МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное
учреждение высшего образования
ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт математики, механики и компьютерных наук
имени И. И. Воровича

Направление подготовки
Прикладная математика и информатика

Кафедра информатики и вычислительного эксперимента

АВТОМАТИЧЕСКОЕ ФОРМИРОВАНИЕ ЗАДАНИЙ ПО ГРАММАТИКЕ АНГЛИЙСКОГО ЯЗЫКА

Выпускная квалификационная работа
на степень бакалавра

Студента 4 курса
А. А. Мезги

Научный руководитель:
кандидат физико-математических наук, доцент В. А. Нестеренко

Ростов-на-Дону
2020

Содержание

Введение

Принято считать, что история развития обработки естественного языка берёт своё начало в 1950-х годах, когда Алан Тьюринг опубликовал свою работу «Вычислительные машины и разум» (англ. Computing Machinery and Intelligence [1]), где был представлен тест Тьюринга.

В настоящий момент в рамках обработки естественного языка (англ. Natural Language Processing), далее NLP, стоящей на пересечении таких компьютерных наук, как машинное обучение и компьютерная лингвистика, решаются проблемы анализа, понимания и извлечения смысла из естественной человеческой речи.

Актуальность темы дипломной работы обусловлена стремительным ростом рынка онлайн-образования и тем, что изучение иностранных языков является приоритетным направлением для потребителей. Создатели онлайн-платформ часто называют персонализацию обучения отличительной особенностью их курсов. Однако несмотря на большую гибкость по сравнению с классическим методом изучения иностранного языка, некоторые материалы могут быть устаревшими и неактуальными. Это связано с тем, что процесс формирования упражнений является трудозатратным ввиду задействования человеческого труда и отсутствия программных средств для решения этой задачи.

Рассмотрим основные этапы изучения грамматических явлений при изучении нового языка:

- Presentation
  Демонстрация примеров и структуры грамматической конструкции.
- Pattern Recognition
  Самостоятельный поиск грамматических конструкций в исходном тексте.

- Controlled Practice
  Построение изучаемых грамматических конструкций в ограниченной форме.
- Semi-Controlled Practice
  Персонализация языка с упором на изучаемые грамматические явления.
- Free Practice
  Свободное использование языка.

Цель исследования - проектирование и реализация системы автоматического формирования заданий видов *Controlled Practice* и *Semi-Controlled Practice*, которая позволит сократить время подготовки методических материалов и дать возможность большей персонализации траектории обучения.

Для достижения результата были поставлены следующие задачи:

- изучить методы Data Mining и Machine Learning в области обработки естественного языка;
- изучить методы векторного представления слов;
- разработать метод предварительной очистки текста;
- изучить модели машинного обучения для извлечения необходимых признаков;
- разработать алгоритм поиска грамматических конструкций;
- разработать алгоритм формирования заданий на основе найденных конструкций;
- реализовать развертывание сервера;
- разработать web-платформу для взаимодействия с пользователями;

- проанализировать возможности повышения производительности системы;
- проанализировать возможности повышения точности обнаружения грамматических конструкций;
- реализовать базу данных результатов пользователей для ведения статистики.

Объектом исследования дипломной работы является применение компьютерной лингвистики в области онлайн-образования.

Предметом исследования дипломной работы является построение алгоритмов поиска грамматических конструкций и формирования на их основе упражнений по грамматике английского языка с использованием методов компьютерной лингвистики и Machine Learning.

1. Теоретические основы задачи обработки естественного языка

Задача обработки естественного языка это целый ряд теоретико-мотивированных вычислительных методов, которые позволяют производить анализ человеческого языка. В рамках NLP решается большое количество задач, затрагивающих различные уровни- начиная от определения частей речи, заканчивая созданием диалоговых систем.

Долгое время в задачах NLP применялись модели поверхностного обучения, такие как SVM (англ. support vector machine), обученные на разреженных данных, представленных в пространстве высокой размерности, что не позволяло достичь достаточной точности и требовало значительных вычислительных ресурсов. Однако в последние годы были разработаны новые методы векторного представления слов, что позволило избежать экспоненциального роста размерности. Это помогло сократить время обучения сетей и перейти к методам, основанным на глубоком обучении, которые обеспечивают автоматическое обучение признакам. Это существенно отличает новые архитектуры от тех, что применялись раньше, так как для них больше не требуется ручное конструирование признаков.

Одна из первых архитектур глубокого обучения в области NLP была продемонстрирована Ронаном Коллобертом и Джейсоном Уэстоном [2]. На тот момент она превосходила большую часть уже существующих моделей в таких задачах, как поиск именованных сущностей и др. С того времени появилось большое количество алгоритмов и моделей, решающих сложные задачи обработки естественного языка.

1.1. Стандартные задачи NLP

В области обработки естественного языка выделяют 4 базовые задачи: *POS tagging (POS), Chunking, Named Entity Recognition (NER), Semantic Role Labeling (SRL)*. Они лежат в основе более высокоуровневых задач, в том числе в задаче автоматизации создания заданий по грамматике. Рассмотрим каждую из них в отдельности.

POS tagging — задача маркировки каждого слова в соответствии с принадлежностью его к той или иной части речи: имя существительное, имя прилагательное, глагол и т.д.

POS классификаторы часто базируются на классификаторах, которые обучены на текстовых "окнах". После этого результат передается алгоритму двунаправленного декодирования [3]. В качестве признаков может браться контекст слова и некоторые созданные вручную признаки. Для проверки качества работы модели в задаче POS tagging зачастую применяется размеченный корпус текстов Wall Street Journal (WSJ).

Chunking — задача поверхностно-синтаксического анализа. В ней производится анализ предложения, который сначала индексирует составные части предложений (существительные, глаголы и т.д.), а затем связывает их в единицы более высокого порядка, такие как именные или глагольные группы и т.д..

Традиционные алгоритмы поверхностно-синтаксического анализа используют поиск по шаблонам (например, регулярные выражения). Алгоритмы с использованием машинного обучения могут учитывать контекстную информацию, что дает возможность улучшить точность восстановления семантических связей.

Named Entity Recognition — задача распознавания именованных сущностей. Является подзадачей Data Mining, в рамках которой необ-

ходимо найти и классифицировать именованные сущности в неструктурированном тексте по заранее заданным категориям, таким как имена людей, названия организаций, количество и т.д.

Распознавание именованных сущностей часто разбивается на две подзадачи: выявление имен и их классификация по типу сущности. Первая фаза рассматривается как проблема сегментации: зачастую именованные сущности рассматриваются как непрерывные промежутки токенов без вложенностей. Для второй фазы требуется определение онтологии, благодаря которой формируются категории вещей.

Существует несколько иерархий именованных сущностей. Так, категории BNN используются в вопросно-ответных системах и включают в себя 29 типов и 64 подтипа. В свою очередь иерархия Sekine является расширенной и состоит из 200 подтипов, а в 2011 Риттер представил иерархию, которая основана на общих типах объектов Freebase.

Системы NER могут быть основаны как на лингвистических методах грамматики, так и на статистических моделях. Несмотря на то, что системы, созданные вручную, имеют большую точность, они требуют месяцев работы профессиональных лингвистов. Подход с использованием методов машинного обучения сейчас является приоритетным, однако является менее устойчивым и требует большого объема аннотированных данных на этапе обучения моделей.

Semantic Role Labeling — задача маркировки семантических ролей. Представляет собой процесс присваивания меток словам или фразам, которые характеризуют их семантические роли в предложении. Результатом работы является порождение поверхностной интерпретации, основанной на теории семантических ролей.

Процесс часто делят на два этапа: обнаружение семантических аргументов, связанных с предикатом или глаголом, и классификации

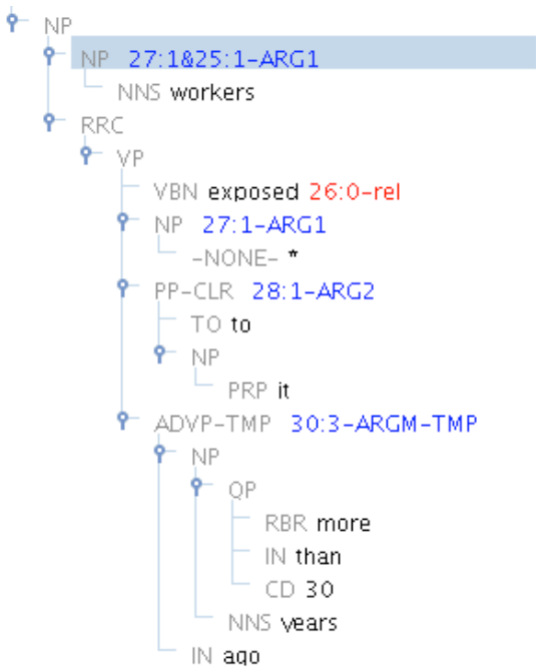

Рисунок 1 — Пример размеченных данных *PropBank*

по их семантическим ролям. Процесс часто делят на несколько этапов: обнаружение и определение значения целевых предикатов, обнаружение и дальнейшую классификацию актантов.

Для реализации автоматической маркировки используются статистические модели, обученные на размеченных корпусах текстов, содержащих информацию о семантических ролях и предикатах. Примером такого корпуса выступает *PropBank*, который был получен добавлением вручную созданных семантических аннотаций в корпус *Penn Treebank*. Пример такой разметки можно увидеть на Рис. 1

Однако при использовании статистических моделей существует недостаток, который проявляется в зависимости от выбранного аннотированного ресурса. Кроме того, создание семантической разметки является трудозатратной и плохо формализуемой задачей. Из-за этого может проявляться некорректная работа на новых данных. Этот аспект принято называть проблемой доменной специфичности SRL.

1.2. Word Embeddings

Статистические модели стали основным инструментом в задачах NLP, однако в начале они страдали от т.н. проклятия размерности [4]. Это повлекло развитие алгоритмов, которые позволили бы представлять слова в низкоразмерном пространстве.

Стоит сказать, что на данный момент нет общепризнанного перевода термина embedding (от англ. ‘вложение’), поэтому будет использоваться англицизм.

Embedding представляет собой преобразование некой сущности в числовой вектор. Далее будут рассмотрены основные подходы, применяющиеся для решения данной задачи.

1.2.1. One-hot encoding

Одним из первых решений задачи векторного представления слов был так называемый унитарный код. Он представлял собой вектор длины n, которая определяется количеством слов некоторого словаря, содержащий (n-1) нулей и 1 единицу. Индекс значащей единицы соответствовал расположению слова в данном словаре —см. таблицу 1.

Несмотря на то, что такая архитектура позволяет решить проблему кодирования слов, она обладает рядом существенных недостатков:

- При добавлении нового слова в середину существующего словаря есть необходимость заново проводить нумерацию его элементов.
- Происходит быстрый рост размерности представления текстов. Например для текста из 9 уникальных слов требуется матрица 9x9
- Данный метод не предоставляет информации о семантической близости слов. Данный аспект связан с тем, что написание слов

Таблица 1 — One-Hot Encoded векторы

Vocabulary	1	2	3	4	5	6	7	8	9	10
bag	1	0	0	0	0	0	0	0	0	0
words	0	1	0	0	0	0	0	0	0	0
model	0	0	1	0	0	0	0	0	0	0
way	0	0	0	1	0	0	0	0	0	0
representing	0	0	0	0	1	0	0	0	0	0
text	0	0	0	0	0	1	0	0	0	0
data	0	0	0	0	0	0	1	0	0	0
simple	0	0	0	0	0	0	0	1	0	0
understand	0	0	0	0	0	0	0	0	1	0
implement	0	0	0	0	0	0	0	0	0	1

не имеет непосредственной связи с объектами, которые они описывают [2.12. Фердинанд де Соссюр (1827–1913). Лингвистический структурализм].

1.2.2. Bag of Words

На основе кодирования слов унитарным кодом, рассмотренным в пункте 1.1.1, был предложен более экономичный вариант представления текстов, называемый “мешком слов” (от англ. Bag of words).

Алгоритм построения такого представления содержит следующие основные этапы:

- Предварительное создание словаря методом OHE.
- Кодирование слов, содержащихся в тексте.
- Сложение всех полученных one-hot векторов.

На выходе получим числовой вектор, который описывает информацию о количестве различных слов в исходном тексте. Такая модель не сохраняет структуру входных данных, в связи с чем теряется информация о взаимном расположении слов. Тем не менее, применяя

Таблица 2 — Терм-документная таблица

Vocabulary	Documents	
	C1	C2
bag	1	1
words	1	1
model	1	1
way	1	0
representing	1	0
text	1	0
data	1	0
simple	0	1
understand	0	1
implement	0	1

данный подход можно сравнивать тексты путем сравнения выходных векторов. Примером такой метрики является косинусная мера:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{1}$$

A_i, B_i – компоненты векторов $\mathbf{A}$ и $\mathbf{B}$ соответственно.

Рассмотрим этот процесс имплементации алгоритма.

1. Зададим экземпляры текста.
   C1 = *“The Bag of Words model is a way of representing text data.”*
   C2 = *“The Bag of Words model is simple to understand and implement.”*

2. Очистим тексты от пунктуационных символов и ‘стоп-слов’[1], которые не несут смысловой нагрузки.

3. Сформируем словарь и закодируем его методом ОНЕ (см. таблицу 1).

[1] the, of, is, a, to, and

4. Для каждого предложения сложим One-Hot векторы слов, которые входят в их составы (см. таблицу 2).

5. Применим косинусную меру (1) к полученным векторам, которые количественно описывают исходные тексты.

$$\text{similarity} = \cos(\theta) = \frac{3}{\sqrt{7} \cdot \sqrt{6}} = 0.46291$$

Кроме того, полученную Таблицу 2 с зависимостью “слово-документ” можно представить в виде произведения матриц “слово-тема” и “тема-документ”. Для этого можно использовать SVD-разложение:

$$(\mathbf{t}_i^T) \rightarrow \begin{matrix} & (\mathbf{d}_j) \\ & \downarrow \end{matrix} \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \\ \mathbf{u}_1 \\ \\ \end{bmatrix} & \dots & \begin{bmatrix} \\ \mathbf{u}_l \\ \\ \end{bmatrix} \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} \cdot \begin{bmatrix} \begin{bmatrix} & \mathbf{v}_1 & \end{bmatrix} \\ \vdots \\ \begin{bmatrix} & \mathbf{v}_l & \end{bmatrix} \end{bmatrix} \tag{2}$$

$\mathbf{t}_i$ – слово, $\mathbf{d}_i$ – документ.

Применим такое разложение для текстов из примера и визуализируем (см. рис. 2). Как можно видеть, получаемые результаты имеют сильную зависимость от корпуса, к которому применяется разложение.

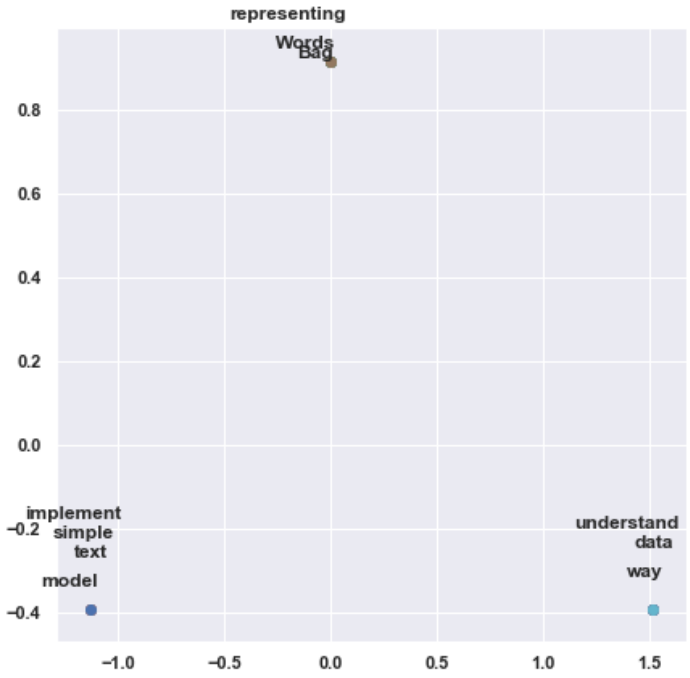

Рисунок 2 — SVD разложение для C1 и C2

1.2.3. Word2vec

Из-за ряда недостатков, которыми обладали традиционные алгоритмы, исследования в области представлений слов продолжились. В 2013 году Томаш Миколов представил[5] подход к кодированию слов, который решал целый ряд проблем, присущих другим моделям, в том числе рост размерности векторного пространства и невозможность сохранять семантическую близость.

Им было представлено 2 подхода, которые он назвал continuous bag-of-words (CBOW) и skip-gram. Они позволяют строить высококачественные распределенные векторные представлений.

Между представленными моделями есть существенное отличие. CBOW вычисляет условную вероятность появления целевого слова исходя из его контекста, который лежит в окне размера k. В свою очередь Skip-gram предсказывает слова контекста по центральному сло-

ву. Предполагается, что контекстные слова расположены симметрично целевым словам на расстоянии, равном размеру окна в обоих направлениях. Схематичное представление данных методов показано на рис. 3

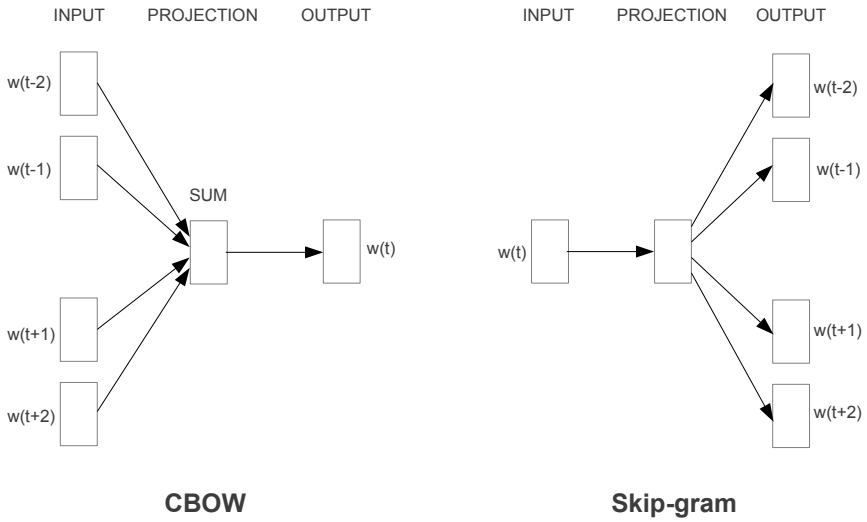

Рисунок 3 — Модели CBOW и Skip-gram (Источник рисунка: Mikolov [5])

Модель CBOW Рассмотрим упрощенную модель CBOW (см. рис. 4) с контекстом из одного слова более детально. Она представляет собой полносвязную нейронную сеть со скрытым слоем. Входной слой, принимающий one-hot вектор, состоит из V нейронов. Скрытый слой в свою очередь имеет N нейронов. На выходном слое применяется операция Softmax, давая на выходе распределение вероятностей по всем словам в словаре. Слои связаны матрицами весов $\mathbf{W} \in \mathcal{R}^{V \times N}$ и $\mathbf{W}^{'} \in \mathcal{R}^{N \times V}$ соответственно.

Каждая строка матрицы $\mathbf{W}$ это N-мерное векторное представление $\mathbf{v}_w$ связанного слова входного слоя. Формально можно записать строку i матрицы $\mathbf{W}$ как $\mathbf{v}_w^T$. Таким образом,

$$\mathbf{h} = \mathbf{W}^T\mathbf{x} = \mathbf{W}^T_{(k,\cdot)} := \mathbf{v}^T_{w_I}, \tag{3}$$

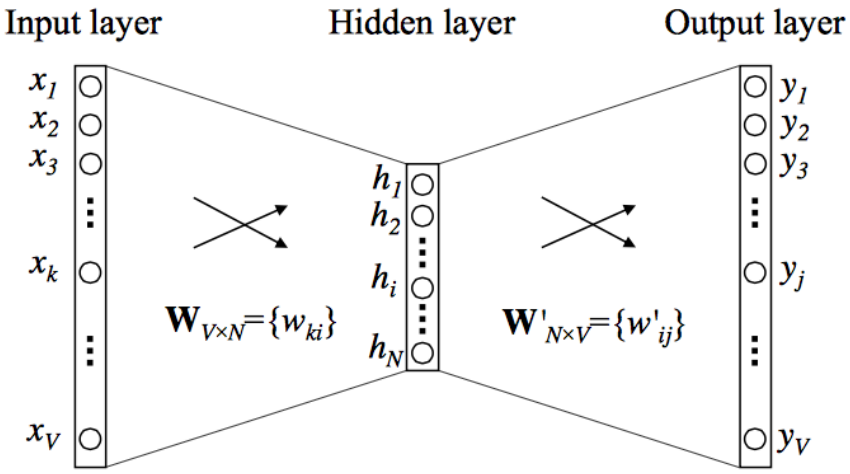

Рисунок 4 — Модель CBOW (Источник рисунка: Rong [6])

где $\mathbf{v}_{w_I}$ - векторное представление входного слова $\mathbf{w_I}$.

Скрытый и выходной слой связаны матрицей $\mathbf{W}' = \{w'_{ij}\}$. Используя данную матрицу весов можно посчитать оценку u_j для каждого слова из словаря

$$u_j = \mathbf{v}'^{T}_{w_j}\mathbf{h}, \tag{4}$$

где $\mathbf{v}'_{w_j}$ это j-я колонка матрицы W'. После этого используется нелинейная функция Softmax(5) для получения апостериорного распределения слов.

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V}\exp(u_{j'})}, \tag{5}$$

где y_j является выходом j-го нейрона на выходном слое. Используя формулы (3) и (4) можно записать (5) в следующем виде:

$$p(w_j|w_I) = \frac{\exp\left(\mathbf{v}'^{T}_{w_j}\mathbf{v}_{w_I}\right)}{\sum_{j'=1}^{V}\exp\left(\mathbf{v}'^{T}_{w_{j'}}\mathbf{v}_{w_I}\right)} \tag{6}$$

Учитывая, что $\mathbf{v}_w$ и $\mathbf{v}'_w$ - представления слова w через строки матрицы $\mathbf{W}$ и колонки матрицы $\mathbf{W}'$ соответственно. Будем называть $\mathbf{v}_w$ «входным вектором», а $\mathbf{v}'_w$ - «выходным вектором» слова w.

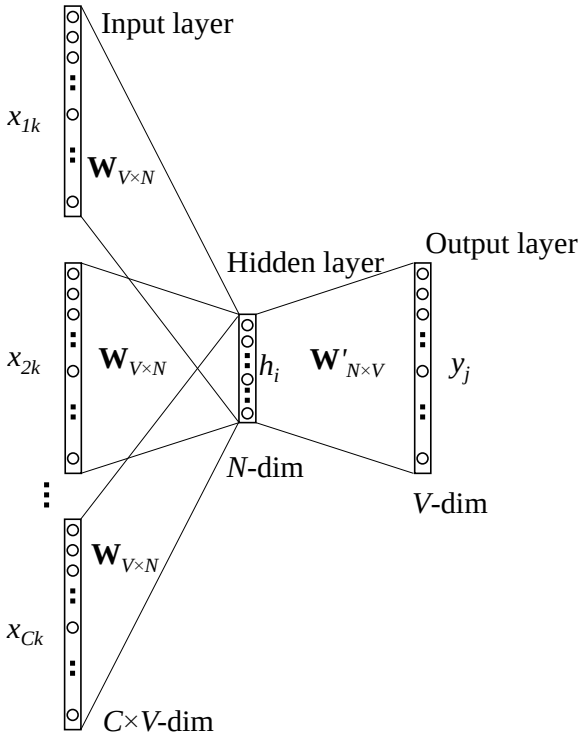

Рисунок 5 — Модель CBOW в общем случае

В качестве функции потерь, которую следует минимизировать, используется логарифмическая функция правдоподобия

$$E = -\log p(w_O|w_I) \tag{7}$$

где w_I - входное слово, а w_O- предсказанное.

В общем случае модели CBOW см. рис. 5, когда контекст превышает одно слово, при вычислении выходных значений скрытого подсчитывается среднее значение векторов контекста:

$$\mathbf{h} = \frac{1}{C}\mathbf{W}^T(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_C) \tag{8}$$

$$= \frac{1}{C}(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \cdots + \mathbf{v}_{w_C})^T \tag{9}$$

где C- количество слов в контексте, $w_1, \cdots, w_C$ - контекстные слова, $\mathbf{v}_w$ - входное слово w.

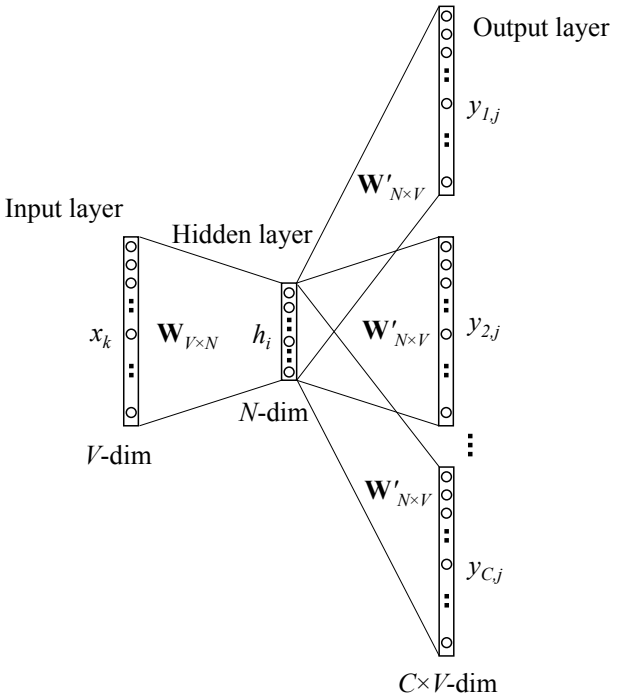

Рисунок 6 — Модель Skip-gram в общем случае

Модель Skip-gram Рассматриваемая общая модель (см. рис. 6) является обратной к CBOW. Теперь целевое слово подается на входной слой, а предсказание контекстных слов является результатом работы модели.

Будем использовать обозначение $\mathbf{v}_{w_I}$ для входного вектора. Через $\mathbf{h}$ обозначим выходные данные скрытого слоя, которые останутся прежними:

$$\mathbf{h} = \mathbf{W}^T_{(k,\cdot)} := \mathbf{v}^T_{w_I}, \tag{10}$$

Будем использовать обозначение $\mathbf{v}_{w_I}$ для входного вектора. Через $\mathbf{h}$ обозначим выходные данные скрытого слоя:

$$\mathbf{h} = \mathbf{W}^T_{(k,\cdot)} := \mathbf{v}^T_{w_I}, \tag{11}$$

Выходной же слой теперь формирует C полиномиальных распределений вместо одного. Каждый элемент выходного слоя считается с использование матриц скрытого и выходного слоя.

$$p(w_{c,j} = w_{O,c}|w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V} \exp(u_{j'})} \tag{12}$$

$w_{c,j}$ – j-е слово в c-м векторе выходного слоя;
$w_{O,c}$ – действительное c-е выходное контекстное слово;
w_I – единственное входное слово;
$y_{c,j}$ – предсказание j-го элемента в c-м векторе выходного слоя;
$u_{c,j}$ – входные данные j-го элемента в c-м векторе выходного слоя;

Функция потерь также меняется и принимает следующий вид:

$$E = -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C}|w_I) \tag{13}$$

$$= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j_c^*})}{\sum_{j'=1}^{V} \exp(u_{j'})} \tag{14}$$

$$= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp(u_{j'}) \tag{15}$$

где j_c^* – индекс верного c-го выходного слова в словаре.

Таким образом, преимущество использования модели Skip-gram перед CBOW в том, что при том же корпусе текстов получается больше экземпляров обучающей выборки, из-за чего повышается точность предсказаний модели.

Резюмируя, модель word2vec позволяет решить основную проблему векторного представления слов- размерность выходных векторов. Кроме того необходимо отметить главную особенность таких архитектур, которая заключается в том, что несмотря на отсутствие сохранения порядка слов в контексте, так как входные векторы суммируются, они сохраняют некоторые семантические характеристики корпусов текста, на которых проводится обучение. В связи с этим схо-

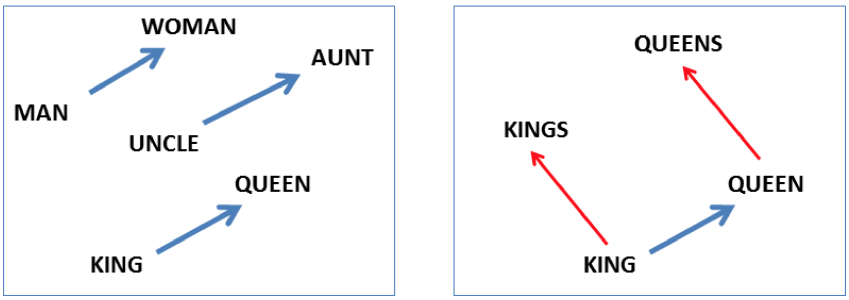

Рисунок 7 — Иллюстрация отношений векторного представления слов (Источник рисунка: Mikolov [7])

жие по смыслу слова имеют схожие векторные представления. Наиболее популярный пример сохранения семантической близости представлен на рис. 7.

1.3. CNN

Рассмотренный в предыдущем пункте метод векторного представления слов word2vec и схожие с ним (GloVe, etc.) дали большой толчок в разработке архитектур глубоких нейронных сетей для задач NLP. Данные представления применяются как первый слой обработки данных в deep learning моделях.

Одним из методов извлечения признаков из текстовых данных является применение сверточных нейронных сетей (англ. convolutional neural network, CNN). В задачах NLP наиболее часто встречаются варианты мультизадачных моделей, в которых результаты работы низкоуровневых слоев служат входными данными (признаками) для более высокоуровневых. Так, например, корректные POS тэги слов помогают улучшить показатель точности на задачах построения дерева зависимости.

Пример такой архитектуры был представлен авторами Коллобертом и Уэстоном [2]. В своей работе они использовать мультизадачную нейронную сеть, которая решает задачи определения частей ре-

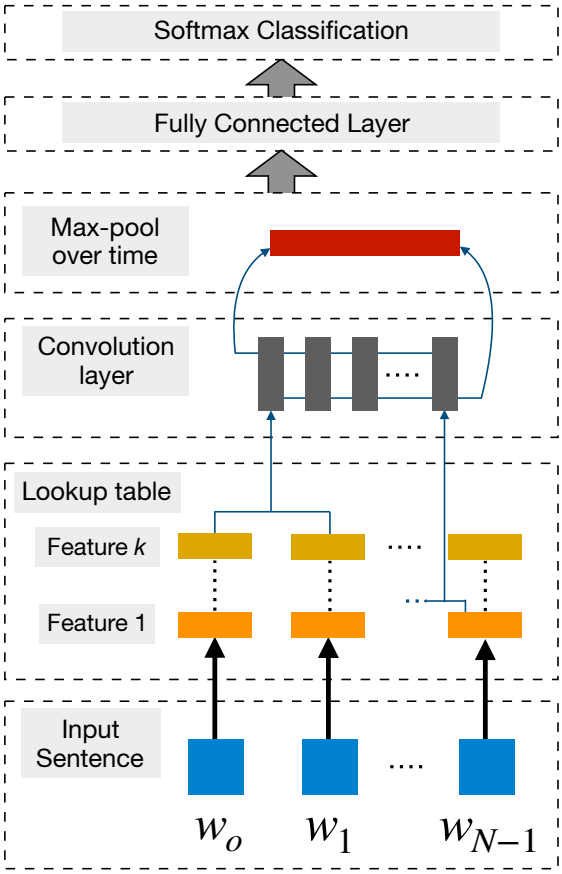

Рисунок 8 — Пример CNN для задачи классификации на уровне слов (Источник рисунка: Mikolov [2])

чи, восстановления семантических ролей, поиск именованных сущностей и др.

Как видно из Рис. 8, в этой архитектуре была использована таблица поиска (англ. lookup table) для векторного представления слов, благодаря которой последовательность слов $\{s_1, s_2, ...s_n\}$ была преобразована в некоторую последовательность векторов $\{w_{s_1}, w_{s_2}, ...w_{s_n}\}$

Сверточные нейронные сети способны извлекать информацию из n-грамм входного предложения, что дает возможность создания информативного семантического представления для последующих задач. Рассмотрим несколько архитектур и применений CNN для обработки естественного языка.

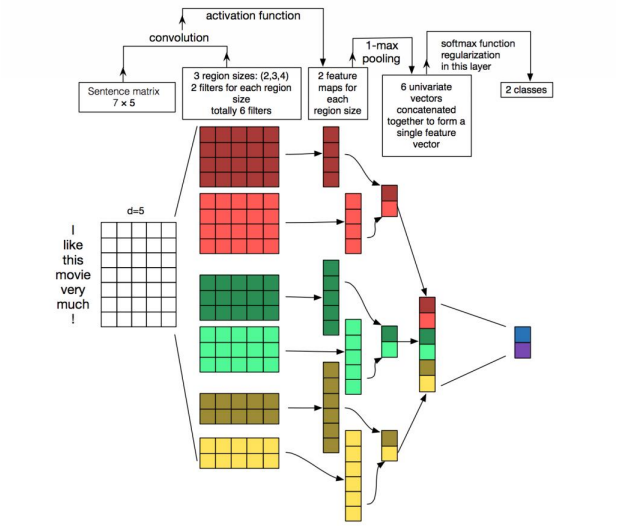

Рисунок 9 — CNN modeling on text (Figure source: [**zhang2015sensitivity**])

1.3.1. Базовая модель CNN

Положим $\mathbf{w}_i \in \mathcal{R}^d$ - векторное представление i-го слова входного предложения, где d - размерность векторов вложений. Тогда предложение, состоящее из n слов, может быть представлено матрицей вложений слов $\mathbf{W} \in \mathcal{R}^{n \times d}$. Пример такого представления можно увидеть на Рис. 9.

Обозначим $\mathbf{w}_{i:i+j}$ как конкатенацию векторов вложений слов $\mathbf{w}_i, \mathbf{w}_{i+1}, ...\mathbf{w}_{i+j}$. Начальная свертка применяется к этому входному слою и представляет собой фильтр $\mathbf{k} \in \mathcal{R}^{h \times d}$, который применяется к h словам, создавая новые признаки. Например, признак c_i получается путем применения окна свертки к словам $\mathbf{w}_{i:i+h-1}$

$$c_i = f(\mathbf{w}_{i:i+h-1}.\mathbf{k}^T + b) \tag{16}$$

$b \in \mathcal{R}$ - смещение, f - нелинейная функция активации (например, сигмоида). Фильтр k применяется ко всем возможным окнам текста и формирует карту признаков

$$c = [c_1, c_2, ..., c_{n-h+1}] \tag{17}$$

Стоит отметить, что представление текстовых данных отличается от представления изображений, к которым изначально применялись CNN. В отличие от них, текстовые данные представлены в двух, а не трех измерениях: ширине и количестве каналов. Ширину формирует количество слов в последовательности, а количество каналов - длина векторного представления слов. Учитывая то, что embedding слов сохраняет смысл только полностью, в задачах NLP применяются одномерные свертки (1D convolution), двигаясь только по оси, которая отвечает за количество слов.

В CNN применяется множество фильтров различной ширины, что позволяет извлекать специфичную информацию из различных n-грамм. Чаще всего после сверточного слоя идет слой с операцией *max-pooling*, которую обозначим $\hat{c} = max\{c\}$. Эта операция выбирает максимальное значение из некоторого окна, проходясь по карте признаков. У такой операции есть несколько предпосылок.

Первая причина - *max-pooling* обеспечивает контроль размерности выходных данных, что необходимо для задач классификации. Вторая причина заключается в том, что операция *max-pooling* позволяет более высокоуровневым слоям работать с информацией, которая захватывает больший участок данных, при этом такая операция передает наиболее значимые данные, полученные на всем предложении.

Список литературы

1. *Turing A. M.* Computing Machinery and Intelligence // Mind. — 1950. — Т. 59, № 236. — С. 433—460. — (New Series). — ISSN 00264423. — URL: http://www.jstor.org/stable/2251299.
2. Natural Language Processing (almost) from Scratch / R. Collobert [и др.] // CoRR. — 2011. — Т. abs/1103.0398. — arXiv: 1103.0398. — URL: http://arxiv.org/abs/1103.0398.
3. *Watanabe T.*, *Sumita E.* Bidirectional Decoding for Statistical Machine Translation // COLING 2002: The 19th International Conference on Computational Linguistics. — 2002. — URL: https://www.aclweb.org/anthology/C02-1050.
4. *Bellman R.*, *Corporation R.* Dynamic Programming. — Princeton University Press, 1957. — (Rand Corporation research study). — URL: https://books.google.com.ua/books?id=rZW4ugAACAAJ.
5. Efficient Estimation of Word Representations in Vector Space / T. Mikolov [и др.] // 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings / под ред. Y. Bengio, Y. LeCun. — 2013. — URL: http://arxiv.org/abs/1301.3781.
6. *Rong X.* word2vec Parameter Learning Explained // ArXiv. — 2014. — Т. abs/1411.2738.
7. *Mikolov T.*, *Yih W.-t.*, *Zweig G.* Linguistic Regularities in Continuous Space Word Representations // Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. — Atlanta, Georgia : Association for Computational Linguistics, 06.2013. — С. 746—751. — URL: https://www.aclweb.org/anthology/N13-1090.