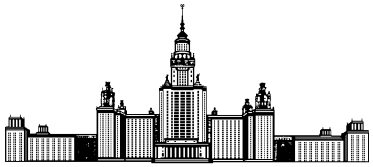
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики
Кафедра Системного Программирования

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА СТУДЕНТА 441 ГРУППЫ

Исследование и разработка моделей векторного представления слов

Выполнил:
студент 441 группы
Кемаев Юрий Юрьевич

Научный руководитель:
к.ф.-м.н., доцент
Турдаков Денис Юрьевич

Москва, 2017

Содержание

Аннотация

Последние годы стремительно растет количество областей применения методов машинного обучения. В частности, данные методы показывают лучшие на данный момент результаты в задачах из области обработки естественного языка (NLP, Natural Language Processing). Во многом это достигается за счет эффективного представления слов языка в форме, “понимаемой” вычислительной машиной. В данной работе будут рассмотрены и проанализированы широко используемые методы построения представлений слов в виде векторов многомерного пространства, а также будет представлено новое семейство алгоритмов, показывающих наилучшие результаты на общепринятых в сообществе исследователей тестовых задачах для русского языка.

Введение

Векторные представления слов — это математические объекты, векторы в многомерном пространстве, отражающие семантические и синтаксические особенности слов в "понятной" вычислительной машине форме.

Каждое измерение такого пространства отвечает за определенное свойство слов естественного языка, и в некоторых случаях данное свойство может иметь семантическую или синтаксическую интерпретацию, понятную для человека (Turian et al. [2010]).

Данные представления позволяют распределить слова естественного языка по классам эквивалентности так, что слова со схожими свойствами попадут в один класс.

Представления слов могут быть как созданы как вручную человеком (в форме лексиконов, словарей, тезаурусов и т.д.), так и автоматически с использованием нейросетевых методов машинного обучения.

Разумеется, ручное создание таких представлений требует большого количества квалифицированных человеческих ресурсов, и, следовательно, является продолжительным по времени и дорогим процессом. Поэтому в последнее время на практике используются представления, полученные автоматически, с помощью вычислительных машин.

Автоматические методы построения таких представлений базируются на распределительной гипотезе, которая утверждает, что ***смысл конкретно взятого слова определяется его контекстом***, т.е. словами, с которыми употребляется данное в большинстве случаев (Harris [1954]). Данная гипотеза оказывается верной на практике, поскольку получаемые представления качественно, эффективно и компактно отражают свойства естественного языка (Faruqui [2016]).

Области применения векторных представлений слов

Основная обширная область применения векторных представлений слов — ***машинное обучение в обработке естественного языка***.

Представления выступают в роли ***факторов обучения***, т.е. *подмножества признаков для некоторого конкретного слова*, которое (подмножество) позволяет более качественно решить такие задачи, как:

1. извлечение информации из текстов
2. распознавание речи
3. анализ тональности
4. тематическое моделирование
5. автоматическое реферирование
6. фильтрация контента
7. машинный перевод
8. генерирование текста
9. синтез речи

и др.

Данные задачи формируют проблемную область такой отрасли, как ***информационный поиск*** — *процесс поиска неструктурированной документальной информации, удовлетворяющей информационные потребности*[1].

Конкретные примеры использования векторных представлений в задачах информационного поиска:

1. перевод запросов пользователей в форму, понятную поисковой системе
2. ранжирование поисковых результатов
3. определение близких по смыслу запросов
4. классификация запросов по сегментам
5. разрешение неоднозначностей
6. исправление опечаток в запросах
7. голосовой поиск

и др.

В настоящее время, в век информационных технологий и глобализации, данные задачи являются одними из самых актуальных, так как позволяют сотням миллионов людей[2] удовлетворять информационные потребности быстро и максимально качественно.

Используемые понятия и обозначения

В дальнейшем будут использованы следующие **понятия**:

- *корпус* — некоторая совокупность текстов
- *лексикон (словарь) корпуса* — множество уникальных слов корпуса
- *база знаний* — структурированный источник информации о смысловых единицах определенного языка
- *тезаурус* — база знаний, содержащая описание лексики определенного языка

В дальнейшем, если не указано отдельно, будут использованы следующие **обозначения**:

- W — множество всех слов входного корпуса
- T — размер входного корпуса (количество содержащихся в корпусе слов)
- V — размер лексикона корпуса (количество уникальных в корпусе слов)
- N — размер выходных векторных представлений слов

[1] en.wikipedia.org/wiki/Information_retrieval

[2] smartinsights.com/search-engine-marketing/search-engine-statistics

1 Постановка задачи

Цель данной работы — **исследование существующих моделей векторных представлений и разработка новых алгоритмов для эффективной и качественной векторизации слов русского языка**.

Этапы работы:

1. Анализ существующих алгоритмов с целью выявления наиболее эффективных и применимых на практике
2. Обзор последних достижений исследователей в данной области с целью определения ключевых идей для построения новых алгоритмов
3. Разработка и реализация новых алгоритмов
4. Выбор данных и метрик для тестирования
5. Сравнительный анализ разработанных алгоритмов и популярных в настоящее время

2 Обзор существующих решений

В данной главе рассмотрены наиболее распространенные алгоритмы векторизации слов естественного языка, активно применяющиеся на практике. Также освещены основные концепции и идеи, объединяющие эти алгоритмы.

Следом приводятся описания существующих открытых баз знаний для русского языка, которые несут полезную для обучения представлений слов информацию.

Дополнительно приводятся методы для измерения качества векторных представлений слов.

2.1 Основные принципы обучения и работы нейронных сетей

В данном разделе приводится краткое описание работы нейронных сетей. Исчерпывающее описание представлено в работе Haykin [2007].

2.1.1 Архитектура нейронных сетей

Нейронная сеть представляет из себя набор *нейронов*, объединенных в слои.

Первый слой называется *входным*, последний — *выходным*, остальные — *скрытыми*.

Нейронная сеть, содержащая хотя бы один скрытый слой, называется *многослойной*.

Каждый нейрон сети преобразует совокупность входных сигналов в выходной на основе своих синаптических весов, порога сигнала и функции активации.

В классических нейронных сетях входными сигналами нейронов текущего слоя являются выходные сигналы предыдущего. Преобразование, проводимое каждым нейроном, можно записать как:

$$y = f(\sum_{i=1}^{n} w_i x_i + w_0),$$

где x — вектор входных сигналов, w — вектор синаптических весов, w_0 — пороговая величина сигнала, $f(.)$ — функция активации, y — выходной сигнал.

Рассмотрим модель нейрона с функцией активации $f(u) = \operatorname{sgn} u$ и задачу *бинарной классификации* объектов из некоторой обучающей выборки $X^l = \{x_i, y_i\}_{i=1}^l$, где $y_i = \operatorname{sgn}\langle w_*, x_i\rangle$.

Пусть алгоритм классификации имеет вид $a(w, x) = \operatorname{sgn}\langle w, x\rangle$. Следовательно, ошибка на объекте x_i происходит при $y_i\langle w, x_i\rangle < 0$.

Пусть правило, по которому *модифицируются* веса, имеет следующий вид: если алгоритм ошибается на объекте x_i, положим $w_{(j+1)} = w_{(j)} + \eta x_i y_i$, где $\eta > 0$ — некоторая константа.

При такой стратегии обновления вектора весов, называемой *правилом Хэбба*, *теорема Новикова* (Haykin [2007]) **гарантирует** его сходимость за конечное число шагов и при любых начальных параметрах к некоторому вектору w^{**}, такому, что гиперплоскость $\langle w^{**}, x\rangle = 0$ безошибочно разделяет два класса из входной выборки.

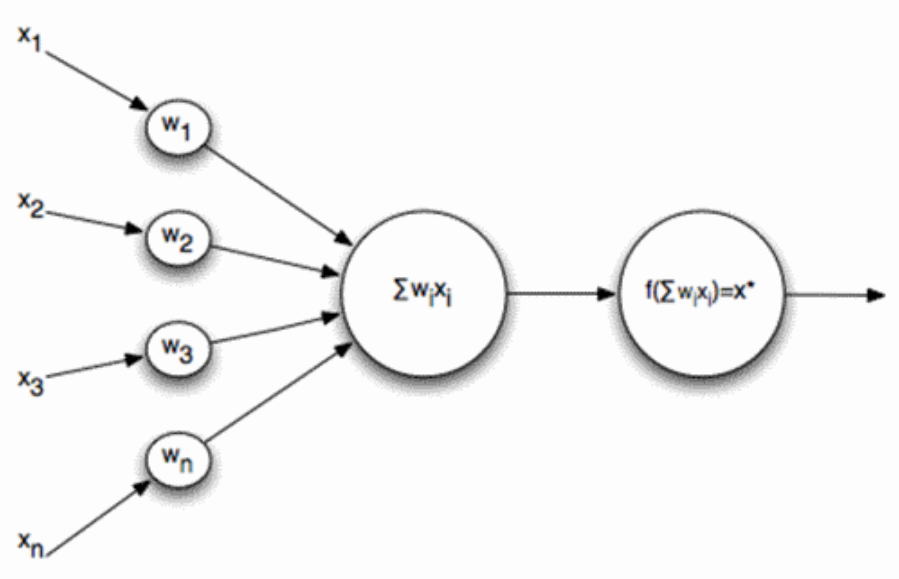


Рис. 1: Модель нейрона, wiki.witology.com

Этот результат **не имеет практической значимости, однако выявляет способность нейрона к обучению**. Выражение $a(x) = \text{sgn}\langle w^{**}x\rangle$ задает алгоритм безошибочной классификации входной выборки.

Фундаментальный смысл для теории нейронных сетей имеет *универсальная теорема аппроксимации* (Haykin [2007]), которая формулируется следующим образом:

Теорема 2.1 (Универсальная теорема аппроксимации)**.** *Пусть* $\phi(.)$ *– отличная от константы ограниченная, непрерывная и монотонно возрастающая функция. Пусть* $I_{m_0} := [0,1]^{m_0}$ *– единичный гиперкуб размерности* m_0*. Также пусть* $C(I_{m_0})$ *– множество непрерывных функций на* I_{m_0}*. Тогда* $\forall f \in C(I_{m_0}), \epsilon > 0$ *существуют такие* $m_1 \in \mathbb{Z}, \alpha_i, \beta_i, w_{ij} \in \mathbb{R}$*, где* $i = \{1, \ldots, m_1\}, j = \{1, \ldots, m_0\}$*, что для функции, определяемой как*

$$F(x_1, \ldots, x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \phi(\sum_{j=1}^{m_0} w_{ij} x_j + b_i),$$

выполняется

$$|F(x_1, \ldots, x_{m_0}) - f(x_1, \ldots, x_{m_0})| < \epsilon$$

для всех $x_1, x_2, \ldots, x_{m_0}$ *из области определения* $f(.)$*.*

Теорема 2.1 имеет **огромное теоретическое значение**, но ничего не утверждает о том, как искать величины $m_1 \in \mathbb{Z}, \alpha_i, \beta_i, w_{ij} \in \mathbb{R}$, что делает ее неприменимой на практике.

Что же касается практики, то для обучения нейронов сети преимущественно часто применяется *алгоритм обратного распространения ошибки*, который будет рассмотрен в следующей секции.

2.1.2 Алгоритм обратного распространения ошибки (backpropagation)

Обучение нейронной сети будем трактовать как задачу оптимизации.

Пусть дана многослойная нейронная сеть. Занумеруем все нейроны от 1 до N (нумерация сквозная). Обозначим за N_{out} множество индексов нейронов выходного (последнего) слоя. Также пусть y_k^* – реальные выходные сигналы на этих нейронах, а y_k – желаемые сигналы, $k \in N_{out}$.

Процесс обучения нейронной сети заключается в том, чтобы приблизить выходные сигналы y_k^* нейронной сети к желаемым y_k.

Пусть w_{ij} - синаптический вес связи от i-го нейрона к j-му, а x_{ij} - сигнал, передающийся по ней. Введем $Children(j) = \{t|\ t \in [1, N],\ \exists x_{jt}\}$ – множество нейронов, каждый из которых имеет связь, идущую из нейрона j (т.е. нейроны последующего за j-м слоя).

Как было указано ранее, $f_j(.)$ – функция активации нейрона j.

Метрикой качества работы нейронной сети возьмем **MSE (mean squared error)** — *среднеквадратичное отклонение значений выходных сигналов от желаемых*:

$$E = \frac{1}{2} \sum_{k \in N_{out}} (y_k^* - y_k)^2$$

Соответственно, в таком случае задача обучение нейронной сети сводится к минимизации введенного функционала качества. Это делается с помощью метода градиентного спуска по весам w_{ij} связей сети:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}},$$

где $\eta \in (0, 1)$ — величина (не обязательно константа), определяющая *скорость спуска*.

Поскольку w_{ij} влияет на выход сети только как часть суммы $u_j = \sum_i w_{ij} x_{ij}$, можно вычислить

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial u_j} \frac{\partial u_j}{\partial w_{ij}} = x_{ij} \frac{\partial E}{\partial u_j}$$

Далее возможны 2 случая:

1) $j \in N_{out}$, тогда u_j вносит вклад только в значение на выходе j-го нейрона y_j^*, следовательно,

$$\frac{\partial E}{\partial u_j} = \frac{\partial E}{\partial y_j^*} \frac{\partial y_j^*}{\partial u_j} = \Big(\frac{\partial}{\partial y_j^*} \frac{1}{2} (y_j^* - y_j)^2 \Big) \Big(\frac{\partial f_j(u)}{\partial u} \Big|_{u=u_j} \Big) = y_j^* \frac{\partial f_j(u)}{\partial u} \Big|_{u=u_j},$$

далее величина явно вычисляется для конкретно взятой функции активации.

2) $j \notin N_{out}$, тогда, применяя правило дифференцирования сложной функции (*цепное правило, chain rule*), выражается

$$\frac{\partial E}{\partial u_j} = \sum_{k \in Children(j)} \frac{\partial E}{\partial u_k} \frac{\partial u_k}{\partial w_{ij}} = \sum_{k \in Children(j)} w_{jk} \frac{\partial E}{\partial u_k}$$

через значения, вычисляемые на последующем слое.

Таким образом, **зная значение градиента на последующем слое, можно вычислить его значение на текущем**. При этом **на последнем слое значение градиента вычисляется явно**.

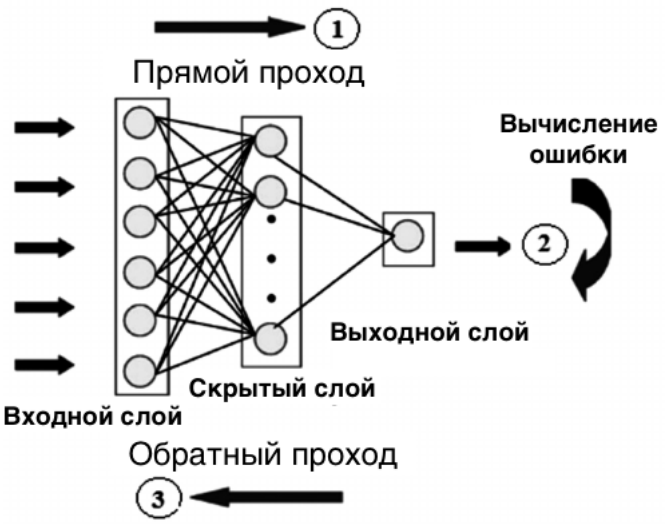


Рис. 2: Схема алгоритма обратного распространения ошибки

Следовательно, **вычисление всех градиентов идет в порядке слоев от последнего к первому**. Именно из-за этой особенности данный алгоритм называется *алгоритмом обраного распространения ошибки*.

Также количество раз, которое каждый из объектов обучающей выборки подается на вход за время всего процесса обучения, называется *количеством эпох*.

Из определения следует, что алгоритм легко обобщается на всевозможные дифференцируемые штрафные функционалы E, которые выбираются в зависимости от решаемой задачи.

Множитель η носит название *learning rate* и, как правило, в процессе обучения постепенно уменьшается до нуля (*learning rate decay*), так как на более поздних шагах веса должны подстраиваться более "тонко", чтобы удовлетворять как можно большему количеству рассмотренных элементов обучающей выборки (процесс *fine tuning*[1]).

Стоит отметить, что **алгоритм не гарантирует сходимости к глобальному минимуму функционала ошибки** E, что следует из свойств метода градиентного спуска.

Тем не менее, алгоритм обратного распространения ошибки является наиболее популярным для обучения нейронных сетей на сегодняшний день. При этом обычно в качестве функции активации $f(.)$ берутся сигмоидальные функции вида

$$f(u) = \frac{1}{1 + \exp^{-\alpha u}}, \alpha > 0,$$

которые являются непрерывно дифференцируемыми.

Также, как правило, в качестве штрафного функционала в задачах регресии берется *среднеквадратичное отклонение (MSE)*, а в задачах классификации - *кросс-*

[1]en.wikipedia.org/wiki/Fine-tuning

энтропия, определяемая как:

$$E = -\frac{1}{n}\sum_i (y_i \ln(y_i^*) + (1 - y_i)\ln(1 - y_i^*)),$$

где y_i^* – вероятность принадлежности объекта классу $y_i = 1$.

Также в задачах классификации часто требуется на выходе получить распределение вероятностей принадлежности входных объектов тому или иному классу. Для этого к нейронной сети добавляется специальный последний слой, называемый *softmax*.

Softmax слой действует из $\mathbb{R}^n \mapsto \mathbb{R}^n$, преобразуя каждую компоненту a_j ($j \in [1, n]$) входного вектора a^n по правилу

$$S_j = \frac{e^{a_j}}{\sum_{i=1}^n e^{a_i}}$$

Легко видеть, что величина каждой из компонент выходного вектора S^n лежит в отрезке $(0, 1)$, и их сумма равна единице. **Это позволяет интерпретировать S^n как вектор распределения вероятности принадлежности объекта каждому из n классов.**

Исследуем возможность применения алгоритма обратного распространения ошибки в сети, содержащей слой *softmax*.

Как можно заметить, сложность заключается в том, что для подсчета значения выхода в каждом нейроне требуется знать выходы всех нейронов этого слоя (сумма в знаменателе).

Однако, оперируя свойствами экспоненты, данное требование можно обойти.

Чтобы обучить такую нейронную сеть с помощью алгоритма обратного распространения ошибки, нужно получить выражения для частной производной по каждой входящей в нейрон *softmax-слоя* связи:

$$\frac{\partial S_i}{\partial a_j} = \frac{\partial \frac{e^{a_i}}{\Sigma}}{\partial a_j},$$

где для компактности $\Sigma = \sum_{k=1}^n e^{a_k}$

Возможны 2 случая:

1) $i = j$, тогда

$$\frac{\partial \frac{e^{a_i}}{\Sigma}}{\partial a_i} = \frac{e^{a_i}\Sigma - e^{a_i}e^{a_i}}{\Sigma^2} = \frac{e^{a_i}}{\Sigma}\frac{\Sigma - e^{a_i}}{\Sigma} = S_i(1 - S_i)$$

2) $i \neq j$, тогда

$$\frac{\partial \frac{e^{a_j}}{\Sigma}}{\partial a_i} = \frac{0 - e^{a_j}e^{a_i}}{\Sigma^2} = -\frac{e^{a_j}}{\Sigma}\frac{e^{a_i}}{\Sigma} = -S_j S_i$$

Обобщая эти два случая, можно записать:

$$\frac{\partial S_i}{\partial a_j} = S_i(\delta_{ij} - S_j),$$

где $\delta_{ij} = \begin{cases} 1 & \text{, если } i = j \\ 0 & \text{, если } i \neq j \end{cases}$ – символ Кронекера.

Таким образом, производная *softmax-слоя* выражается через значения выходных сигналов на самом слое.

2.2 Методы построения векторных представлений слов

В этой главе будут рассмотрены некоторые популярные алгоритмы для построения распределенных векторных представлений слов естественного языка.

Рассматриваемые методы относятся к классу *unsupervised*, т.е. для для их работы **не требуются какие-либо размеченные данные**.

Как было указано ранее, данные методы базируются на *распределительной гипотезе*, которая утверждает, что **смысл конкретно взятого слова определяется его контекстом** (Harris [1954]).

Качество получаемых представлений сильно зависит от входных данных и используемых гиперпараметров алгоритма их построения. Например, чем более обширным будет входной корпус, тем более качественные представления получаются на выходе (Levy et al. [2015]).

2.2.1 Семейство моделей Word2Vec

В работах Mikolov et al. [2013a] и Mikolov et al. [2013b] были представлены две нейросетевые модели, которые на данный момент являются одними из самых популярных вследствие своей простоты, эффективности, устойчивости к входным данным и высокого качества выходных представлений относительно предшествующих аналогов.

Каждому слову в представленных моделях ставится в соответствие ровно один уникальный вектор (*one-hot-encoding*, биекция между лексиконом и входным слоем).

Обе модели строят векторные представления в процессе прохода по словам входного корпуса *скользящим окном* и максимизации своей целевой функции.

Авторы модели определяют размер скользящего окна k динамическим, а именно: k равновероятно принимает целые значения из отрезка $[1, C]$, где C — параметр модели, определяющий максимальный размер окна. Данный прием можно интерпретировать как **усреднение выходных сигналов модели по всевозможным размерам контекста от 1 до** C.

Скорость обучения моделей (*learning rate*) линейно уменьшается в процессе обучения до нуля.

Начальные значения синаптических весов входного слоя инициализируются случайно из равномерного распределения на отрезке $[-\frac{1}{2N}; \frac{1}{2N}]$.

Далее приводится краткий обзор каждой из представленных моделей. Исчерпывающий обзор (с явным выводом формул обновления весов) можно найти в работах Rong [2014] и Goldberg and Levy [2014].

Стоит отметить, что публикация работ Mikolov et al. [2013a] и Mikolov et al. [2013b] привлекла внимание многих исследователей в области обработки естественного языка, и через некоторое время начали появляться новые модели векторизации слов на основе контекста, имеющие в своей основе рассматриваемые.

Основные свойства данных моделей:

- Простая архитектура
- Устойчивость ко входным данным
- Высокое качество выходных представлений
- Наличие в открытом доступе авторской эффективной многопоточной реализации на языке *C*
- Отсутствие теоретического обоснования хороших результатов
- Большое количество гиперпараметров
- *Skipgram with Negative Sampling* на практике обычно оказывается лучше *Continuous Bag-of-Words*

Continuous Bag-of-Words Первая из этих моделей — *Continuous Bag-of-Words* — предсказывает слово w_t по его контексту $w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}$.

Обучение сети заключается в минимизации штрафной функции следующего вида:

$$E = -\sum_{i,k} \log P(w_t | w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k})$$

Архитектура нейронной сети состоит из 3-х полносвязных слоев, которые называются *input*, *projection(hidden)* и *output* слои соответственно.

В синаптических весах *input-слоя* размера $V \times N$ заключены векторные представления слов входного корпуса.

Projection-слой размера $N \times 1$ предназначен для усреднения распределенных векторов, соответствующих словам контекста.

Output-слой имеет размер $V \times N$ и реализует *softmax* с целью получения распределения вероятности каждого слова в контексте рассматриваемого.

Вычислительная сложность процесса обучения определяется как $O(V \times D + D \times V)$. Данную оценку можно улучшить до $O(V \times D + D \times \log(V))$, если использовать *иерархический (hierarchical) softmax*, предложенный авторами в оригинальной статье и рассмотренный далее.

Skipgram Вторая представленная модель называется *Skipgram* и отличается от *Continuous Bag-of-Words* тем, что предсказывает контекст $w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}$ по слову w_t, а не наоборот.

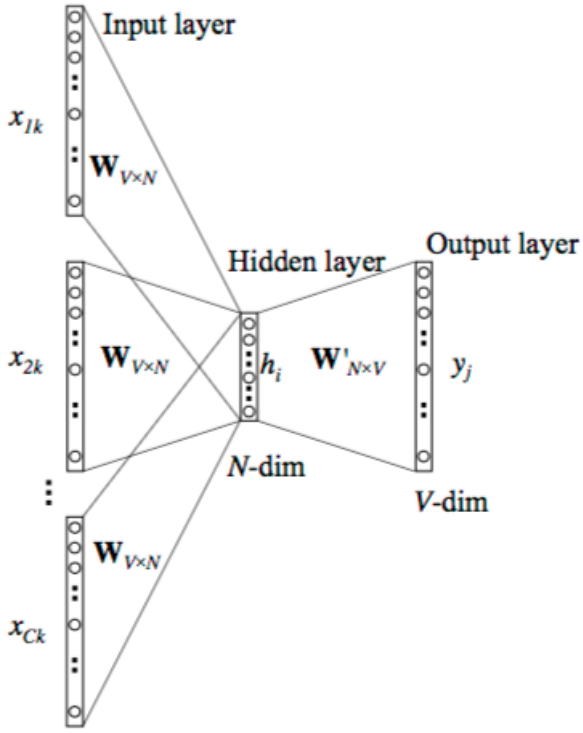


Рис. 3: Архитектура *Continuous Bag-of-Words*, Rong [2014]

Исходя из этого, штрафная функция принимает следующий вид:

$$E = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j\in Context_k(t)}\log P(w_j|w_t),$$

где $Context_k(t) = \{\max(0, t-k), \ldots, \min(T, t+k)\}$.

Условная вероятность контекстного слова при заданном в соответствии с определением *softmax* равна:

$$P(w_j|w_t) = \frac{\exp^{\langle v_j, v_t\rangle}}{\sum_{i\in AllContexts_k(t)}\exp^{\langle v_i, v_t\rangle}} \quad ,$$

где $AllContexts_k(t)$ – мультимножество всевозможных контекстов размера k слова w_t.

Соответствующая архитектура нейронной сети также состоит из 3-х полносвязных слоев, однако ***hidden-слой* представляет из себя копию строки *input-слоя*, которая соответствует рассматриваемому слову**.

Слои отражают тот же смысл, что и в модели *Continuous Bag-of-Words*.

Ключевое отличие данной модели от первой в том, что слово w_t предсказывается столько раз, сколько слов содержится **во всех** его контекстах, на основе только одного из слов **в текущем** контексте.

Сложность обучения данной модели $O(V \times D + N \times D \times V)$, с помощью *иерархического softmax* улучшается до $O(V \times D + N \times D \times \log(V))$

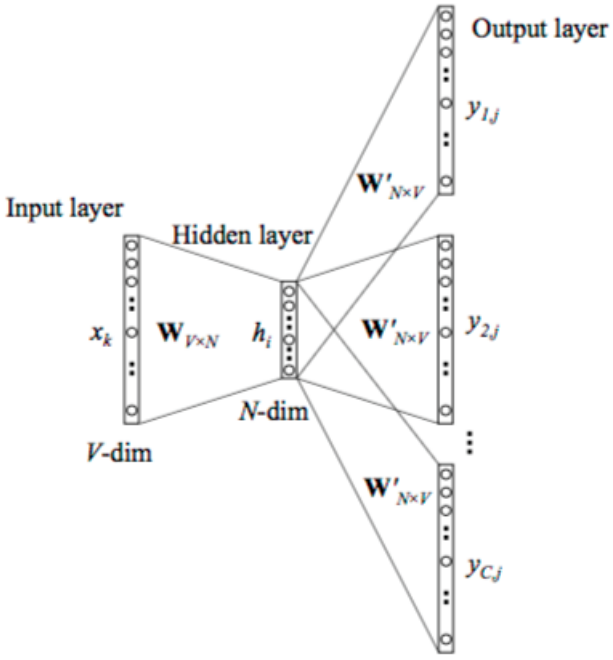


Рис. 4: Архитектура *Skipgram*, Rong [2014]

Отдельно стоит отметить модификацию под названием *Negative Sampling*, предложенную авторами в оригинальной статье.

2.2.2 Модификации моделей Word2Vec

В данной секции рассмоатриваются две модификации, понижающие сложность и сохраняющие качество оригинальных моделей.

Negative Sampling — это приближение условной вероятности $P(w_j|w_t)$ в функции штрафа модели *Skipgram* следующим образом:

$$P(w_j|w_t) = \frac{\sigma(\langle v_j, v_t\rangle)}{\sum_{i=1}^{q} \mathbb{E}_{w_i\sim P_{noise}(w)}\sigma(\langle v_i, v_t\rangle)}$$

Данное приближение означает следующее: **целью обучения модели становится отделение рассматриваемого контекстного слова от случайных q (параметр модели) "шумных" слов**, которые порождаются некоторым распределением $P_{noise}(w)$.

Авторы в Mikolov et al. [2013b] рекомендуют брать q из отрезка $[5, 20]$, а в качестве распределения случайных слов брать

$$P_{noise}(w) = \frac{\#(w)^{\frac{3}{4}}}{\sum_{w\in Vocab}\#(w)^{\frac{3}{4}}} \quad ,$$

где $\#(w)$ — абсолютная частота слова w во входном корпусе.

Данная модификация позволяет улучшить оценку сложности обучения до $O(V \times D + N \times D \times q)$, при этом сохраняя (в некоторых случаях *улучшая*) качество выходных представлений слов (Mikolov et al. [2013b]).

Иерархический softmax заключается в следующем: по словарю размера V строится *дерево Хаффмана*, каждой нелистовой вершине которого соответствует один нейрон *output-слоя* с N синаптическими весами.

Теперь пусть X — множество всех нейронов третьего слоя сети, которые оказываются на пути от корня к слову w_t. Каждый нейрон из X производит скалярное умножение вектора своих синаптических весов на вектор выходных сигналов скрытого слоя и к результату применяет логистическую функцию. Упорядоченная в ходе спуска по дереву совокупность выходных сигналов нейронов из X (их количество равно длине кода Хаффмана слова w_t) формирует вектор $w_{t'}$, который сравнивается с кодом Хаффмана слова w_t. В итоге принимается:

$$P(w_t|w_{t-k}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+k}) = \prod_i |w_t^i - w_{t'}^i|,$$

где $w_t^i \in \{0, 1\}$ — цифра, стоящая в позиции i в коде Хаффмана слова w_t.

2.2.3 SSPMI Matrix Factorization

Как уже было упомянуто ранее, публикация работ Mikolov et al. [2013a] и Mikolov et al. [2013b] вызвала волну интереса к моделям векторизации слов на основе контекста.

Вскоре появилась работа Levy and Goldberg [2014], в которой был проведен подробный разбор модели ***SGNS*** *(Skipgram with Negative Sampling),* а также описан новый подход к построению таких моделей.

В данной публикации рассматриваются матрицы $W^{V \times D}$ и $C^{V \times D}$ синаптических весов нейронов *input* и *output* слоев нейронной сети метода *SGNS* соответственно.

Вводится в рассмотрение матрица

$$M = W \cdot C^T,$$

элементы которой M_{ij} отражают величину *меры ассоциации* слова w_i и w_j.

Авторы в Levy and Goldberg [2014] показывают, что метод ***SGNS*** **является алгоритмом неявной факторизации матрицы $M^{V \times V}$, элементами которой являются величины некоторой меры ассоциации между словами лексикона**.

Пусть имеются две случайных величины ξ и η, определенные на вероятностных пространствах X и Y соответственно. ***Поточечной взаимной информацией*** *(pointwise mutual information)* называется величина

$$PMI(x, y) = log \frac{P(x, y)}{P(x)P(y)},$$

где $x \in X$ и $y \in Y$ – некоторые элементарные исходы.

Данная величина является *мерой ассоциации* между элементарными событиями x и y случайных величин ξ и η соответственно[1].

Авторы рассматриваемой работы показывают, что в методе $SGNS$ факторизуемая матрица M^{SGNS} описывается как:

$$M_{ij}^{SGNS} = W_i \cdot C_j = PMI(w_i, c_j) - \log k,$$

где w_i – слово во входном корпусе, $c_j \in Context(w_i)$ – слово из контекста w_i, $k \in \mathbb{N}$ – количество взятых шумных слов, а

$$PMI(w_i, c_j) = \log \frac{\#(w_i, c_j) \cdot V}{\#(w_i)\#(c_j)}$$

— *эмпирическая оценка поточечной взаимной информации слов* w_i *и* c_j.

Использование PMI для измерения величины ассоциации между словами в задачах естественной обработки текстов было введено в работе Church and Hanks [1989], где приводятся свойства и подробное описание данной функции.

На следующем шаге авторы модифицируют PMI, получая *смещенную положительную взаимную поточечную информацию*:

$$SPPMI_k(w, c) = max(PMI(w, c) - \log k, 0)$$

Данная модификация позволяет избежать неопределенности в случае $\#(w, c) = 0$, а также учитывает лишь положительные ассоциации (что более свойственно человеку, так как, например, пара слова “лето” и “тепло” интуитивно несет в себе больший смысл, чем “лето” и “снег”).

Итак,

$$M_{ij}^{SPPMI_k} = SPPMI_k(w_i, w_j) = max(PMI(w, c) - \log k, 0)$$

Метод, предложенный авторами, заключается в применении сингулярного разложения[2] к матрице M^{SPPMI_k} на произведение матриц $U \cdot \Sigma \cdot V^T$, где U и V ортонормированы, а Σ – диагональная матрица, составленная из собственных значений M^{SPPMI_k}.

Пусть Σ_d — диагональная матрица, составленная из d наибольших собственных значений M^{SSPMI_k}, и пусть U_d и V_d — матрицы, полученные взятием соответствующих столбцов U и V. По свойствам сингулярного разложения $M_d = U_d \cdot \Sigma_d \cdot V_d^T$ — матрица ранга d, наилучшим образом приближающая исходную матрицу M^{SSPMI_k} по матричной норме L_2.[3]

Далее, на последнем шаге метода берутся:

$$W = U_d \cdot \sqrt{\Sigma},$$

[1] en.wikipedia.org/wiki/Pointwise_mutual_information

[2] en.wikipedia.org/wiki/Singular_value_decomposition

[3] https://en.wikipedia.org/wiki/Matrix_norm

$$C = V_d \cdot \sqrt{\Sigma}.$$

Итоговые векторные представления слов являются соответствующими строками матрицы W.

Основные свойства данной модели:

- Имеется теоретическое обоснование метода
- Высокое качество выходных представлений
- В классе некоторых задач результат превосходит другие модели
- Небольшое количество гиперпараметров

2.2.4 FastText

Следующая модель, предложенная в работах Bojanowski et al. [2016] и Joulin et al. [2016], представляет из себя модификацию *SGNS*.

В модели *SGNS* используется *one-hot-encoding*, т.е. с каждым словом сопоставляется **ровно один** вектор, из-за чего информация о структуре слова и его морфемах полностью игнорируется.

В предложенной модификации авторы вводят функцию штрафа специального вида, которая учитывает эту информацию.

Функционал штрафа модели *SGNS* имеет вид:

$$E = -\sum_{t=1}^{T} \sum_{c \in Context(w_t)} \log p(w_c|w_t) \quad ,$$

$$p(w_c|w_t) = \frac{exp^{s(w_t,w_c)}}{\sum_{j=1}^{V} e^{s(w_t,w_j)}} \quad ,$$

где $s(w_i, w_j)$ – некоторая оценка близости слов w_i и w_j.

Задача минимизации данного функционала может быть рассмотрена как совокупность независимых задач *бинарной классификации*, целью которых является предсказание появления (или отсутствия) слова в контексте.

Для слова w_t и его контекста w_c функция правдоподобия имеет вид:

$$-log(1 + e^{-s(w_t,w_c)}) - \sum_{n \in Noise_{t,c}} log(1 + e^{s(w_t,w_n)}) \quad ,$$

где $Noise_{t,c}$ – множество индексов шумных слов для w_t.

Обозначая логистическую функцию $l(x) = log(1+e^{-x})$, итоговый штрафной функционал принимает вид:

$$E = -\sum_{t=1}^{T} \sum_{c \in Context_t} l(s(w_t, w_c)) + \sum_{n \in Noise_{t,c}} l(-s(w_t, w_n))$$

Стоит отметить, что в оригинале функция, определяющая оценку близости двух векторных представлений, имеет вид:

$$s(w, c) = \langle v(w), v(c) \rangle$$

Авторы модели *FastText* **модифицируют эту функцию так, чтобы она учитывала особенности структуры слов языка**.

Пусть дана строка S длины L, S_i — символ, стоящий в S на позиции i. Тогда

$$G_k = S_i S_{i+1} \dots S_{i+k-1}, \quad i = 1, \dots, L - k + 1$$

– множество *n-граммм* длины k данной строки S, т. е. всевозможные непрерывные подстроки S длины k.

Введем

$$G^w_{k_1 k_2} = \cup_{k=k_1}^{k_2} G_k$$

— множество всевозможных n-грамм длины от k_1 до k_2 слова w.

Оценка близости двух векторных представлений в модели *FastText* принимает вид:

$$s(w, c) = \langle v(w), v(c) \rangle + \sum_{g \in G^w_{k_1 k_2}} \langle v(g), v(c) \rangle,$$

т.е. **добавляется слагаемое, равное сумме скалярные произведений векторных представлений n-грамм и контекстного слова**.

Авторы в Joulin et al. [2016] отмечают повышение качества получаемых на выходе алгоритма векторных представлений для языков с богатой морфологией.

Особенности модели:

- Модель является модификацией *SGNS*
- Учитывает морфологические особенности языка
- Качество превосходит *SGNS*
- В открытом доступе имеется авторская эффективная реализация модели на языке *C++*

2.3 Особенности получаемых векторных представлений слов

Выходные представления являются компактными векторами, отражающими особенности строения языка.

В Mikolov et al. [2013a] было отмечено, что вектора *аналогичных понятий языка* обладают следующим свойством:

$$v_{king} - v_{man} \approx v_{queen} - v_{woman},$$

где v_w – вектор слова w.

Таким образом, зная **три слова из такого отношения, можно алгебраически получить четвертое**:

$$v_{king} = \underset{w \in Vocabulary}{\text{argmin}} \; ||v_w - (v_{queen} - v_{woman} + v_{man})||$$

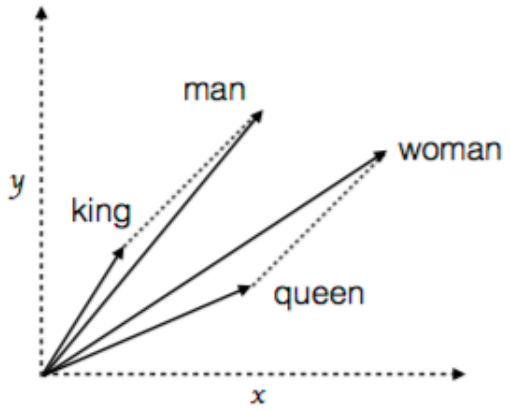


Рис. 5: Закономерность между парами аналогий, Faruqui [2016]

Пусть даны два вектора $v_{1,2} \in \mathbb{R}^n$. Тогда мера, определяемая как косинус угла между ними,

$$cos(\theta) = \frac{\langle v_1, v_2 \rangle}{||v_1|| \cdot ||v_2||},$$

называется *косинусной мерой*.

Еще одно свойство представлений заключается в том, что **векторы близких по смыслу слов близки по косинусной мере**.

Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

Рис. 6: Ближайшие к “sweden” по косинусной мере слова, Mikolov et al. [2013a]

2.4 Открытые базы знаний для русского языка

В данном разделе приводится краткий обзор некоторых **открытых** *баз знаний* русского языка, которые содержат потенциально полезную информацию для построения векторных представлений.

2.4.1 Тезаурус РуТез-2.0

Данный тезаурус содержит:

- 31.5 тыс. понятий
- 111.5 тыс. различных слов и выражений русского языка
- 4 типа отношений между словами

Единицей описания в данном тезаурусе является *понятие*, отражающее значимые классы сущностей, различаемых людьми в разных областях жизнедеятельности: политике, психологии и др.

Понятия в *РуТез-2.0* могут иметь достаточно большие ряды онтологических синонимов.

В тезаурусе имеется четыре основных типа отношений:

1. родовидовое отношение ниже-выше
2. отношение часть-целое
3. отношение несимметричной ассоциации
4. отношение симметричной ассоциации

Рис. 7: Информация о слове “сад”, labinform.ru

Исчерпывающее описание тезауруса РуТез можно найти в работе Loukachevitch and Dobrov [2013] и на странице ресурса[1].

2.4.2 Russian Distributional Thesaurus

Russian Distributional Thesaurus (RDT) — проект создания открытого дистрибутивного тезауруса русского языка.

На данный момент ресурс содержит следующие компоненты:

- вектора слов
- граф подобия слов
- множество гиперонимов
- инвентарь смыслов слов

[1] labinform.ru/pub/ruthes/index.htm

Количество понятий ≈ 932 тыс.

Все ресурсы были построены автоматически на основании корпуса текстов книг на русском языке (12.9 **млрд словоупотреблений).**

Для создания дистрибутивного тезауруса была использована модель *Skipgram*, обученная на корпусе 12.9 млрд словоупотреблений.

Подробную информацию можно найти в работе Panchenko et al. [2017] и на странице ресурса[1].

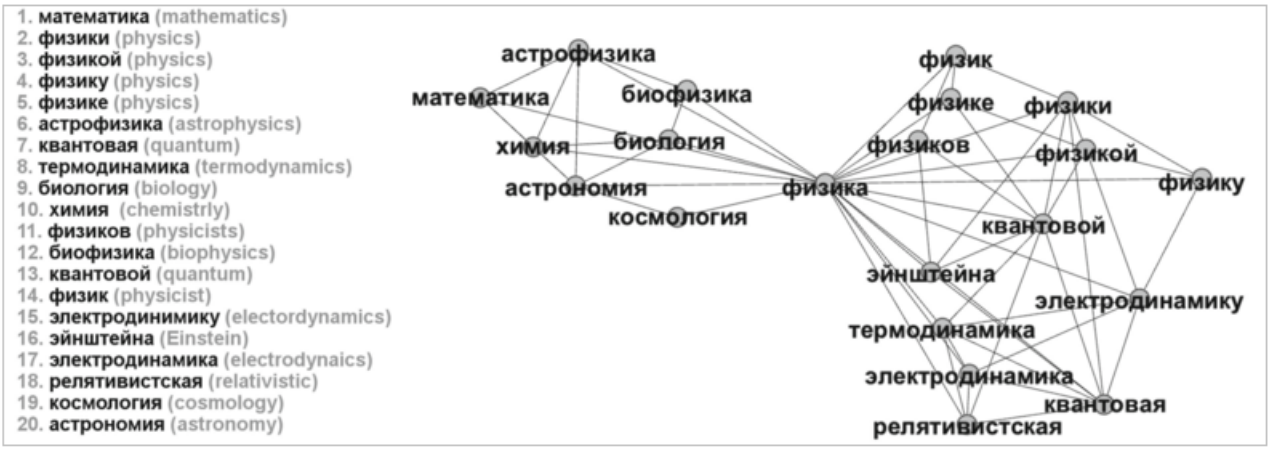


Рис. 8: Ближайшие соседи слова “физика” в дистрибутивном тезаурусе *RDT*, *nlpub.ru*

2.5 Методы оценки качества представлений

В среде исследователей качество векторных представлений слов оценивается по нескольким принципиально разным задачам.

В данной секции рассмотрены основные типы таких задач и составлен обзор комплекта *RUSSE* для оценки качества векторов слов русского языка.

2.5.1 Меры сходства, основанные на корелляции

Далее приводится описание задач, на которых **оценивается способность векторных представлений к отражению сходства слов и выявлению языковых аналогий**.

В задачах этой секции используется понятие *корреляция Спирмена*[2], которая является **непараметрической оценкой ранговой корелляции (статистической зависимости между упорядоченными значениями) двух случайных величин**.

Пусть имеется случайная величина ξ и набор ее независимых наблюдений (выборка) $X_1, ..., X_n$ размера n.

[1] nlpub.ru/Russian_Distributional_Thesaurus

[2] en.wikipedia.org/wiki/Spearman‘s_rank_correlation_coefficient

Пусть после упорядочивания данной выборки наблюдение X_i займет позицию $rg(X_i)$, называемую *рангом* X_i.

Тогда корелляция Спирмена может быть подсчитана как

$$r_s = 1 - \frac{6\sum_{i=1}^{n}(rg(X_i) - rg(Y_i))^2}{n(n^2-1)}$$

Данная оценка показывает **насколько хорошо отношение двух случайных величин сохраняет свойство монотонности**.

Оценки сходства понятий Задача оценки сходства понятий позволяет определить, **насколько близость векторов отражает семантическую и синтаксическую близость слов в языке (относительно человеческого восприятия)**.

Данные для этого типа задач обычно собираются в результате *краудсорсинга* и представляют из себя тройки

$$(w_1, w_2, sim_{12}),$$

где $w_{1,2}$ – два слова из корпуса, а sim_{12} – оценка их сходства, определяемая *усреднением оценок нескольких людей* по шкале от 1 до 10.

Для оценки *способности векторного представления к отражению сходства понятий* каждой паре слов из тестового набора сопоставляется косинус угла между их соответствующими векторами. Данная величина объявляется ***машинной оценкой близости понятий***.

Далее подсчитывается корреляция Спирмена r_s между рангами, полученными по оценкам людей и оценкам векторного представления.

Соответственно, **чем выше показатель корреляции, тем более схожим образом машина и человек оценивают пары слов из тестового набора**.

Словесные аналогии Задача поиска словесных аналогий была поставлена в работе Mikolov et al. [2013b].

Тестовые наборы данной задачи представляют из себя наборы, состоящие из кортежей из двух пар слов $((w_1, w_2), (w_3, w_4))$.

Слова в парах являются аналогиями в определенном аспекте языка, например кортеж (“Англия”, “Лондон”), (“Франция”, “Париж”)) является примером отношения “Страна-Столица”.

Отношения делятся на *семантические* и *синтаксические*.

Примеры **семантических** отношений: “Страна-Валюта”, “Мужское-Женское”, “Синоним-Антоним”.

Примеры **синтаксических** отношений: “Прилагательное-Наречие”, “Прилагательное-Сравнительная степень”, “Настоящее-Прошедшее”, “Нация-Национальность”.

2.5.2 Оценки применимости в характерных задачах

Данные оценки определяют, **насколько качественно векторные представления позволяют решить некоторые из задач, относящихся к области обработки естественного языка**.

Определение эмоциональной окраски текста Целью данной задачи является **выявление субъективной** информации, представленной в текстах на естественном языке, **с целью дальнейшего определения ее *полярности* — оценки ее эмоциональной окраски**.

Такие эмоции как *злость*, *грусть*, *равнодушие*, *несчастность* и др. помечаются **негативными**.

Напротив, *счастье*, *радость*, *удовлетворение* и др. относятся к **позитивным**.

Например, утверждение "этот фильм радует прекрасной игрой актеров" относится к *позитивным*, в то время как "на этом фильме можно умереть со скуки" — к *негативным*.

Таким образом, в данной задаче **оценивается информативность векторных представлений слов для оценки эмоциональной окраски текстов**, т.е. насколько качественными признаками оказываются компоненты векторов и их различные комбинации.

Тестовые наборы представлены наборами текстов и меток уровня эмоциональной окраски от *очень негативного* до *очень позитивного*.

Определение категории понятий В данной задаче **оценивается информативность векторов представлений как признаков для определения категории входного понятия**.

Например, "холодильник" относится скорее к категории *Бытовая техника*, чем к *Пляжному отдыху*.

Тестовыми данными для данной задачи являются фразы и оценки их принадлежности к некоторым избранным категориям.

2.5.3 Russian Semantic Similarity Evaluation (RUSSE)

В этой секции приводится обзор комплекта задач *Russian Semantic Similarity Evaluation* для оценки качества векторов слов русского языка.

Данный комплект был представлен в работе Panchenko et al. [2016] и включает в себя следующие задачи.

Задача поиска семантических аналогий В этой задаче слова определяются как *семантические аналоги*, если они являются *синонимами* (близкими по значению понятиями), *гиперонимами* или *гипонимами* (одно из слов является обобщением другого).

Примеры:

- "авиация" и "авиа" — синонимы
- "абориген" и "индеец" — гипонимы
- "бизнесмен" и "владелец" — гиперонимы

Оценка качества векторных представлений определяется по двум тестовым наборам данных.

Первый набор составлен **вручную** и представляет из себя тройки (кортежи), состоящие из двух слов и целого числа от 0 до 5 — оценки их схожести. Метрикой оценивания выступает корреляция Спирмена между показателями схожести, полученными от людей и с помощью векторных представлений.

Данный набор состоит из 3-х частей:

1. переведенный на русский язык набор **WordSim353 Rel** с парами формата w_1 *относится к* w_2 *как* sim_{12}, всего 250 кортежей
2. переведенный на русский язык набор **WordSim353 Sim** с парами формата w_1 *по значению схоже с* w_2 *примерно на* sim_{12}, всего 202 кортежа
3. **RusseHJ** — набор понятий и оценок, выставленных людьми, всего 333 кортежа

Наборы **WordSim353** были представлены в работе Mikolov et al. [2013a].

Второй набор сформирован на основе тезауруса **РуТез-2.0**, оценка на нем определяется следующим образом:

1. выбирается некоторое понятие w из тезауруса, $a_{1,\dots,k}$ семантических аналогий к нему (понятие *аналогия* было определено выше) и $r_{1,\dots,k}$ случайных слов
2. для каждой пары $(w, a_1)_1, (w, a_2)_2, \dots, (w, a_k)_k, (w, r_1)_{k+1}, \dots, (w, a_k)_{2k}$ подсчитывается косинусная мера sim_i векторов соответствующих слов
3. берутся k пар $(w, c_{1,\dots,k})$ с наибольшими значениями $sim_\cdot$ и считается отношение количества пар $(w, a_{1,\dots,k})$ среди отобранных к k
4. далее полученные величины, называемые *точностью*, **усредняются по всем тестовым понятиям** и получается итоговая оценка векторного представления

Размер второго набора — 9549 кортежей.

Соответственно, **чем выше каждая из двух получаемых оценок, тем качественнее векторное представление отражает семантические аналогии русского языка**.

Задача поиска ассоциаций В данной задаче два слова считаются *схожими*, если второе является ассоциацией к первому. Под *ассоциацией* понимается первая пришедшая на ум опрашиваемого человека ассоциация, данная по первому слову (по *стимулу*).

Для построение тестового набора были использованы результаты двух крупномасштабных экспериментов:

1. *Russian Associative Thesaurus* (проводился в *оффлайн* режиме)
2. *Sociation.org* (*онлайн* эксперимент)

Цель задачи — **поиск когнитивных ассоциаций ко входному слову**. В экспериментах опрашиваемых просили дать первую пришедшую на ум ассоциацию на определенный стимул, и подсчитывали статистику полученных ответов.

Пример:

- 'время, деньги' – 14 ответов
- 'россия, страна' – 23 ответа
- 'рыба, жаренная' – 71 ответ

и т.д.

Таким образом, полученные когнитивные ассоциации могут быть как связаны семантическими отношениями (синонимы, антонимы, гиперонимы и т.д.), так и не иметь такого отношения.

Оценка качества в данной задаче определяется так же, как и в задаче поиска семантических аналогий, описанной ранее.

Из **Russian Associative Thesaurus** получено 1952 кортежа, из онлайн-эксперимента на **Sociation.org** — 3003.

2.6 Выводы

В результате исследования трудов предметной области данной работы можно сделать следующие основные выводы:

- в настоящее время проводится большое количество исследований в области создания и модификации моделей векторных представлений слов
- в основе построения данных моделей лежит ***распределительная гипотеза*** (Harris [1954])
- существующие модели на основе нейронных сетей, такие как *Skipgram*, *Continuos Bag-of-Words* и *FastText*, эффективно строят компактные представления слов, которые в простой алгебраической форме отражают семантические и синтаксические особенности понятий входного языка
- качество векторных представлений оценивается исходя из их применимости в характерных для области обработки естественного языка задачах
- качество получаемых представлений сильно зависит от входных данных и используемых гиперпараметров алгоритма их построения (Levy et al. [2015])
- существующие модели слабо учитывают морфологию входного языка
- ... и никак не используют накопленные априорные знания о нем.

Для решения поставленной в настоящей работе задачи (создание эффективных моделей для русского языка) оказываются полезными следующие замечания:

- русский язык обладает огромным морфологическим разнообразием
- существуют открытые базы знаний для русского языка, такие как **РуТез-2.0** и **Russian Distributional Thesaurus**, содержащие информацию, которая потенциально может быть использована для построения векторных представлений
- в открытом доступе имеется комплект тестовых задач **RUSSE** для оценки векторных представлений слов русского языка

3 Исследование и построение решения задачи

В данной главе вводится семейство моделей, использующих идеи, заложенные в алгоритм *FastText*.

Далее следует подробное описание введенного семейства.

Следом рассматриваются некоторые алгоритмы данного семества и описываются идеи, лежащие в основе каждого из них.

На следующем этапе приведено описание природы, свойств и процедуры преобработки данных, которые использовались для обучения алгоритмов.

В последней части находится описание методики проведенного тестирования и его результаты, а также анализ разработанных методов.

3.1 Основные идеи разработанных алгоритмов

Разработанные автором данной работы алгоритмы являются обобщением метода *FastText*, рассмотренного ранее в **Главе 2**.

Обобщение заключается в следующем: **слово представляется в виде совокупности своих свойств из некоторого количества *предметных областей.***

Под *предметными областями* здесь понимаются некоторые системы структурированных знаний об объектах, в качестве которых выступают слова языка.

Примерами предметных областей служат:

- морфемы слов языка — знанием о слове является его морфемный состав
- граф аналогий языка — знаниями о слове являются его связи с другими словами
- n-граммый состав слова — знаниями о слове являются его n-граммы

В дальшейшем под *свойством слова* будем понимать конкретную единицу знания из некоторой предметной области (например, каждая морфема слова является его *свойством*).

Определим теперь более формально описание введенного семейства моделей.

Пусть имеется словарь $W = \{w_i \mid i = 1,\ \ldots,\ N\}$ и набор из k предметных областей

$$G^* = \{G_i \mid i = 1,\ \ldots,\ k\}, \quad |G_i| = S_i$$

где

$$G_i = \{p_j^i(.) \mid j = 1,\ \ldots,\ S_i;\ p(w) : W \mapsto \{1, 0\}\}$$

— т.е. каждая область состоит из упорядоченного набора функций, называемых *предикатами*.

Каждому слову из лексикона ставится в соответствие набор

$$F^w = \{(i, j) \mid i = 1,\ \ldots,\ k;\ j = 1,\ \ldots,\ S_i;\ p_j^i(w) = 1\}$$

— т.е. набор индексов предикатов из каждой предметной области, которые возвращают **1** на слове w.

Соответственно, в процессе обучения каждая предметная область G_i представляет из себя матрицу синаптических весов, в которой каждому предикату p_j^i ставится в соответствие вектор $v(i, j)$.

Мера близости двух векторных представлений во введенном семействе моделей принимает вид:

$$s(w,c) = \langle v(w), v(c)\rangle + \sum_{(i,j)\in F^w} \langle v(i,j), v(c)\rangle \quad ,$$

т.е. **для слова w добавляется слагаемое, равное сумме скалярных произведений векторных представлений предикатов, которые обращаются в 1 на слове w, и контекстного слова.**

В итоге слово w представляется как усреднение по векторам своих свойств:

$$v(w) = \frac{1}{|F^w|} \sum_{(i,j)\in F^w} v(i,j)$$

Теперь рассмотрим модель *FastText*.

Пусть множество A — алфавит языка.

Модель *FastText* представляющет каждое слово множеством его n-грамм длины от l_1 до l_2. Перенумеруем все $S = \sum_{l=l_1}^{l_2} |A|^l$ возможных n-грамм длины от l_1 до l_2.

FastText **представляет из себя частный случай модели введенного семейства**, так как

$$G^* = \{G_1\}; \quad S_1 = S; \quad G_1 = \{p_i^1 \mid i = 1,\ \ldots,\ S_1\};$$

$$p_1^i(w) = \begin{cases} 1 & \text{если слово w содержит n-грамму под номером i} \\ 0 & \text{иначе} \end{cases}$$

Обобщение *FastText* до описанного выше семейства моделей позволяет ввести некоторые модификации архитектуры оригинальной сети для максимально эффективного использования информации, предоставляемой каждой предметной областью.

3.2 Модификации архитектуры нейронной сети

Базовую архитектуру модели *FastText* назовем *полносвязной сетью*.

Для извлечения наибольшей пользы из каждой предметной области автор предлагает следующие 3 модифицированных архитектуры базовой нейросети:

1. *полносвязная сеть с dropout*
2. *semi-boosting сеть*
3. *усредненная по приоритетам сеть*

Рассмотрим оригинальную и предложенные архитектуры.

3.2.1 Полносвязная сеть

Оригинальная архитектура модели *FastText*.

Все предметные области равнозначны, вектора предикатов обновляются одновременно на каждом шаге скользящего окна.

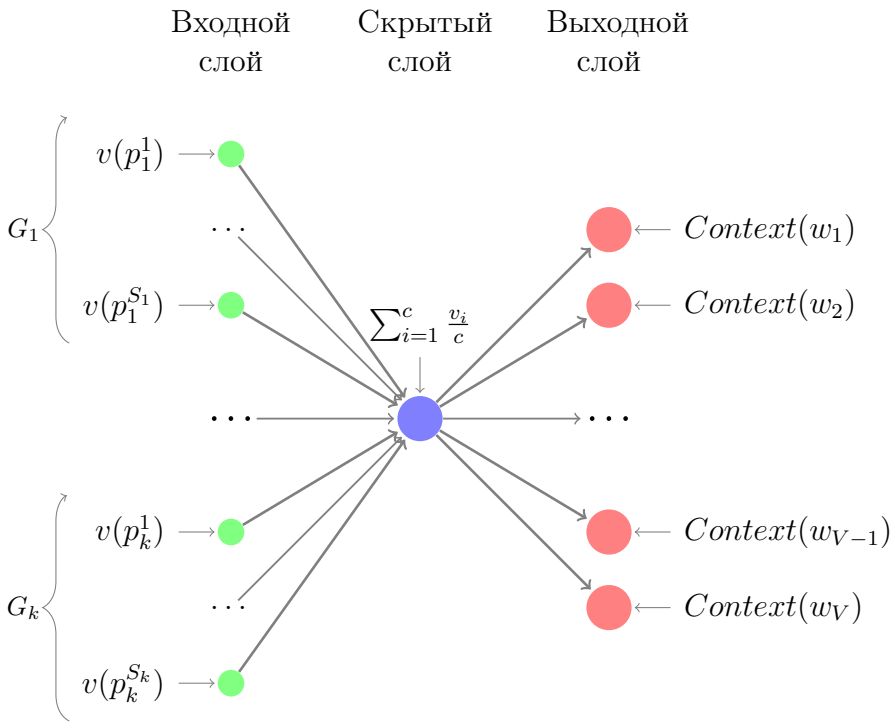


Рис. 9: Схема архитектуры полносвязной сети

3.2.2 Полносвязная сеть с dropout входного слоя

Dropout – это способ регуляризации нейронной сети путем предотвращения полной адаптации всех нейронов сети под входные данные.

Данный способ был предложен в работе Srivastava et al. [2014].

Идея заключается в том, что **в процессе обучения при проходе по сети каждый из нейронов деактивируется с некоторой вероятностью** p. Авторы концепции показывают, что при $p = \frac{1}{2}$ в результате получается усреднение параметров нейронной сети по всевозможным 2^N архитектурам, где N — количество нейронов в сети.

Кроме того, данная модификация существенно снижает время обучения сети.

Таким образом, архитектура сети остается полносвязной (рис. 9), но при каждом проходе в процессе обучения с вероятностью p каждое из свойств рассматриваемого слова может быть отброшено.

В данной работе $p = \frac{1}{2}$, как рекомендуют авторы в Srivastava et al. [2014].

3.2.3 Semi-boosting сеть

Данная модификация работает по принципу градиентного бустинга, описанного в работе Breiman [1997].

На каждом шаге скользящего окна совершается k проходов, на проходе i берутся свойства из $G_i(w)$.

Таким образом получается, что **обучаются** k **моделей, каждая из которых исправляет коллективную ошибку предшествующих** (по аналогии с градиентным бустингом).

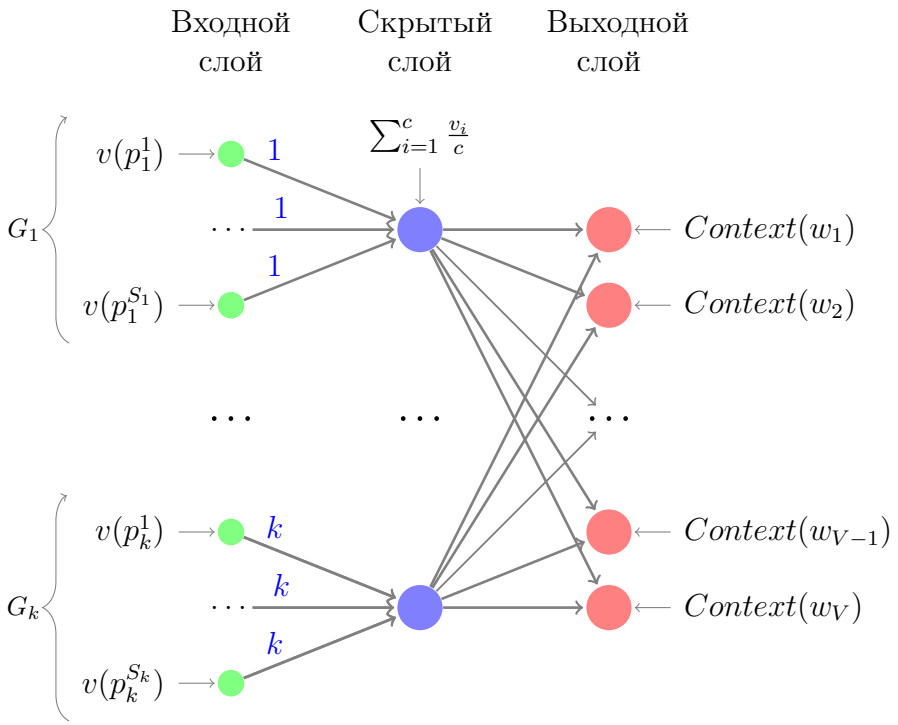


Рис. 10: Схема архитектуры semi-boosting сети

На схеме (рис. 10) в момент i являются активными только связи с меткой i.

3.2.4 Усредненная по приоритетам сеть

В данной модификации на каждом шаге скользящего окна совершается k проходов по сети, на каждом из которых в множество свойств G^* включаются свойства из не выбранной ранее предметной области. Таким образом, первый проход совершается со свойствами одной области, второй – двух областей и т.д.

Порядок включения предметных областей определяется случайно, все области равнозначны, за счет чего приоритеты областей усредняются в ходе обучения.

Схема модели совпадает со схемой *semi-boosting* (рис. 10), только в этот раз на шаге i множество свойств $G^* = \cup_{j=1}^{i} G_j$, а начальный порядок предметных областей $G_{...}$ определяется случайным образом.

В итоге получается модель, усредненная по взвешенным (так как первая по порядку область внесет вклад k раз, вторая $k-1$ раз и т. д.) приоритетам предметных областей.

3.3 Предметные области

В данной секции приводится описание выбранных предметных областей для русского языка.

Автор предлагает ввести в рассмотрение 4 предметных области (в дополнение к двум, одна из которых содержит само слово, а вторая — его n-граммы определенных размеров):

1. морфемы слов, полученные с помощью морфологического анализатора **pymorphy2**
2. морфемы слов, полученные эвристически
3. аналогии, полученные из тезауруса **Russian distributional thesaurus**
4. синонимы, полученные из тезауруса **PyТез-2.0**

3.3.1 Морфемы, полученные с помощью морфологического анализатора pymorphy2

Данный морфологический анализатор представлен в работе Korobov [2015].

Инструмент **pymorphy2** написан на языке *Python* и умеет:

1. приводить слово к нормальной форме
2. ставить слово в нужную форму падежа, числа и др.
3. возвращать грамматическую информацию о слове
4. возвращать всевозможные формы слова

В ходе работы анализатора используется словарь **OpenCorpora**.

Свойства данной предметной области представляют из себя информативные лексемы, которые получаются для каждого слова лексикона по следующему алгоритму:

1. берутся всевозможные формы слова с помощью анализатора *pymorphy2*
2. выделяются мультимножество всевозможных n-грамм $S = \{s_1, \ldots, s_n\}$ каждой формы слова
3. для каждой n-граммы s_i определяется рейтинг по формуле $rank(s_i) = length(s_i) * \#(s_i)$, где $\#(s_i)$ — количество повторений данной n-граммы в S
4. берутся k n-грамм с максимальным рейтингом

Таким образом, **каждому слову сопоставляется k самых частотных n-грамм среди всех лексических форм данного слова**.

Число k является параметром алгоритма.

Пример получаемых данных:

1. *понадобиться: понадоб, понадоби, понадо, надо, онадоби, ...*
2. *появление: появлен, появле, оявлен, появл, ...*
3. *смешать: смеша, смеш, меша, еша, меш*

и т.д.

3.3.2 Эвристические морфемы

Автор данной работы собрал из свободных источников коллекции всевозможных *приставок*, *окончаний* и *суффиксов* слов русского языка.

Каждое слово лексикона входного корпуса разбирается на морфемы по следующему алгоритму:

1. в слове выделяются флексии, содержащиеся в коллекции. Если такие нашлись, то берется самое длинное и удаляется из слова
2. по тому же принципу в слове после шага 1) выделяется и удаляется суффикс
3. та же процедура, только для приставок, повторяется для слова, полученного после шага 2)

4. оставшаяся часть слова считается корнем

На каждом шаге алгоритма проверяется, что длина слова не становится меньше 3.

Пример получаемых данных:

1. *разглашать: раз-глаш-ать*
2. *необжитой: необ-жи-той*
3. *обозревать: обо-зрев-ать*

и т.д.

3.3.3 Аналогии из Russian distributional thesaurus

Из тезауруса **RDT** для каждого слова из входного лексикона берутся k ближайших аналогий.

Число k — параметр алгоритма.

Пример получаемых данных:

1. *открытка: записка, новогодний, поздравительный, сувенир, ...*
2. *производная: производный, константа, функция, компонента, ...*
3. *проприетарный: дистрибутив, файлообменник, портировать, браузерный, ...*

и т.д.

3.3.4 Синонимы из тезауруса РуТез-2.0

Из данного тезауруса берутся группы синонимов.

Слова, не содержащиеся во входном лексиконе, фильтруются.

Каждому слову из группы синонимов сопоставляются остальные слова из той же группы.

Пример получаемых данных:

1. *бой: давать, бить, вмазывать, влеплять, побивать, ударный, ...*
2. *публикование: обнародование, опубликовывать, помещать, печататься, ...*
3. *невредимый: целый, неповрежденный, целость*

и т.д.

3.4 Выводы

В данной главе определено семейство моделей векторного представления слов, обобщающее идеи алгоритмов *Word2Vec* и *FastText*.

Представлены некоторые модели из введенного семейства, обладающие следующими **двумя ключевыми свойствами**:

1. Эффективность — модели успешно извлекают пользу из априорной информации (такой, как морфемный состав слова)

2. Расширяемость — модели позволяют использовать любую полезную информацию о словах (единственное требование, не ограничивающее общности, — информация должна быть приведена к специальному формату).

Таким образом, новое семейство моделей позволяет соединить знания о языке, собранные человеком (например, в открытых базах знаний), и знания, извлеченные из входного корпуса машиной, для получения наиболее качественных векторных представлений слов русского языка.

4 Описание практической части

В данной главе приведено описание реализации разработанных алгоритмов, а именно: диаграммы классов и настраиваемые параметры.

Также для каждой из представленных моделей выведены оценки сложности и потребляемой памяти.

4.1 Структура реализации

Реализация выполнена на языке программирования *C++* с использованием объектно-ориентированного подхода.

Далее приведены полученные в результате декомпозиции классы и их роли в системе.

Args — класс отвечает за разбор, валидацию и хранение переданных пользователем в командной строке аргументов.

Vector — реализация вектора, включает в себя содержимое вектора и набор операций с ним (умножение на другой вектор, умножение на константу и др.).

Matrix — реализация матрицы, включает в себя содержимое матрицы и набор операций с ней (умножение строки на вектор, проекция строки, умножение на константу и др.).

Dictionary — разделяемый словарь, отвечает за хранение и доступ ко всей информации о корпусе (лексикон, свойства слов и т.д.). Построен на ассоциативных словарях и хеш-таблицах из библиотеки *STL*.

SlaveModel — реализует модель нейросети и алгоритм обратного распространения ошибки. Каждая нить представляет из себя SlaveModel.

MasterModel — реализует мастер-модель нейросети, отвечает создание и синхронизацию нитей (SlaveModel) в течение всего процесса обучения. Имеются методы для загрузки и сохранения обученных моделей.

4.2 Поддерживаемые параметры обучения

Реализация поддерживет указание пользователем в командной строке следующих параметров:

- используемую архитектуру сети
- размерность выходных представлений
- размер скользящего окна
- количество шумных слов для данного
- длина n-грамм
- learning rate

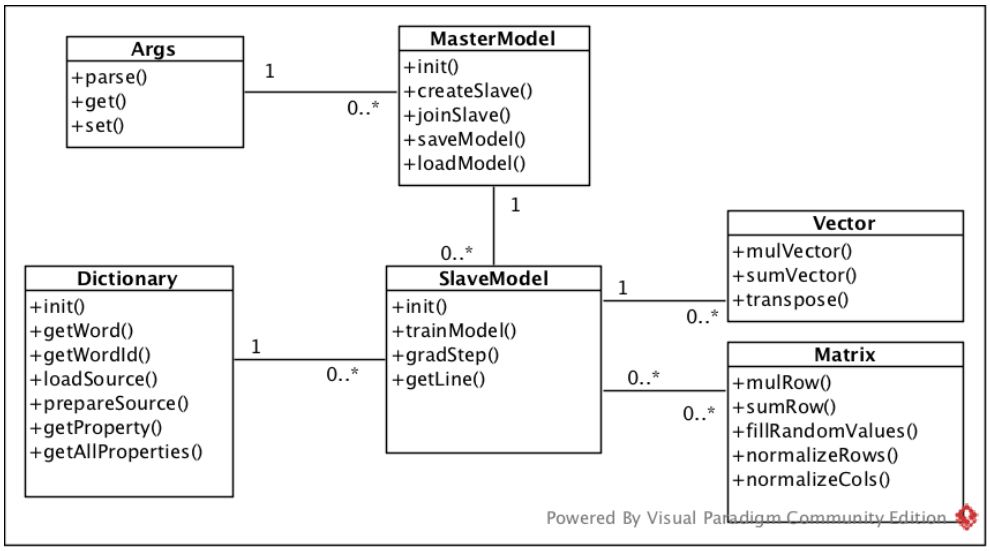


Рис. 11: Диаграмма классов

- количество эпох
- количество шагов, через которое происходит линейное подавление learning rate
- максимальное количество свойств для слова из каждой предметной области
- количество нитей
- путь к файлам с информацией о предметных областях

4.3 Оценки сложности и потребляемой памяти

Далее будут использованы следующие обозначения:

- f — количество используемых предметных областей
- k — максимальное количество свойств каждой области для каждого слова
- l — ограничение на длину названия каждого свойства
- N — размерность векторов
- V — размер лексикона входного корпуса
- T — количество токенов входного корпуса

Оценки сложности и потребляемой памяти реализации представленных в работе моделей сведены в *таблицу 2*:

Модель	Сложность	Потребляемая память
fully-connected	$O(kfNT)$	$O(kfV(N+l))$
fully-connected + dropout	$O(kfNT)$	$O(kfV(N+l))$
semi-boosting	$O(kfNT)$	$O(kfV(N+l))$
mean-priority	$O(k^2fNT)$	$O(kfV(N+l))$

Таблица 1: Оценки сложности и потребляемой памяти моделей

4.4 Оценка качества разработанных моделей

В следующей секции описаны входные данные для обучения сравниваемых моделей, а также задачи, на которых оценивалось качество выходных представлений.

В конце приведена сводная таблица с результатами работы каждой модели на выбранных задачах.

4.4.1 Подготовка входных данных

Входными данными для обучения были взяты примерно 12 млн сообщений из различных открытых сообществ социальной сети "Вконтакте".

Сообщение представляет из себя от 1 до 10 предложений на русском языке.

Процесс предобработки состоит из следующих шагов:

1. замена знаков препинания на пробелы
2. приведение слов в нижний регистр
3. замена всех числительных на "1"
4. лемматизация с помощью морфологического анализатора **Yandex Mystem 3**
5. создание словаря, содержащего **100 тыс. наиболее частотных уникальных понятий**
6. фильтрация слов корпуса, которые не вошли в сформированный словарь

Решение проводить лемматизацию — приведение каждого слова в нормальную форму — было принято исходя из работы Levy et al. [2015], где было показано, что данная процедура:

- уменьшает размер словаря, что увеличивает производительность
- повышает качество получаемых представлений

Предварительные эксперименты с русским языком подтвердили данные тезисы.

Принципы работы морфологического анализатора **Yandex Mystem 3** описаны в работе Segalovich [2003], где также приведены результаты экспериментов.

Данный инструмент показал лучшие результаты для русского языка среди популярных стеммеров и лемматизаторов, вследствие чего в данной работе было принято решение использовать именно его.

В открытом доступе имеется обертка для данного анализатора pymystem3[1] , реализованная на языке программирования *Python*.

Фильтрация словаря позволяет сильно увеличить производительность алгоритмов, а также уменьшить т.н. "шумы" во входных данных — слова с ошибками, имена и названия, жаргонизмы и т.д.

Применительно к данному корпусу сообщений пользователей социальной сети было отфильтровано $\approx 90\%$ всех понятий, частотный порог для вхождения в словарь оказался равен 20. Однако корпус уменьшился всего на 1.4%, что означает потерю **пренебрежимо малой части** потенциально полезной информации в обмен на существенное увеличение производительности.

[1] github.com/Digsolab/pymystem3

4.4.2 Методика тестирования

Качество выходных представлений моделей тестировалось с использованием комплекта **RUSSE**, описанного ранее в **Главе 2**.

Данное решения было принято по следующим причинам:

1. данный комплект позволяет всесторонне оценить качество векторов, так как включает в себя несколько тестовых задач (6 штук)

2. в комплекте представлены почти все существующие на данный момент задачи для русского языка

3. в свободном доступе имеется обертка[1], написанная на *Python*

Также было принято решение провести две серии тестов разработанных моделей:

1. **с использованием всех предметных областей, описанных ранее** — для оценки способности архитектуры разработанных моделей к эффективному использованию предоставленной потенциально полезной информации

2. **без использования RDT и РуТез-2.0** — так как данные источники частично использовались для подготовки задач разработчиками комплекта *RUSSE*

4.4.3 Использованные параметры моделей

Общие для всех моделей параметры были взяты по умолчанию (предложенные авторами базовых моделей), а именно:

- размерность выходных представлений – 300
- размер скользящего окна – 5
- количество шумных слов для данного – 5
- длина n-грамм – от 3 до 6
- learning rate – 0.025
- количество эпох – 1
- линейное подавление learning rate через каждые 100 шагов скользящего окна

Каждая модель из разработанных использует **не более 10 самых частотных своих свойств** из каждой предметной области. Данное ограничение позволяет не учитывать уникальные свойства слов (так как они вносят вклад только в данное слово и не имеют полезной информации для других) и существенно повысить производительность (в ≈ 2 раза по скорости и потребляемой памяти).

4.4.4 Результаты

Итоговые оценки качества, усредненные по 5 независимым выходным представлениям каждой из моделей, представлены в *таблице 1.*

Модели, не использующие в работе базы знаний **RDT** и **РуТез-2.0**, содержат в названии “no thes”

[1]github.com/nlpub/russe-evaluation

Модель	*WS353 rel*	*WS353 sim*	*HumJudge*	*RuTes sem*	*AssocThes*	*AssocOnline*
CBoW	0.5951	0.6869	0.6782	0.7120	0.5184	0.8554
SGNS	0.6327	0.7249	0.6967	0.7241	0.5256	0.8581
FastText	0.6021	0.7264	0.6896	0.7130	0.5287	0.8434
fully-conn.	0.6344	0.7607	0.7237	0.7315	0.5315	0.8604
fully-conn. (no thes.)	0.6359	0.7557	0.7240	0.7240	0.5255	0.8584
f-c + dropout	0.5972	0.7154	0.7175	0.7160	0.5183	0.8481
f-c + dropout (no thes.)	0.5809	0.6362	0.6832	0.6718	0.5134	0.8215
semi-boosting	**0.6594**	**0.7855**	0.7642	**0.7405**	**0.5420**	**0.8701**
semi-boosting (no thes.)	0.6538	0.7196	**0.7704**	0.7260	0.5400	0.8634
mean-prior.	0.5907	0.6695	0.6632	0.7009	0.5010	0.8315
mean-prior. (no thes.)	0.5847	0.6564	0.6248	0.6843	0.5051	0.8241

Таблица 2: Оценка выходных векторных представлений слов

4.5 Выводы

Реализация моделей имеет обладает следующими основными свойствами:

- Реализация выполнена на языке *C++*
- Единый код для всех моделей
- Архитектура модели выбирается пользователем
- Параметры модели задаются пользователем
- Обучение происходит параллельно на нитях (threads)
- Параллелизация *по данным*
- *Вертикальная* и *горизонтальная* масштабируемость
- Расширяемость — можно подключать любые предметные области (требуется соблюсти формат).

Оценки времени работы и потребляемой памяти позволяют обучать модели на среднестатистических вычислительных машинах (*не рекомендуется брать корпус размером более* 20 *млн токенов*).

Разработанные модели для русского языка **показывают лучшее качество** среди самых распространенных на данный момент методов.

Заключение

В рамках настоящей выпускной квалификационной работы были получены следующие результаты:

1. Проведен анализ существующих методов построения векторных представлений слов, составлен их подробный обзор

2. Введено семейство моделей векторного представления слов, обобщающее идеи существующих алгоритмов

3. Разработаны и реализованы несколько методов векторизации, показывающих одни из лучших результатов в задачах обработки русского языка

4. Приведено полное описание новых методов, для каждого получены оценки сложности и потребляемой памяти.

Список литературы

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

Leo Breiman. Arcing the edge. Technical report, 1997.

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th Annual Meeting on Association for Computational Linguistics*, ACL '89, pages 76–83, Stroudsburg, PA, USA, 1989. Association for Computational Linguistics. doi: 10.3115/981623.981633. URL `http://dx.doi.org/10.3115/981623.981633`.

Manaal Faruqui. *Diverse Context for Learning Word Representations*. PhD thesis, Carnegie Mellon University, 2016.

Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014. URL `http://arxiv.org/abs/1402.3722`.

Zellig S. Harris. Distributional structure. *WORD*, 10(2-3):146–162, 1954. doi: 10.1080/00437956.1954.11659520. URL `http://dx.doi.org/10.1080/00437956.1954.11659520`.

Simon Haykin. *Neural Networks: A Comprehensive Foundation (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2007. ISBN 0131471392.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

Mikhail Korobov. Morphological analyzer and generator for russian and ukrainian languages. In Mikhail Yu. Khachay, Natalia Konstantinova, Alexander Panchenko, Dmitry I. Ignatov, and Valeri G. Labunets, editors, *Analysis of Images, Social Networks and Texts*, volume 542 of *Communications in Computer and Information Science*, pages 320–332. Springer International Publishing, 2015. ISBN 978-3-319-26122-5. doi: 10.1007/978-3-319-26123-2_31. URL `http://dx.doi.org/10.1007/978-3-319-26123-2_31`.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf`.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015. ISSN 2307-387X. URL `https://transacl.org/ojs/index.php/tacl/article/view/570`.

Natalia Loukachevitch and Boris Dobrov. Ruthes linguistic ontology vs. russian wordnets. In *In Proceedings of Global Wordnet Conference*, 2013.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013a. URL `http://arxiv.org/abs/1301.3781`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013b. URL `http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality`.

A. Panchenko, D. Ustalov, N. Arefyev, D. Paperno, N. Konstantinova, N. Loukachevitch, and C. Biemann. Human and Machine Judgements for Russian Semantic Relatedness. In *Analysis of Images, Social Networks and Texts: 5th International Conference, AIST 2016, Yekaterinburg, Russia, April 7-9, 2016, Revised Selected Papers*, pages 221–235, Yekaterinburg, Russia, 2017. Springer International Publishing. ISBN 978-3-319-52920-2. doi: 10.1007/978-3-319-52920-2_21.

Alexander Panchenko, Dmitry Ustalov, Nikolay Arefyev, Denis Paperno, Natalia Konstantinova, Natalia Loukachevitch, and Chris Biemann. Human and machine judgements for russian semantic relatedness. In *Analysis of Images, Social Networks and Texts (AIST'2016)*. Springer, 2016.

Xin Rong. word2vec parameter learning explained. *CoRR*, abs/1411.2738, 2014. URL `http://arxiv.org/abs/1411.2738`.

Ilya Segalovich. A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine. In *MLMTA*, 2003.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1858681.1858721`.