Текстовая классификация.

Введение.

Алексей Андреевич Сорокин
Yandex Research,
МГУ, отделение теоретической и прикладной лингвистики.

Школа РАИИ 2021
лекция 1.

Основные задачи

- Текстовая классификация
  - Определение темы или жанра текста.
  - Анализ тональности.
  - Определение автора и его характеристик (пол, возраст).

Основные задачи

- Текстовая классификация
  - Определение темы или жанра текста.
  - Анализ тональности.
  - Определение автора и его характеристик (пол, возраст).
- Автоматический перевод.
- Исправление опечаток.

Основные задачи

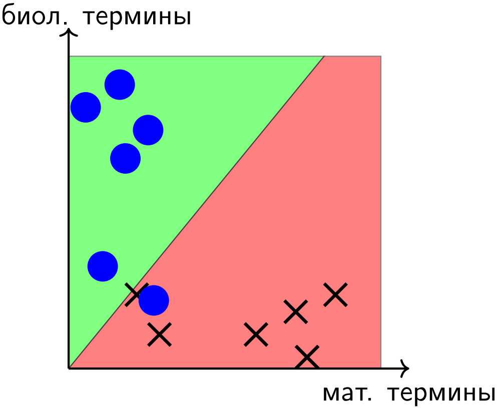
- Текстовая классификация
  - Определение темы или жанра текста.
  - Анализ тональности.
  - Определение автора и его характеристик (пол, возраст).
- Автоматический перевод.
- Исправление опечаток.
- Информационный поиск.
- Распознавание именованных сущностей.

Основные задачи

- Текстовая классификация
  - Определение темы или жанра текста.
  - Анализ тональности.
  - Определение автора и его характеристик (пол, возраст).
- Автоматический перевод.
- Исправление опечаток.
- Информационный поиск.
- Распознавание именованных сущностей.
- Задачи порождения текста:
  - Диалоговые системы.
  - Автоматическое реферирование.
- Вопросно-ответные системы.

Линейная классификация

Упрощённая тематическая классификация на 2 класса:



Линейный классификатор (случай 2 классов)

- Решающая функция:

$$
\begin{aligned}
h(x) = \langle w, x\rangle - w_0 \quad &- \quad \text{решающая функция,}\\
f(x) = \operatorname{sgn} h(x) \quad &- \quad \text{предсказанная метка класса,}\\
|h(x)| \quad &- \quad \text{расстояние от разделяющей поверхности}
\end{aligned}
$$

- Сравнение с эталоном:

$$
\begin{aligned}
y(x) \in \{-1, 1\} \quad &- \quad \text{метка класса,}\\
(y(x)f(x) > 0) \quad &- \quad \text{условие правильности классификации,}\\
\max(-y(x)f(x), 0) \quad &- \quad \text{ошибка классификации}
\end{aligned}
$$

- Введём $M_w(x, y(x)) = y(x)f(x) = y(x)(\langle w, x\rangle - w_0)$ – отступ объекта x.

$$
M_w(x, y(x)) > 0 \leftrightarrow x \text{ классифицируется правильно.}
$$

Линейный классификатор (случай 2 классов)

- Задача линейного классификатора – подобрать разделяющую поверхность
$$\langle w, x \rangle - w_0 = 0$$
наилучшим образом.
- Самый наивный алгоритм – пытаться улучшить отступ у тех объектов, у кого он отрицателен.

Линейный классификатор (случай 2 классов)

- Задача линейного классификатора – подобрать разделяющую поверхность
$$\langle w, x \rangle - w_0 = 0$$
наилучшим образом.
- Самый наивный алгоритм – пытаться улучшить отступ у тех объектов, у кого он отрицателен.
- Это можно сделать с помощью персептрона Розенблатта ($\eta > 0$ – темп обучения):
$$w \leftarrow w + \eta xy, \quad y \in \{-1, 1\}$$

Линейный классификатор (случай 2 классов)

- Задача линейного классификатора – подобрать разделяющую поверхность
$$\langle w, x\rangle - w_0 = 0$$
наилучшим образом.
- Самый наивный алгоритм – пытаться улучшить отступ у тех объектов, у кого он отрицателен.
- Это можно сделать с помощью персептрона Розенблатта ($\eta > 0$ – темп обучения):
$$w \leftarrow w + \eta xy, \quad y \in \{-1, 1\}$$
- Действительно,
$$M' = y\langle w + \eta xy, x\rangle = y\langle w, x\rangle + \eta(x, x)y^2 = M(x) + \eta(x, x)y^2$$

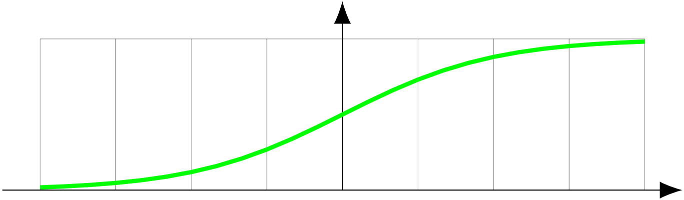
Вероятность классов в линейном классификаторе

- Пусть линейный классификатор имеет вид

$$\begin{aligned} h(x) &= \operatorname{sgn} a(x) \\ a(x) &= \langle w, x \rangle - w_0 \end{aligned}$$

- Тогда вероятность положительного класса равна:

$$p(y=1|x) = \sigma(a(x)) = \frac{e^{a(x)}}{e^{a(x)}+1} = \frac{1}{e^{-a(x)}+1}$$



- Сигмоидная функция переводит $(-\infty; +\infty)$ в $(0;1)$ (вероятность).
- С этого момента $y \in \{0, 1\}$.

Метод максимального правдоподобия

- Метод максимального правдоподобия максимизирует вероятность обучающей выборки.

$$\prod_{(x,y)\in X} yp(y=1|x)+(1-y)p(y=0|x)\to \max$$

- Это эквивалентно минимизации суммы отрицательных логарифмов.

$$L(X,Y)=\sum_{(x,y)\in X} -\ln\left(yp(y=1|x)+(1-y)p(y=0|x)\right)\to \min$$

- Для минимизации используют градиентный спуск:

$$w\leftarrow w-\eta\frac{\partial L(X,Y)}{\partial w}$$

- На практике применяют стохастический градиентный спуск, минимизируя штраф на конкретном объекте.

$$w\leftarrow w-\eta\frac{\partial L(x,y)}{\partial w}$$

Вероятность классов в линейном классификаторе

- Пусть линейный классификатор имеет вид

$$
\begin{aligned}
p(x) &= \sigma(a(x)) \\
a(x) &= \langle w, x\rangle - w_0 \\
\sigma(z) &= \tfrac{1}{1+e^{-z}}
\end{aligned}
$$

- Рассмотрим шаг градиентного спуска в случае $y = 1$:

$$L(x, y) = -\ln p(x) = -\ln \sigma(a(x))$$

$$\frac{\partial L}{\partial w} = -p(x = 0)x.$$

Вероятность классов в линейном классификаторе

- Пусть линейный классификатор имеет вид

$$
\begin{aligned}
p(x) &= \sigma(a(x)) \\
a(x) &= \langle w, x \rangle - w_0 \\
\sigma(z) &= \tfrac{1}{1+e^{-z}}
\end{aligned}
$$

- Рассмотрим шаг градиентного спуска в случае $y = 1$:

$$
\begin{aligned}
L(x, y) &= -\ln p(x) = -\ln \sigma(a(x)) \\
\frac{\partial L}{\partial w} &= -\frac{\partial \sigma(z)}{\partial z} * \frac{1}{\sigma(z)}\Big|_{z=a(x)} \frac{\partial a(x)}{\partial w} \\
\frac{\partial L}{\partial w} &= -p(x = 0)x.
\end{aligned}
$$

Вероятность классов в линейном классификаторе

- Пусть линейный классификатор имеет вид

$$
\begin{aligned}
p(x) &= \sigma(a(x)) \\
a(x) &= \langle w, x \rangle - w_0 \\
\sigma(z) &= \tfrac{1}{1+e^{-z}}
\end{aligned}
$$

- Рассмотрим шаг градиентного спуска в случае $y = 1$:

$$
\begin{aligned}
L(x, y) &= -\ln p(x) = -\ln \sigma(a(x)) \\
\frac{\partial L}{\partial w} &= -\frac{\partial \sigma(z)}{\partial z} * \frac{1}{\sigma(z)}\Big|_{z=a(x)} \frac{\partial a(x)}{\partial w} \\
\frac{\partial L}{\partial w} &= -p(x = 0)x.
\end{aligned}
$$

- То есть шаг обновления весов

$$
w \leftarrow w + \eta * p(x = 0)x
$$

Вероятность классов в линейном классификаторе

- Шаг обновления весов:

$$
\begin{aligned}
w &\leftarrow w + \eta * p(x = 0)x, \quad y = 1, \\
w &\leftarrow w - \eta * p(x = 1)x, \quad y = 0.
\end{aligned}
$$

- То есть логистическая регрессия – это сглаженный вариант персептрона.

Вероятность текста

- Задача: понять, на каком языке написано слово *человек* (русский, болгарский, украинский, ...)
- Идея: посчитать $p_{ru}(человек)$, $p_{bg}(человек)$ и понять, какая больше.
- Самый простой способ подсчёта – униграммная модель:

$$p_{ru}(человек) = p_{ru}(ч)p_{ru}(е)\dots p_{ru}(к)$$

Вероятность текста

- Задача: понять, на каком языке написано слово *человек* (русский, болгарский, украинский, ...)
- Идея: посчитать $p_{ru}(\textit{человек})$, $p_{bg}(\textit{человек})$ и понять, какая больше.
- Самый простой способ подсчёта – униграммная модель:

$$p_{ru}(\textit{человек}) = p_{ru}(\textit{ч})p_{ru}(\textit{е}) \dots p_{ru}(\textit{к})$$

- Формально (по формуле Байеса):

$$\begin{aligned} g(\textit{человек}) &= \operatorname{argmax}_y p(y|\textit{человек}), \\ p(y|\textit{человек}) &= \frac{p(\textit{человек}|y)p(y)}{p(\textit{человек})}, \\ \operatorname{argmax}_y p(y|\textit{человек}) &= \operatorname{argmax}_y p(\textit{человек}|y)p(y) \end{aligned}$$

Наивный байесовский классификатор

- Наивный байесовский классификатор использует формулу

$$\begin{aligned} g(x) &= \text{argmax}_y\, p(y|x), \\ \text{argmax}_y\, p(y|x) &= \text{argmax}_y\, p(x|y)p(y), \\ p(x|y) &= p(x^1|y)\dots p(x^m|y) \end{aligned}$$

- Здесь $x^1, \dots, x^m$ — признаки объекта x, их число зависит от объекта (отдельные буквы)
- Они независимы.

Наивный байесовский классификатор

- Наивный байесовский классификатор использует формулу

$$\begin{aligned} g(x) &= \mathrm{argmax}_y\, p(y|x), \\ \mathrm{argmax}_y\, p(y|x) &= \mathrm{argmax}_y\, p(x|y)p(y), \\ p(x|y) &= p(x^1|y)\dots p(x^m|y) \end{aligned}$$

- Здесь $x^1, \dots, x^m$ — признаки объекта x, их число зависит от объекта (отдельные буквы)
- Они независимы.
- Можно считать, что у нас есть фиксированное число признаков $x_1, \dots, x_d$, каждый из них может встречаться в объекте несколько раз.
- Тогда

$$p(x|y) = p(x_1|y)^{m_1} \dots p(x_d|y)^{m_d}$$

Подсчёт вероятностей

- Как посчитать $p(x_i|y) = p_{iy}$?
- Можно взять вероятность признака x_i в обучающей выборке среди класса y.

$$p_{iy} = \frac{n_{iy}}{\sum\limits_i n_{iy}} = \frac{n_{iy}}{n_y}$$

Подсчёт вероятностей

- Как посчитать $p(x_i|y) = p_{iy}$?
- Можно взять вероятность признака x_i в обучающей выборке среди класса y.

$$p_{iy} = \frac{n_{iy}}{\sum\limits_i n_{iy}} = \frac{n_{iy}}{n_y}$$

- Чтобы не было нулевых вероятностей:

$$p_{iy} = \frac{n_{iy} + \alpha}{\sum\limits_i (n_{iy} + \alpha)} = \frac{n_{iy} + \alpha}{n_y + \alpha d}$$

- Аналогично, априорные вероятности классов:

$$p(y) = \frac{N_y}{N} = \frac{\text{число объектов класса } y}{\text{общее число объектов}}$$

Общая вероятность

- Получили формулы:

$$
\begin{aligned}
g(x) &= \operatorname{argmax}_y p(y|x), \\
p(y|x) &= p_y \prod_{i=1}^{d} p_{iy}^{m_i}
\end{aligned}
$$

Общая вероятность

- Получили формулы:

$$\begin{aligned} g(x) &= \operatorname{argmax}_y p(y|x), \\ p(y|x) &= p_y \prod_{i=1}^{d} p_{iy}^{m_i} \end{aligned}$$

- Логарифмируем:

$$g(x) = \operatorname{argmax}_y \left(\sum_{i=1}^{d} m_i \log p_{iy} + \log p_y\right)$$

Общая вероятность

- Получили формулы:

$$\begin{aligned} g(x) &= \mathrm{argmax}_y\, p(y|x), \\ p(y|x) &= p_y \prod_{i=1}^{d} p_{iy}^{m_i} \end{aligned}$$

- Логарифмируем:

$$g(x) = \mathrm{argmax}_y \Big(\sum_{i=1}^{d} m_i \log p_{iy} + \log p_y\Big)$$

- Если положить

$$\begin{aligned} w_y &= [\log p_{1y}, \ldots, \log p_{dy}], \\ b_y &= \log p_y, \end{aligned}$$

то снова

$$g(x) = \mathrm{argmax}_y \langle w_y, x\rangle + b_y$$

Наивный байесовский классификатор как линейный

- Наивный байесовский классификатор тоже строит линейную разделяющую поверхность.
- Он опирается на предположение о независимости элементов (слов, символов) исходного текста

Наивный байесовский классификатор как линейный

- Наивный байесовский классификатор тоже строит линейную разделяющую поверхность.
- Он опирается на предположение о независимости элементов (слов, символов) исходного текста
- Чаще всего это предположение неверно.
- Как следствие, наивный байесовский классификатор в таких случаях уступает логистической регрессии.

Недостатки линейной модели

- Линейный классификатор требует, чтобы классы разделялись плоскостью.
- Это будет зависеть от способа построения вектора по тексту.

Недостатки линейной модели

- Линейный классификатор требует, чтобы классы разделялись плоскостью.
- Это будет зависеть от способа построения вектора по тексту.
- Самый простой способ:
    - Вектор текста – сумма (или среднее) векторов входящих в него слов.
    - Вектор слова содержит ровно 1 единицу на месте индекса слова в словаре.

Недостатки линейной модели

- Линейный классификатор требует, чтобы классы разделялись плоскостью.
- Это будет зависеть от способа построения вектора по тексту.
- Самый простой способ:
  - Вектор текста – сумма (или среднее) векторов входящих в него слов.
  - Вектор слова содержит ровно 1 единицу на месте индекса слова в словаре.
- Часто вектора слов взвешивают обратно пропорционально логарифму их частоты (tf-idf).
- Можно добавлять дополнительные признаки:
  - Учитывать самые частые биграммы в тексте.
  - Добавлять вместе с вектором слова вектора его синонимов с весом $\alpha < 1$.

Недостатки линейной модели

- Линейный классификатор требует, чтобы классы разделялись плоскостью.
- Это будет зависеть от способа построения вектора по тексту.
- Самый простой способ:
  - Вектор текста – сумма (или среднее) векторов входящих в него слов.
  - Вектор слова содержит ровно 1 единицу на месте индекса слова в словаре.
- Часто вектора слов взвешивают обратно пропорционально логарифму их частоты (tf-idf).
- Можно добавлять дополнительные признаки:
  - Учитывать самые частые биграммы в тексте.
  - Добавлять вместе с вектором слова вектора его синонимов с весом $\alpha < 1$.
- Общий недостаток этих методов – их надо подгонять под задачу.
- Решать сложные задачи всё равно не получится (не учитывается порядок слов).

Модель мешка слов

- Пусть $T = [w_{i_1}, \dots, w_{i_m}]$ – исходный текст.
- Модель мешка слов складывает вектора для отдельных слов:

$$\begin{aligned} x_T &= x(w_{i_1}) + \ldots + x(w_{i_m}), \\ x(w_{i_1}) &= [0, \dots, 0, 1, 0, \dots, 0]. \end{aligned}$$

- Позиция единицы в $0/1$-векторе равна индексу слова в словаре.

Модель мешка слов

- Пусть $T = [w_{i_1}, \dots, w_{i_m}]$ – исходный текст.
- Модель мешка слов складывает вектора для отдельных слов:

$$\begin{aligned} x_T &= x(w_{i_1}) + \ldots + x(w_{i_m}), \\ x(w_{i_1}) &= [0, \dots, 0, 1, 0, \dots, 0]. \end{aligned}$$

- Позиция единицы в $0/1$-векторе равна индексу слова в словаре.
- Все слова одинаково далеки друг от друга.
- Чтобы учесть семантику, нужно чтобы вектора отражали смысловую близость.

Дистрибутивные вектора

- Основная идея дистрибутивной семантики:

  You should know the word by the company it keeps.

- Похожие слова встречаются в похожих контекстах.
- Можно представлять слово как усреднённый вектор контекста (Latent Semantic Analysis).

Дистрибутивные вектора

- Основная идея дистрибутивной семантики:

  You should know the word by the company it keeps.

- Похожие слова встречаются в похожих контекстах.
- Можно представлять слово как усреднённый вектор контекста (Latent Semantic Analysis).
- Можно предсказывать контекст слова по его вектору (или наоборот).
- Тогда похожие слова будут приводить к похожим контекстам.

Предсказание слова

- Дистрибутивные вектора обучаются на задаче предсказания слова по контексту:

  вкусное
  Я съел зелёное яблоко .
  печёное

- Будем для простоты считать, что контекст тоже состоит из одного слова (например, предсказывается следующее слово справа).

Предсказание слова

- Дистрибутивные вектора обучаются на задаче предсказания слова по контексту:

  вкусное
  Я съел зелёное яблоко .
  печёное

- Будем для простоты считать, что контекст тоже состоит из одного слова (например, предсказывается следующее слово справа).
- Для каждого слова вводятся два вектора: u_i (входной) и v_i (выходной).
- По u_i предсказывается вероятностное распределение на множестве v_j:

$$p(w_j|w_i) = \frac{\exp((u_i, v_j))}{\sum_k \exp((u_i, v_k))}$$

Предсказание слова

- Контекст (слово w_i) порождает вероятностное распределение $p = [p_1, \dots, p_{|D|}]$:

$$p_j = p(w_j | w_i) = \frac{\exp((u_i, v_j))}{\sum\limits_k \exp((u_i, v_k))}$$

- Это распределение сравнивается с эталонным $\hat{p} = [0, \dots, 1, \dots, 0]$:

$$\hat{p}_r = 1 \leftrightarrow w_r \text{ находится в контексте с } w_i$$

Предсказание слова

- Контекст (слово w_i) порождает вероятностное распределение $p = [p_1, \ldots, p_{|D|}]$:

$$p_j = p(w_j|w_i) = \frac{\exp\left((u_i, v_j)\right)}{\sum\limits_k \exp\left((u_i, v_k)\right)}$$

- Это распределение сравнивается с эталонным $\hat{p} = [0, \ldots, 1, \ldots, 0]$:

$$\hat{p}_r = 1 \leftrightarrow w_r \text{ находится в контексте с } w_i$$

- Штраф за различие – кросс-энтропия:

$$Q(\hat{p}, p) = -\sum_k \hat{p}_k \log p_k = -\log p_r$$

Предсказание слова

- Штраф за различие можно минимизировать градиентным спуском:

$$\begin{aligned} u_i &\leftarrow u_i - \frac{\partial Q(\hat{p}, p)}{\partial u_i}, \\ v_k &\leftarrow u_i - \frac{\partial Q(\hat{p}, p)}{\partial v_k}, k = 1, \dots, |D| \end{aligned}$$

- Этот штраф изменяет один контекстный вектор и все выходные вектора.

Предсказание слова

- Штраф за различие можно минимизировать градиентным спуском:

$$
\begin{aligned}
u_i &\leftarrow u_i - \frac{\partial Q(\hat{p}, p)}{\partial u_i}, \\
v_k &\leftarrow u_i - \frac{\partial Q(\hat{p}, p)}{\partial v_k}, k = 1, \ldots, |D|
\end{aligned}
$$

- Этот штраф изменяет один контекстный вектор и все выходные вектора.
- На практике рассматривают контекст из нескольких слов $(5-10)$ и предсказывают центральное.
- Вектор контекста: среднее векторов входящих в него слов.

Предсказание соседних слов

- В матричном виде можно записать как

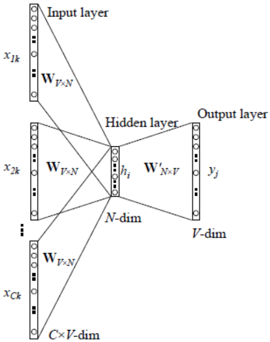
$$
\begin{aligned}
p &= \mathrm{softmax}(W^T U) \\
softmax(x_1, \dots, x_n) &= [\frac{e^{x_1}}{\sum_j e^{x_j}}, \dots, \frac{e^{x_1}}{\sum_j e^{x_j}}]
\end{aligned}
$$

- $\mathrm{softmax}$ применяется к каждой строчке матрицы.
- i-ый столбец каждой из матриц — представление i-го слова в словаре.

Обучение векторных представлений

- Схема предсказания:

$$
\begin{aligned}
h &= \frac{1}{C}(Wx_1 + \ldots + Wx_C),\\
p &= softmax(W'h)
\end{aligned}
$$

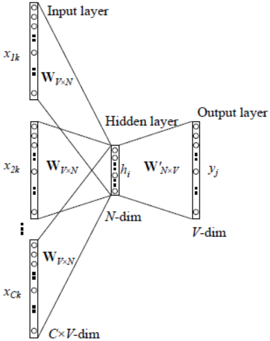


Обучение векторных представлений

- Схема предсказания:

$$
\begin{aligned}
h &= \frac{1}{C}(Wx_1 + \ldots + Wx_C),\\
p &= softmax(W'h)
\end{aligned}
$$

- Столбцы матриц W и W' — сжатые представления слов.



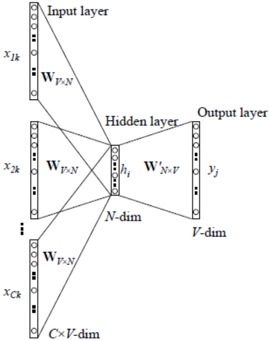
Обучение векторных представлений

- Схема предсказания:

$$\begin{aligned} h &= \frac{1}{C}(Wx_1 + \ldots + Wx_C), \\ p &= softmax(W'h) \end{aligned}$$

- Столбцы матриц W и W' — сжатые представления слов.
- Семантически близкие слова переходят в близкие вектора.
- Вектора сохраняют семантические связи:

$$\vec{king} - \vec{queen} = \vec{man} - \vec{woman}$$



Типы векторных представлений

- В матрице W похожие представления у векторов, приводящих к одинаковым контекстам.
- То есть у тех, которые часто встречаются вместе (синтагматические отношения).
- В выходной матрице W' похожи вектора, встречающиеся в одинаковых контекстах.
- Они лучше выражают парадигматические отношения (синонимия)
- Обычно (например, в gensim) используется второй вариант.

Негативная выборка

- Стандартный штраф — кросс-энтропия:

$$Q(\pi, \pi') = -\sum_i \pi_i \log \pi'_i$$

- Его можно минимизировать градиентным спуском.

Негативная выборка

- Стандартный штраф — кросс-энтропия:

$$Q(\pi, \pi') = -\sum_i \pi_i \log \pi'_i$$

- Его можно минимизировать градиентным спуском.
- Обычно делают по-другому: на каждом шаге выбирают "положительное" слово u и отрицательное слово v и стараются максимизировать разницу между их вероятностями:

$$\log p(u|w_i) - \log p(v|w_i) \to \max$$

- Положительные — слова, встречавшиеся в данном контексте в корпусе, отрицательные — все остальные.

Негативная выборка

- Стандартный штраф — кросс-энтропия:

$$Q(\pi, \pi') = -\sum_i \pi_i \log \pi'_i$$

- Его можно минимизировать градиентным спуском.
- Обычно делают по-другому: на каждом шаге выбирают “положительное” слово u и отрицательное слово v и стараются максимизировать разницу между их вероятностями:

$$\log p(u|w_i) - \log p(v|w_i) \to \max$$

- Положительные — слова, встречавшиеся в данном контексте в корпусе, отрицательные — все остальные.
- Например, для “корпуса” и $w_i = мама$,

Мама мыла раму
Моя мама красивая

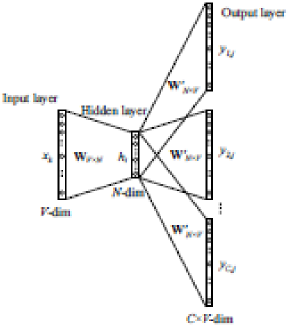
“положительными” словами будут *мыла, моя, красивая*, а “отрицательными” – все остальные.

Типы векторных представлений

- Мы рассмотрели модель CBOW (continuous bag of words), предсказывающую слова по контексту.

Вчера был	…	день
	замечательный прекрасный	
	преотличнейший?	

- Недостаток: система хуже запоминает редкие слова из того же контекста.
- Противоположная модель skipgram предсказывает контекст по слову.

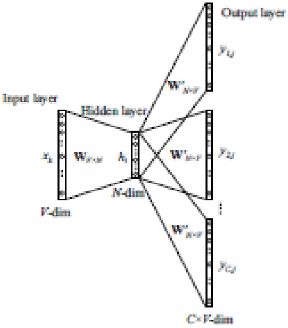


Типы векторных представлений

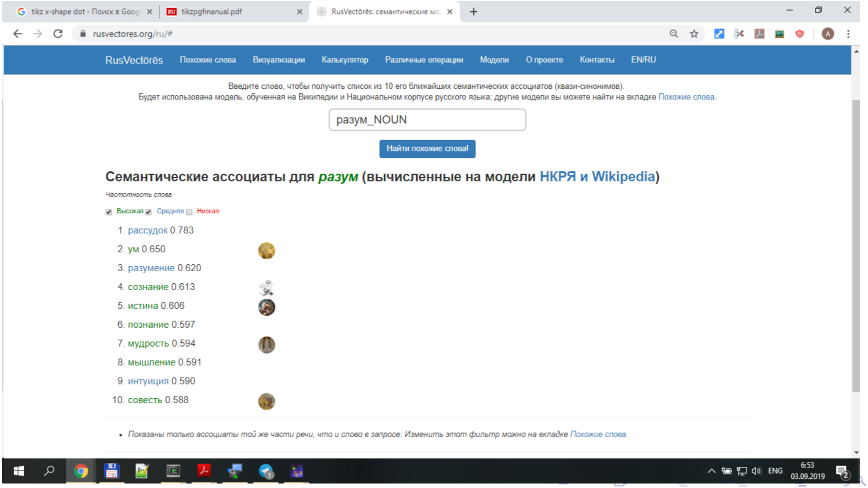
- Мы рассмотрели модель CBOW (continuous bag of words), предсказывающую слова по контексту.

Вчера был	...	ден
	замечательный	
	прекрасный	
	преотличнейший?	

- Недостаток: система хуже запоминает редкие слова из того же контекста.
- Противоположная модель skipgram предсказывает контекст по слову.
- Skipgram (при достаточном количестве данных) лучше работает для редких слов.
- CBOW требует меньше данных для обучения.



Векторные представления: пример



Векторные представления: пример