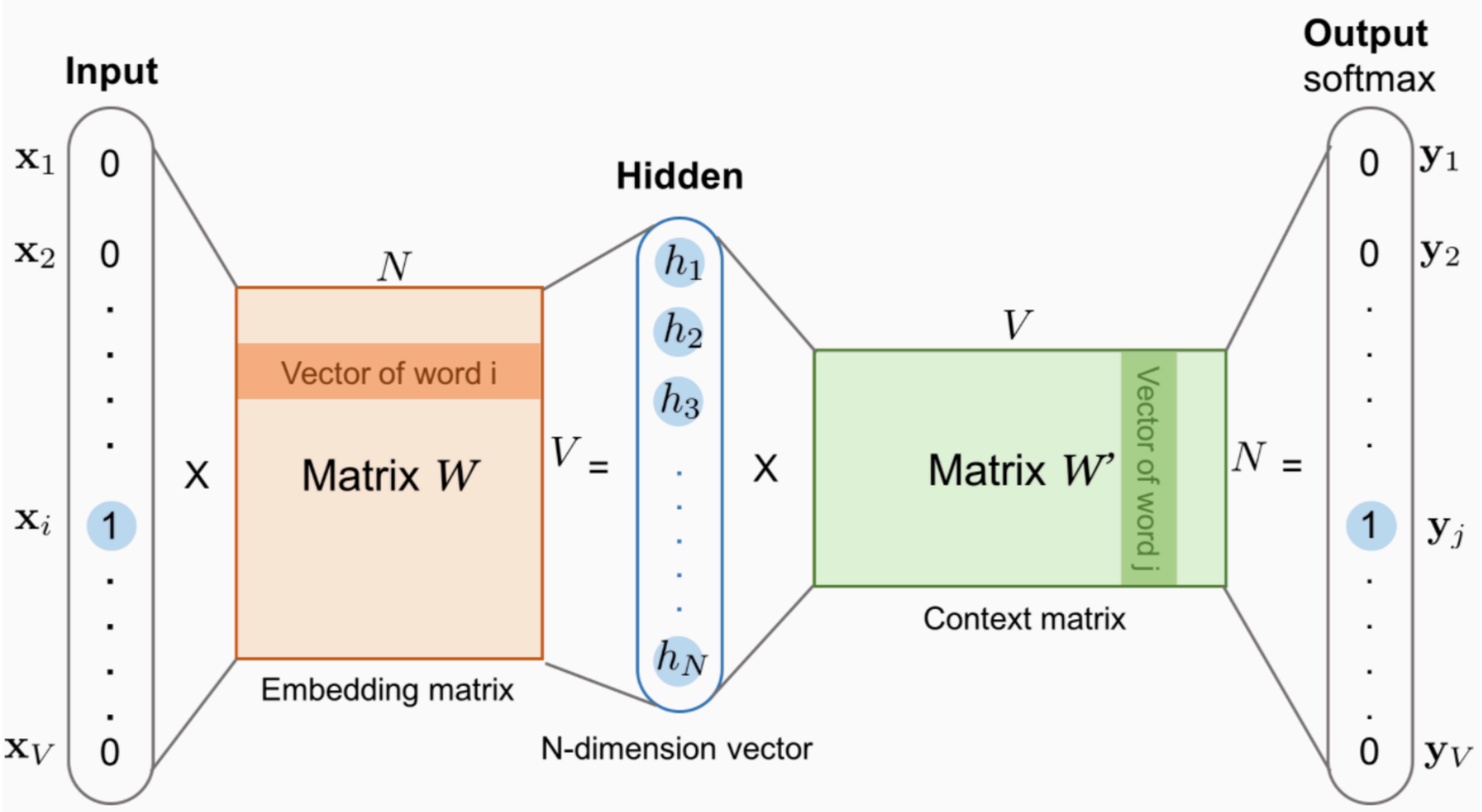
… I saw a cute grey cat playing in the garden …



Минимизируем $L = -\ln p(w_{Output,1}, \ldots, w_{Output,C} \mid w_{Input})$

На вход нашей сети поступают one-hot encoded вектор центрального слова.

Word2Vec. Skip-gram

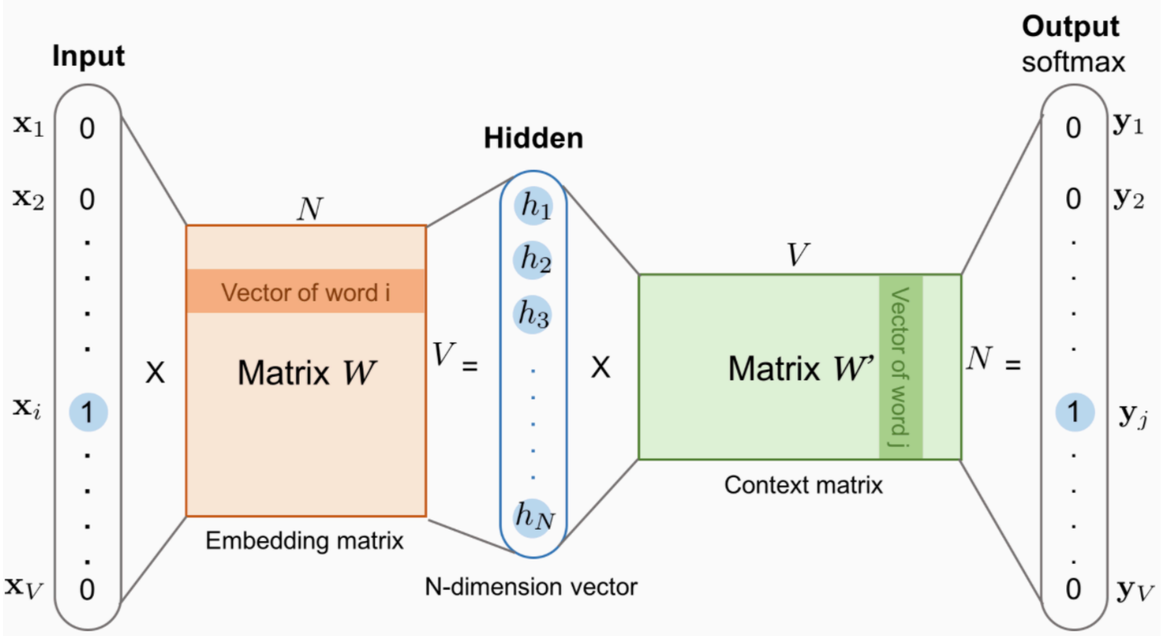


Word2Vec. Skip-gram

Матрица W содержит векторное представление слов как центральных (v).
Матрица W' содержит векторное представление слов как контекстных (u).

$$h = W^T_{(i,\cdot)} = v^T_{w_I}$$

$$y_j = p(w_j | w_{Input}) = \frac{exp(u^T_{w_j} h)}{\sum_{i=1}^{V} exp(u^T_{w_i} h)}$$



Заметим, что v_w и u_w - два векторных представления одного и того же слова w.

Word2Vec. Skip-gram

$$h = W^T_{(i,\cdot)} = v^T_{w_I} \qquad y_j = p(w_j \mid w_{Input}) = \frac{exp(u^T_{w_j} h)}{\sum_{i=1}^{V} exp(u^T_{w_i} h)}$$

Функция потерь

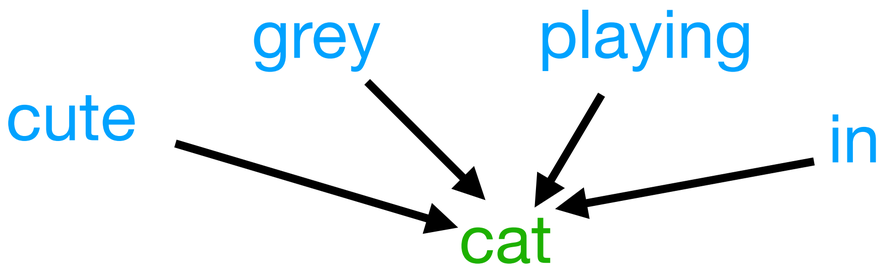
$$L = -\ln p(w_{Output,1}, \ldots, w_{Output,C} \mid w_{Input}) = -\ln \prod_{c=1}^{C} p(w_{Output,c} \mid w_{Input})$$

Модель содержит $2 \cdot V \cdot N$ параметров (по 2 векторных представления для каждого слова из словаря).

Word2Vec. CBOW

Максимизируем правдоподобие центрального слова в зависимости от контекста

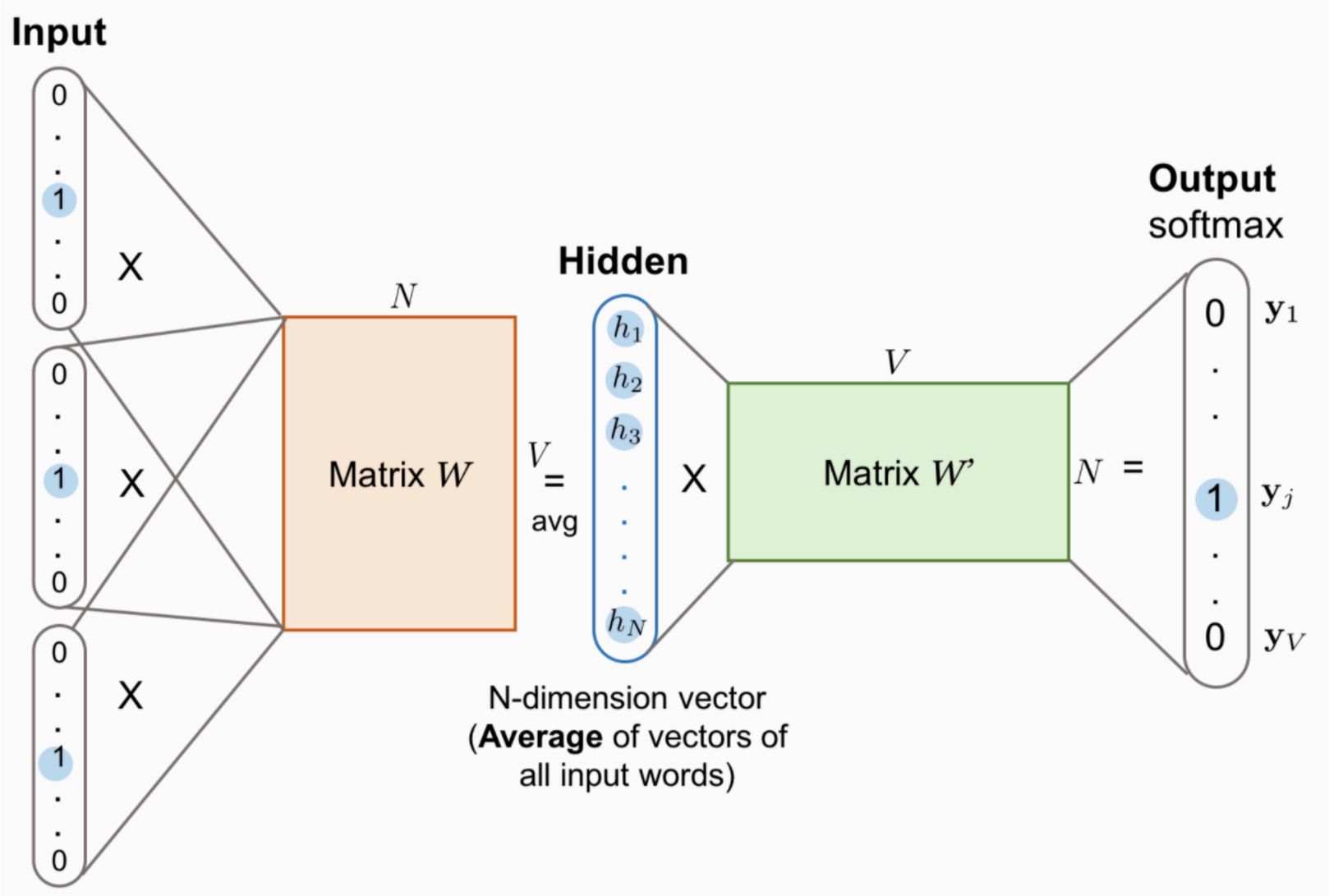
… I saw a cute grey cat playing in the garden …



Минимизируем $L = -\ln p(w_{Output} \mid w_{Input,1}, \ldots, w_{Input,C})$

На вход нашей сети поступают one-hot encoded вектора контекстных слов для текущего центрального слова.

Word2Vec. CBOW



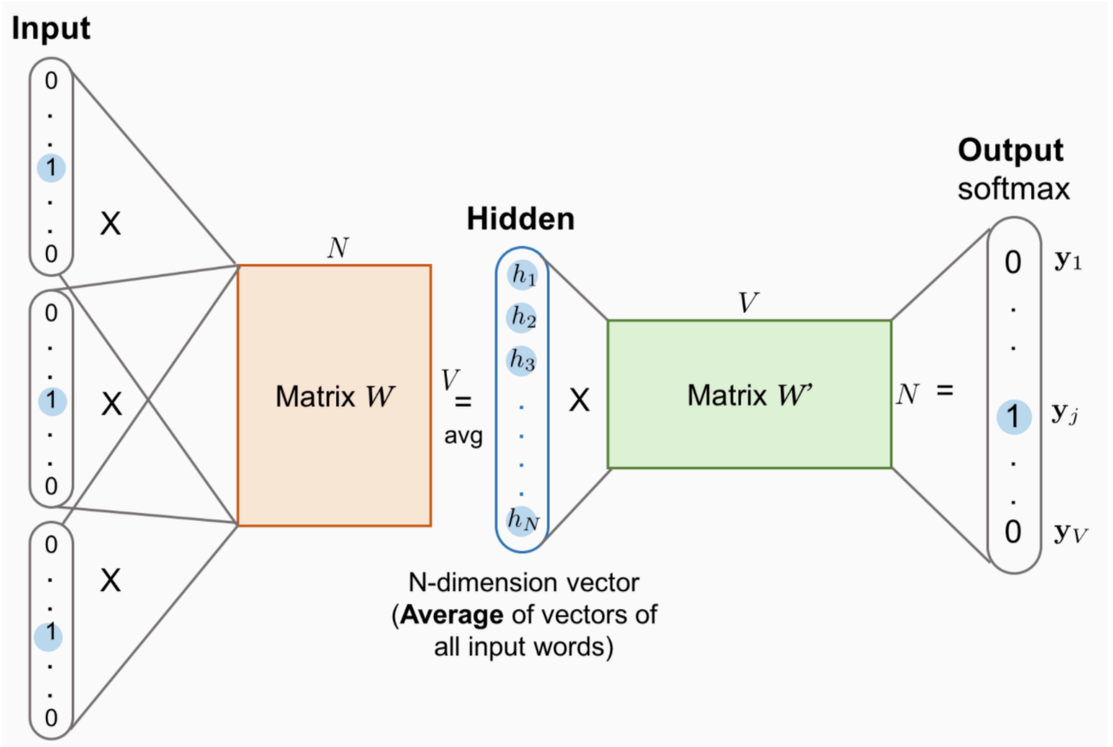
Word2Vec. CBOW

Матрица W содержит векторное представление слов как контекстных (u).

Матрица W' содержит векторное представление слов как центральных (v).

$$h = \frac{1}{C} W^T (x_1 + \ldots + x_C) = \frac{1}{C} (u_{w_1} + \ldots + u_{w_C})$$

$$y_j = p(w_j \mid w_{Input}) = \frac{exp(v_{w_j}^T h)}{\sum_{i=1}^{V} exp(v_{w_i}^T h)}$$



Заметим, что v_w и u_w - два векторных представления одного и того же слова w.

Word2Vec. CBOW

$$h = \frac{1}{C} W^T (x_1 + \ldots + x_C) = \frac{1}{C} \left(u_{w_1} + \ldots + u_{w_C} \right)$$

$$y_j = p(w_j \mid w_{Input}) = \frac{exp(v_{w_j}^T h)}{\sum_{i=1}^{V} exp(v_{w_i}^T h)}$$

Функция потерь

$$L = -\ln p(w_{Output} \mid w_{Input,1}, \ldots, w_{Input,C})$$

Модель содержит $2 \cdot V \cdot N$ параметров (по 2 векторных представления для каждого слова из словаря).

Word2Vec

Skip-gram

Хорошо работает с небольшим количеством данных. Хорошие эмбеддинги для редких слов.

CBOW

Быстрее, чем Skip-gram. Лучше производит эмбеддинги для часто-встречаемых слов.

Зачем 2 векторных представления?

Когда центральное и контекстное представления имеют разные векторы, функция потерь линейна по соответствующим параметрам. Градиенты считать легче.

Word2Vec

Преимущества

- Не требует большой вычислительной мощности
- Инструмент не нуждается в предобработке данных
- Модели, натренированные на большом количестве текстов показывают удивительные результаты: алгебраические операции на векторах отображают семантические операции.

Недостатки

- Хорошо работает на словах и словосочетаниях, однако показывает не удовлетворительные результаты на длинных текстах
- Не могут быть представлены слова, которые не встречались в обучающей выборке (в ее словаре)
- The car pollute the air.

w2v не может сказать, является ли the каким-то особенным контекстом или же это просто шум.

GloVe (Global vectors)

Основная идея: семантические отношения между словами можно вывести из матрицы совпадений

The cat sat on the mat

X_{ij} - содержит информацию о том, как часто j-ое слово встречалось с i-ым.

$$P_{ij} = P(j \mid i) = \frac{X_{ij}}{X_i} = \frac{X_{ij}}{\sum_k X_{ik}}$$

	the	cat	sat	on	mat
the	0	1	0	1	1
cat	1	0	1	0	0
sat	0	1	0	1	0
on	1	0	1	0	0
mat	1	0	0	0	0

GloVe

Как посчитать семантическую близость между словами?

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

Если слово k …

- Близко к слову ice, но далеко от слова steam $P_{ik}/P_{jk} > 1$
- Близко к слову steam, но далеко от слова ice $P_{ik}/P_{jk} < 1$
- Близко или далеко от этих слов одновременно $P_{ik}/P_{jk} \sim 1$

GloVe

Предположим, существует функция F, принимающая на вход вектора для слов i, j и k, такая, что:

1) Линейность: $F((w_i - w_j)^T u_k) = P_{ik}/P_{jk}$

2) Симметрия: $F(w_i^T u_k - w_j^T u_k) = F(w_i^T u_k)/F(w_j^T u_k)$ $\longrightarrow$ $F(w_i^T u_k) = P_{ik}$, $F(x) = \exp(x)$

$$w_i^T u_k = \ln P_{ik} = \ln X_{ik} - \ln X_i$$

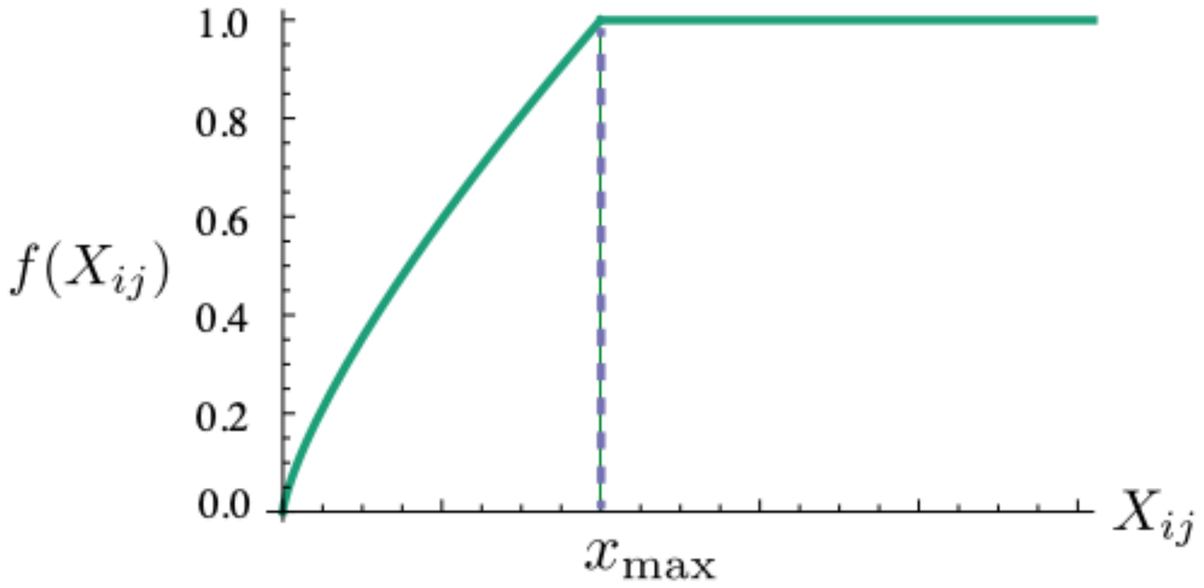
$$w_i^T u_k + b_i + b_k = \ln X_{ik}$$

GloVe

Функция потерь

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T u_j + b_i + b_k - \ln X_{ik})^2$$

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$



FastText

Модели word2vec и glove показывают очень хорошие результаты. Однако, данные подходы не учитывают символы слова, а значит не учитываются приставки и суффиксы, корни и составные слова.

Каждое слово можно представить в виде n-gram:

FastText

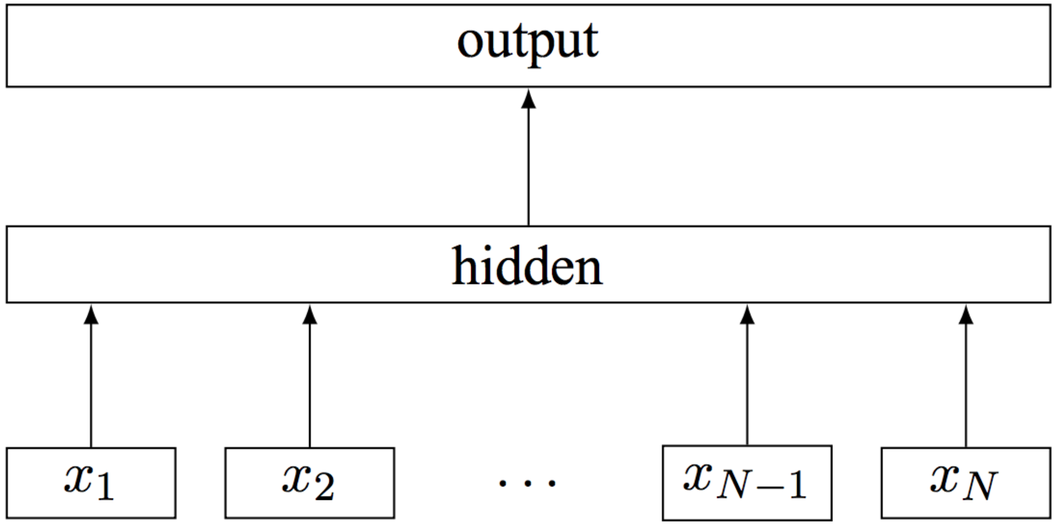


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

FastText

Функция потерь

$$L = -\ln p(w_{Output,1}, \ldots, w_{Output,C} \mid w_{Input}) = -\ln \prod_{c=1}^{C} p(w_{Output,c} \mid w_{Input})$$

$$p(w_c \mid w_t) = \frac{e^{s(w_t,\, w_c)}}{\sum_{j=1}^{W} e^{s(w_t,\, j)}}$$

$$s(c, w) = \sum_{g \in G_w} z_g^T v_c$$

где v_c - векторное представление контекстного слова

FastText

Преимущества

- Хорошие представления для редких слов и морфологически богатых языков
- Инструмент не нуждается в предобработке данных

Недостатки

- Работает не так уж и быстро (не быстрее w2v)

Источники

- https://www.aclweb.org/anthology/D14-1162.pdf
- https://papers.nips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf
- https://arxiv.org/pdf/1411.2738.pdf
- https://lena-voita.github.io/nlp_course/word_embeddings.html#glove
- https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html
- https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010
- https://arxiv.org/pdf/1607.04606.pdf

Вопросы

Чем отличаются варианты Word2Vec: Skip-gram и CBOW?

Пусть имеется корпус текстов со словарем 10000 слов. Мы хотим найти векторные представления для слов с помощью Word2Vec. В качестве размерности векторов возьмем N=50, длинна скользящего окна = 2*2. Сколько параметров будет у нашей модели?

В чем глобальное отличие Word2Vec и GloVe?

Зачем в функции потерь GloVe нужна $f(X_{ij})$?