МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА

На правах рукописи

Тихомиров Михаил Михайлович

# Методы автоматизированного пополнения графов знаний на основе векторных представлений

Специальность 05.13.11 —
«Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
доктор технических наук
Лукашевич Наталья Валентиновна

Москва — 2022

## Оглавление

Стр.

Стр.

Стр.

# Введение

Одним из основных направлений в области искусственного интеллекта является исследование моделей представления знаний (онтологий) [1; 2], которые предназначены для формализованного описания знаний о мире и предметной области. В приложениях автоматической обработки текстов особенно востребованы онтологии в виде семантических сетей [3]. В последнее время активно исследуются подходы к применению так называемых графов знаний (Wikidata, Freebase, ConceptNet) [4—6], в том числе в сочетании с подходами на основе машинного обучения [7]. Графы знаний представляют собой семантические сети большого объема, в состав которых входит как система классов и подклассов понятий (таксономия), так и описания конкретных (именованных) сущностей [8]. Отношения в графах знаний представлены в виде триплетов: субъект-отношение-объект.

Графы знаний используются в ряде задач обработки естественного языка, таких как информационный поиск [9], вопросно-ответные системы [10], чат боты [11], извлечение именованных сущностей [12] и др. Подходы, основанные на явных знаниях, являются более интерпретируемыми. Также, некоторые задачи требуют дополнительной точности и специализированных знаний в предметных областях.

Созданные онтологии и графы знаний необходимо уметь пополнять, поэтому часто обсуждается задача автоматического пополнения онтологий на основе больших текстовых коллекций, в которых содержится разнообразная информация и знания [13]. Кроме того, важной задачей является создание онтологий для конкретных предметных областей.

Методы автоматического извлечения знаний из текстовых коллекций включают несколько этапов, такие как извлечение новых понятий, терминов, именованных сущностей, определение синонимов и вариантов терминов, извлечение отношений новых сущностей [14]. Одной из важных задач в построении онтологий является построение таксономии классов, т.е. выявление отношений между более широкими (родовыми) классами и их более конкретными (видовыми) классами сущностей. В извлечении знаний из текстов данная задача ставится как извлечение гиперонимов - родовых слов для данного нового слова [15]. Часто тестирование подходов к извлечению гиперонимов производится на

основе лексико-семантических ресурсов типа WordNet, содержащего представления значений более 100 тысяч слов английского языка в виде семантической сети [16].

Самыми первыми подходами к извлечению таксономических отношений из текстов были подходы на основе шаблонов, например "X – это Y" [17], однако такие подходы обладают очень низкой полнотой, поскольку требуют присутствия соответствующих слов в одних и тех же предложениях в ограниченном количестве заданных конструкций.

Новые возможности для извлечения знаний из текстов появились на основе векторных представлений слов (эмбеддингов), которые формируются на основе контекстов упоминания слов [18—20]. Сходство контекстов слов приводит к сходству их векторных представлений, что дает возможность автоматического определения семантической близости слов на основе текстовых коллекций. Одних из первых успешных шагов в этом направлении была модель Word2Vec [21], разработанная в 2013 году. Дальнейшем развитием стали контекстуализированные векторные представления, которые формируют вектор для слов в зависимости от используемого контекста. Представителями таких подходов являются ELMO [22], BERT [23] и др. Однако векторные модели не могут предсказывать тип отношения между словами с достаточной точностью, требуют их дополнительной обработки для извлечения интерпретируемых отношений. Тестирование подходов для извлечения таксономических отношений (отношений класс-подкласс), извлечение гиперонимов по текстовым коллекциям в рамках разных конференций, подходов, показывает, что качество извлечения знаний является недостаточно высоким, поэтому задача пополнения онтологий, графов знаний по текстам является актуальной.

Описанные выше проблемы, представляют интерес для исследований из-за того, что необходимы методы переноса подходов и ресурсов на новые предметные области, что делает данное исследование **актуальным**.

**Степень разработанности темы.** Отношения гиперонимии-гипонимии составляют основу структуры множества онтологий и графов знаний. Поэтому многочисленные исследования посвящены извлечению подобных отношений из текстовых коллекций. Гиперонимы могут быть извлечены с нуля, без каких-либо целевых ресурсов или таксономии, но качество таких подходов обычно достаточно низкое и не позволяет строить качественные таксономии, которые можно было бы использовать в рамках других задач. Также задача извлече-

ния гиперонимов может ставиться как задача поиска гиперонимов для новых слов в существующей таксономии, то есть как задача обогащения или пополнения таксономии.

В 2016 г. задача по обогащению таксономии была организована как соревнование на семинаре SemEval (задача 14) [24]. Участники должны были связать слова с определениями для исправления гиперонимов в WordNet [16]. Однако в реальных приложениях определения новых слов и их значений, скорее всего, отсутствуют. В 2020 году было организовано новое соревнование RUSSE'2020 [25] по обогащению таксономии для русского RuWordNet, аналога WordNet для русского языка, содержащего представление значений слов для более 100 тысяч слов и выражений [26]. Задача состояла в том, чтобы найти правильные гиперонимы из опубликованной версии RuWordNet для слов, добавленных в новой версии RuWordNet. Дальнейшим развитием набора данных RUSSE'2020 стал набор данных диахронических ворднетов (Diachronic wordnets) [27], которые были созданы на основе английских и русских таксономий типа ворднет (WordNet). Эти наборы данных содержат новые слова, добавленные в более поздние версии ворднетов по сравнению с более ранними версиями, вместе с их гиперонимами в более старых версиях.

Разделение задачи пополнения графов знаний на а) пополнение таксономии абстрактными понятиями, и б) последующее пополнение именованными сущностями исследовалось в ряде работ. Например, авторы графа знаний AliCoCo [28] таким образом развивали свой граф знаний для электронной коммерции. В их подходе, в частности, использовались методы извлечения именованных сущностей, как для пополнения графа знаний непосредственно именованными сущностями, так и для пополнения абстрактными понятиями. Но предложенный подход, помимо того, что содержит большое количество ручных действий, не может быть прямо применен из-за отсутствия описания ряда шагов системы и закрытости решения.

Задачи представления и пополнения знаний исследовались в работах Т.А. Гавриловой, В.Ф. Хорошевского, И.М. Зацмана, И.Л. Артемьевой, Ю.А. Загорулько, О.А. Невзоровой, С.О. Кузнецова. Задачи извлечения знаний из текстов, а также использования векторных представлений для определения семантических отношений между словами исследовались в работах таких исследователей как Т. Mikolov, А.И. Панченко, Е.И. Большакова, Н.Э. Ефремова, Д.А. Усталов, П.И. Браславский.

**Целью** работы является исследование и разработка методов пополнения графов знаний новыми понятиями и именованными сущностями. Для достижения поставленной цели необходимо решить следующие **задачи**:

1. Исследовать существующие подходы к задаче пополнения графов знаний новыми понятиями и именованными сущностями,
2. Разработать методы пополнения таксономии графа знаний новыми понятиями и именованными сущностями,
3. Исследовать возможности адаптации графа знаний на конкретную предметную область, используя разработанные подходы,
4. Реализовать систему для автоматизированного пополнения графа знаний новыми понятиями и именованными сущностями.

**Научная новизна:**

1. Разработан и реализован метод пополнения таксономии графа знаний с использованием мета-векторных представлений. Исследована применимость разработанного метода на русском и английском языках, в общей области и конкретной предметной области информационной безопасности,
2. Разработан новый подход к порождению псевдоразметки для задачи извлечения именованных сущностей,
3. Разработан новый подход к задаче извлечения именованных сущностей в области информационной безопасности для русского языка с использованием псевдоразметки, двухэтапного обучения и специализированной языковой модели в области компьютерной безопасности RuCyBERT,
4. Реализована автоматизированная программная система для пополнения графа знаний новыми понятиями и именованными сущностями.

**Теоретическая и практическая значимость.** Теоретическая значимость работы состоит в том, что исследованы различные способы комбинирования векторных представлений слов и показано, что комбинация представлений с помощью автокодировщиков с учетом дополнительной информации о задаче приводит к улучшению качества векторных представлений, что в свою очередь приводит к улучшению качества решения целевой задачи.

Практическая значимость работы состоит в разработке и реализации подходов к пополнению таксономии графа знаний новыми понятиями и к извлечению именованных сущностей в предметной области информационной

безопасности. Разработанные методы позволяют пополнять графы знаний как абстрактными понятиями, так и именованными сущностями. Подход показал свою работоспособность не только на общей предметной области, но и на конкретной предметной области информационной безопасности. Разработанные методы могут использоваться в автоматизированных системах. Разработанные подходы по пополнению таксономии новыми понятиями показали наилучший результат на рассмотренных наборах данных, метод для адаптации модели BERT для задачи извлечения именованных сущностей в области информационной безопасности для русского языка показал наилучшее качество на описанном наборе данных.

**Методология и методы исследования.** Для решения поставленных задач использовались элементы теории вероятностей, методы машинного обучения, математической статистики, методы построения векторных моделей на основе дистрибутивной семантики и методы построения мета-векторных представлений. При разработке использовались методы объектно-ориентированного программирования, язык Python.

**Основные положения, выносимые на защиту:**

1. Комбинированный подход к задаче пополнения таксономии на основе шаблонов и векторных представлений слов,
2. Комбинированный подход к задаче пополнения таксономии на основе мета-векторных представлений,
3. Метод получения псевдоразметки для задачи извлечения именованных сущностей,
4. Подход к извлечению именованных сущностей с использованием псевдоразметки, двухэтапного обучения модели RuCyBERT,
5. Автоматизированная программная система для пополнения таксономии графа знаний новыми понятиями.

**Достоверность** полученных результатов обеспечивается проведенными экспериментами, открытым кодом реализованных методов и подходов, обоснованием принимаемых решений, публикациями в рецензируемых журналах и апробацией на российских и международных конференциях.

**Апробация работы.** Основные результаты работы докладывались на:

1. Text, Speech, and Dialogue 22nd International Conference, TSD 2019, Ljubljana, Slovenia, September 11–13, 2019,

2. Ломоносовские чтения 2020 - Секция вычислительной математики и кибернетики, Москва, Россия, 2020,
3. International Conference on Computational linguistics and intellectual technologies Dialog-2020, Москва, Россия, 17-20 июня 2020,
4. International Conference on Applications of Natural Language to Information Systems (NLDB-2020), Saarbruken, Germany, June 24-26, 2020,
5. International Conference on Computational Linguistics and Intellectual Technologies Dialogue 2021, Москва, Россия, 16-19 июня 2021,
6. XXIII "Data Analytics and Management in Data Intensive Domains"conference (DAMDID), Moscow, Россия, 26-29 октября 2021,
7. XII Международная научная конференция «Интеллектуальные системы и компьютерные науки», Москва, МГУ имени М.В. Ломоносова, Россия, 29 ноября - 3 декабря 2021.

**Личный вклад.** Все представленные в диссертации результаты получены лично автором. Подготовка части материалов к публикации проводилась совместно с соавторами, причем вклад диссертанта был определяющим. В работах [29—31] Н.В. Лукашевич принадлежит постановка задачи пополнения таксономии графа знаний, а также предоставление наборов данных. В работах [32—34] Б.В. Доброву принадлежат рекомендации к методологии исследований и постановка задачи, Н.В. Лукашевич предоставила набор данных для задачи извлечения именованных сущностей, списки дескрипторов, а также сформулировала идею о пополнении тренировочных данных за счет методов псевдоразметки. В работе [35] автор проводил вычислительный эксперимент, а идея, постановка и анализ результатов принадлежат Н.В. Лукашевич. В работе [36] автору принадлежат все эксперименты с использованием предложенного автором алгоритма с использованием мета-векторных представлений, а И.А. Никишиной обучение графовых векторных представлений, визуализация результатов, формирование набора данных, другие соавторы участвовали в постановке задачи, анализе результатов.

**Публикации.** Основные положения и выводы диссертационного исследования в полной мере изложены в 9 научных работах [29—37], в том числе в 8 публикациях в рецензируемых научных изданиях [29—36], определенных п. 2.3 «Положения о присуждении ученых степеней в Московском государственном университете имени М.В.Ломоносова».

**Объем и структура работы.** Диссертация состоит из введения, 4 глав и заключения. Полный объём диссертации составляет 119 страниц, включая 19 рисунков и 32 таблицы. Список литературы содержит 127 наименований.

# Глава 1. Векторные представления в задачах автоматической обработки текстов

Для решения задач автоматической обработки языка текст должен представляться в виде, понятным для компьютера. Для этого было разработано множество различных методов, которые могут представлять слова, предложения и документы таким образом, чтобы эффективно решать целевые задачи. В подавляющем большинстве случаев подобные представления являются векторными. Условно можно выделить несколько классов векторных представления: статические векторные представления слов, контекстуализированные векторные представления слов, векторные представления графов.

## 1.1 Векторные представления слов

Одним из первых способов представления слов является кодирование каждого слова в виде one-hot вектора. One-hot вектор — это вектор размерности словаря, у которого все значения равны нулю, кроме позиции, которая относится к кодируемому слову (на этой позиции устанавливается значение 1). Подобное представление слов не может быть эффективно использовано напрямую, так как они не несут в себе какой-либо семантики, но может использоваться для первичной векторизации с дальнейшим построением других векторных представлений.

В основе более современных векторных представлений лежит дистрибутивная гипотеза, которая постулирует, что слова имеют похожий смысл, если они используются в похожих контекстах. Лежащая в основе идея была популяризирована Фёрсом в 1957 г. [38] "Слово характеризуется контекстом, в котором оно содержится".

| | Компьютер | Данные | Сорвать | Результат | Сахар |
|---|---|---|---|---|---|
| Абрикос | 0 | 0 | 1 | 0 | 1 |
| Ананас | 0 | 0 | 1 | 0 | 1 |
| Цифровой | 2 | 1 | 0 | 1 | 0 |
| Информация | 1 | 6 | 0 | 4 | 0 |

Рисунок 1.1 — Пример матрицы совместной встречаемости

### 1.1.1 Матрица совместной встречаемости слов

Одним из традиционных подходов к построению векторных представлений, используя дистрибутивную гипотезу, является построение матрицы совместной встречаемости слов в документах с последующем построением векторных представлений на основе этой матрицы [18—20]. Обычно, при формировании матрицы совместной встречаемости, говорят, что слова находятся в одном контексте, если они находятся в пределах $w$ слов (параметр окна). Подобная матрица называется матрицей слов к контекстам, или же матрицей совместной встречаемости. После построения подобной матрицы, ее значения обычно взвешиваются, так как простые частотные значения не являются хорошей мерой ассоциации слов. Часто используется: логарифмическое взвешивание, *поточечная взаимная информация* (PMI) или *положительная поточечная взаимная информация* (PPMI). В экспериментах 2000-2014 годов было выявлено, что PPMI является лучшей мерой взвешивания контекстов.

$$PMI(x,y) = log\frac{P(x,y)}{P(x)*P(y)};$$
$$PPMI(x,y) = max(0, PMI(x,y)). \tag{1.1}$$

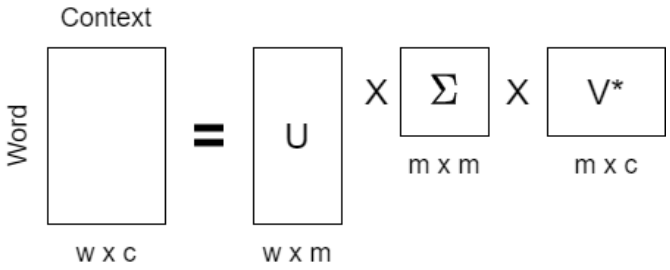


Рисунок 1.2 — SVD разложение матрицы слов к контекстам

В случае матрицы совместной встречаемости, соответствующие вероятности рассчитываются, исходя из частотных характеристик слов и контекстов:

$$P(x,y) = \frac{f_{xy}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}};$$

$$P(x) = \frac{\sum_{j=1}^{C} f_{xj}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}; \tag{1.2}$$

$$P(y) = \frac{\sum_{i=1}^{C} f_{iy}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}.$$

Получившаяся матрица уже содержит некоторые знания о словах, но имеет существенный недостаток - размерность матрицы равна размерности словаря, помимо этого, матрица очень разрежена. Для борьбы с этими недостатками применяется метод сжатия размерности на основе сингулярного разложения (SVD).

В подобном разложении (см. Рис. 1.2) $m$ — это ранг матрицы, а $\Sigma$ - матрица, состоящая из сингулярных значений исходной матрицы. Сокращая $m$ до некоторого $k \ll m$ и оставляя только $k$ наибольших сингулярных значений и соответствующих им сингулярных векторов, можно получить аппроксимацию исходной матрицы. В случае с матрицей совместной встречаемости слов, обычно вместо исходной матрицы используют матрицу $U$.

### 1.1.2 Векторная модель Word2Vec

В 2013 году в обработке естественного языка произошел прорыв за счет работы [39]. Авторы предложили новый способ эффективного расчета векторных представлений слов по корпусу текстов, не прибегая к построению

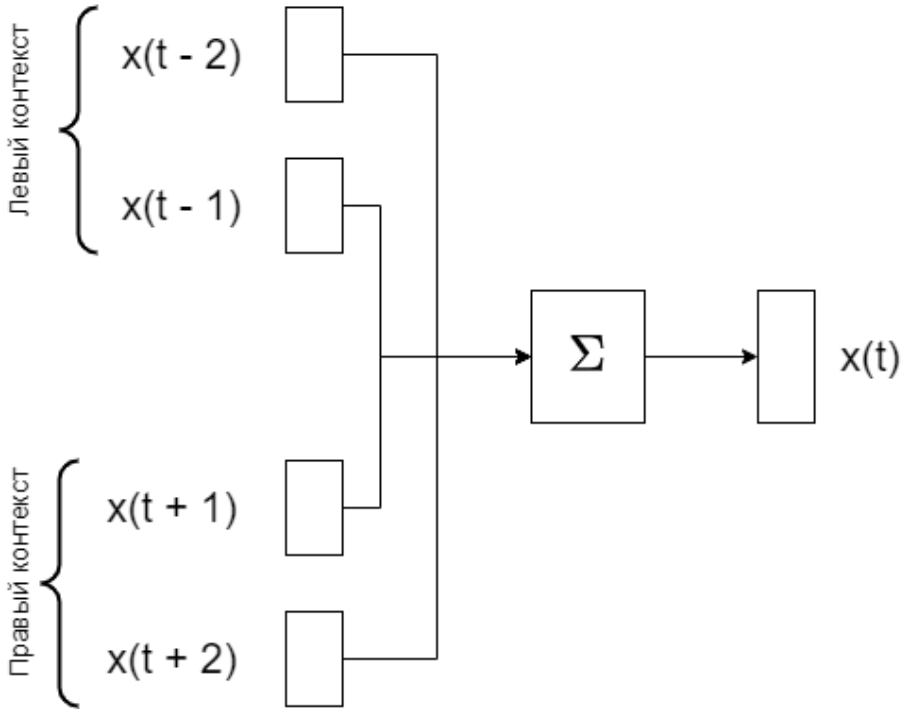


Рисунок 1.3 — Архитектура нейронной сети CBOW

огромной матрицы совместной встречаемости. Подход, названный Word2Vec, представляет собой использование нейронной сети для обучения дистрибутивных векторных представлений на основе текстового корпуса. Авторами были предложены две архитектуры нейронной сети для решения данной задачи: CBOW (Continuous Bag-of-Words Model) и Skip-Gram.

### Алгоритм CBOW

В алгоритме CBOW модель обучается предсказывать слово по его контекстам (см. Рис. 1.3). Контекст представляет собой окно вокруг целевого слова, то есть $k$ слов слева и $k$ слов справа. Для этого входные слова представляются в виде one-hot векторов, которые суммируются, после чего подобный вектор идет на вход полносвязного слоя $W_1^{||V||\times d}$, за которым следует второй полносвязный слой $W_2^{d\times||V||}$ с softmax функцией активации на выходе.

Полносвязный слой нейронной сети преобразует входной вектор $V_{in}$ размерности $N$ в другой вектор размерности $d$ следующим образом:

$$V_{out} = \sigma(V_{in} * W^{N\times d} + B), \tag{1.3}$$

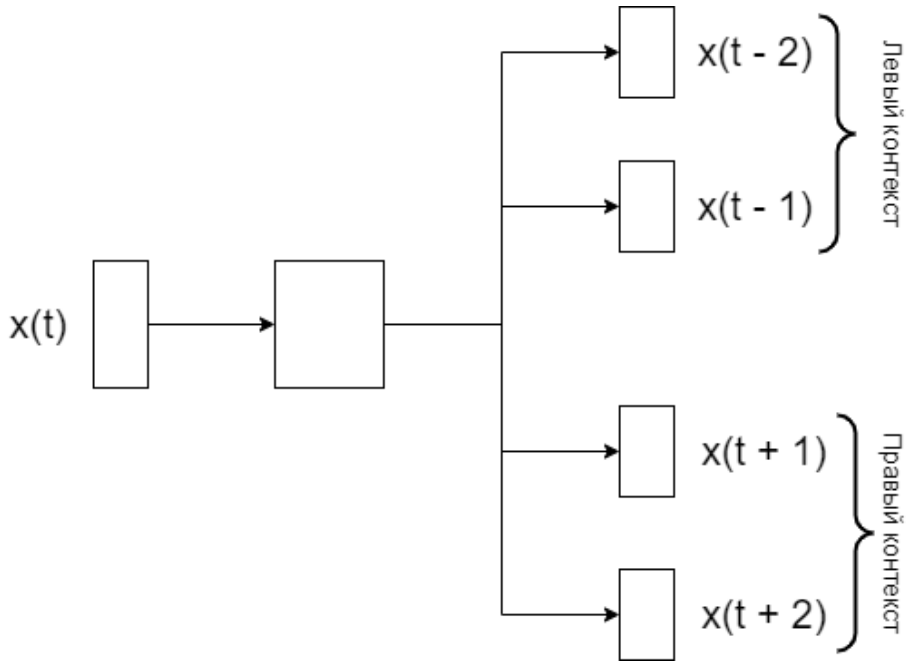


Рисунок 1.4 — Архитектура нейронной сети Skip-Gram

где σ называется функцией активации, которая добавляет нелинейность в модель, $W^{N\times d}$ - обучаемая матрица весов слоя, $B$ - обучаемый вектор весов (биас). Функция активации может быть представлена любой дифференцируемой функцией, а в случае с последним слоем для классификации, обычно используется softmax.

$$softmax(w_t) = \frac{e^{w_t}}{\sum_{w'\in V} e^{w'}}. \tag{1.4}$$

### Алгоритм Skip-Gram

Алгоритм Skip-Gram устроен схожим с CBOW образом, но целевой задачей обучения является предсказание слов контекста по целевому слову (см. Рис. 1.4).

После обучения, в качестве векторных представлений слов, часто используется матрица на скрытом слое, в которой для каждого слоя из словаря имеется вектор размерности $d$.

### Негативное семплирование

Важным результатом работы [39] для обучения Word2Vec было использование подхода, названного негативным семплированием. При негативном семплировании функция потерь рассчитывается не для всех возможных предсказаний (то есть не для всего словаря), а только для 1) позитивных примеров (то есть правильных ответов), и 2) для некоторого выбранного количества негативных примеров, которые не являются правильными ответами. Суть негативного семлирования в том, чтобы на каждом примере из обучающей выборке обновлять веса не всей модели, а только для части весов, соответствующим набору позитивных и негативных слов.

Выбор слов для негативного семплирования реализован таким образом, чтобы наиболее частые слова в корпусе имели больший шанс быть выбранными в соответствии с формулой:

$$P(w) = \frac{f(w)^{3/4}}{\sum_{j=0}^{n} f(w_j)^{3/4}}. \tag{1.5}$$

Итоговая функция потерь преобразуется в следующий функционал:

$$Loss = log\, \sigma(u_i^T v_c) + \sum_{j \sim P(w)} [log\, \sigma(-u_j^T v_c)], \tag{1.6}$$

где первый член формулы отвечает за правильные примеры и максимизирует их вероятность, а второй член формулы отвечает за негативные примеры, минимизируя их вероятность.

### Эквивалентность Word2Vec и PPMI SVD

Авторы работы [18] показали, что Word2Vec Skip-Gram — это на самом деле неявная факторизация матрицы контекста слов, строки которой представляют собой поточечную взаимную информацию (PMI) соответствующей паре слова и контекста, сдвинутые на некоторую константу. Они также показали, что использование разреженной матрицы контекста слова со сдвинутым PPMI для представления слов улучшает результаты на двух задачах на близость слов и на

одной из двух задач на аналогию слов. Точная факторизация с помощью SVD может обеспечить решения, которые по крайней мере не хуже, чем word2vec, а иногда и лучше, но стоит учитывать вычислительные возможности, так как Word2Vec позволяет обучаться на намного больших наборах данных.

### 1.1.3 Векторная модель FastText

FastText [40] это другой способ построения векторных представлений слов, который можно считать расширением алгоритма Word2Vec. Ключевым отличием данного подхода является то, что вместо выучивания векторных представлений слов, модель на самом деле обучается представлять n-граммы символов. Например, для слова **вектор** и n-грамм, где $n = 3$, FastText представляет слово как набор триграмм <ве, век, ект, кто, тор, ор>. В случае FastText, также, как и в Word2Vec, можно обучать модель, как с использованием CBOW, так и с Skip-Gram.

Подобный подход позволяет как получать векторные представления слов, которых не было в наборе данных для обучения, так и позволяет лучше выучивать смысл коротких и редких слов. Помимо этого, при обучении FastText можно работать с текстом без предварительной лемматизации, позволяя модели выучить морфологию самостоятельно, при этом не опасаясь за то, что итоговый размер словаря будет слишком большим.

### 1.1.4 Векторная модель GloVe

GloVe [41] — это алгоритм обучения векторных представлений на основе частотных характеристик слов и их совместной встречаемости. GloVe использует отношения вероятностей из матрицы совместной встречаемости слов, объединяя идеи моделей на основе частотных характеристик, таких как PPMI SVD, с идеями таких методов, как Word2Vec.

Word2Vec строится исключительно на основе локальной информации (контекст слов в некотором окне). При всей своей вычислительной эффективности

такого подхода, теряется важная информация о глобальной информации слов, что и было исправлено в модели GloVe (откуда и следует название Global Vectors). GloVe основан на идее "можно вывести семантические отношения между словами на основе матрицы совместной встречаемости ".

Целевой функцией обучения GloVe является построение векторов слов таким образом, чтобы их скалярное произведение равнялось логарифму вероятности совместной встречаемости слов. Так как логарифм отношения равен разности логарифмов, эта постановка связывает (логарифм) отношения вероятностей совместной встречаемости с разностями векторов в векторном пространстве слов. Поскольку эти отношения могут кодировать некоторую форму смысла, эта информация также кодируется как векторные разности.

Для этого авторы ввели следующую функцию потерь:

$$J = \sum_{i,j} f(X_{ij})(w_i^T \tilde{w}_j - log\, X_{ij})^2;$$

$$f(x) = \begin{cases} (x/x_{max})^{\alpha} & \text{если } x < x_{max} \\ 1 & \text{иначе} \end{cases}, \tag{1.7}$$

где $W$ и $\tilde{W}$ - две обучаемые матрицы векторов, $X_{ij}$ - количество раз, когда слово $j$ встретилось в контексте слова $i$, $f$ - некоторая функция взвешивания, $x_{max}$ в работе был выбран равным 100.

## 1.2 Векторные представления графов

Множество данных имеет графовую природу: социальные графы, графы переходов по ссылкам, графы знаний, таксономии и др. В связи с этим возникает вопрос, как учесть информацию о связях между вершинами при построении векторных представлений вершин. Эту задачу решает класс методов для построения графовых векторных представлений.

Рассмотрим граф $G = (V, E)$, где $V$ - вершины графа, $E$ - ребра графа, $E \subseteq (V \times V)$. Пусть $X^{||V||\times f}$ - матрица атрибутов\признаков для вершин из $V$, $Y^{||V||\times l}$ - матрица классов для этих вершин, $R^{||E||\times r}$ - матрица классов для ребер. Для подобной постановки, в зависимости от типа графа и условий, могут

быть поставлены различные задачи, которые могут решаться с использованием векторных представлений вершин\ребер\графов:

- В случае, если не все вершины имеют разметку по классам, можно ставить задачу предсказания классов для подобных вершин,
- Предсказание близости вершин,
- Предсказание не размеченных классов ребер (когда ребро существует, но его класс не известен),
- Предсказание наличия ребра между двумя вершинами (иногда вместе с предсказанием класса ребра),
- И другие.

### 1.2.1 Графовая векторная модель DeepWalk

В зависимости от задачи, векторные представления для подобных графов могут строится разными способами. Например, в алгоритме DeepWalk [42] авторы ставили задачу построения векторных представлений для вершин $X^{||V||\times d}$, где $d$ представляет собой небольшое число скрытых размерностей, таким образом, чтобы решать задачу многоклассовой классификации графа (multi-label network classification). Для этого, они предложили представлять граф в виде текста и обучать векторное представление похожим на Word2Vec [21] образом. Более формально, для каждой вершины $v$ из $V$ с использованием случайного блуждания производятся образцы цепочек переходов длины $t$, на таких цепочках происходит обучение векторных представлений, используя алгоритм SkipGram [21]. Подобная процедура производится $k$ раз.

### 1.2.2 Графовая векторная модельNode2Vec

Дальнейшим развитием идеи DeepWalk стал алгоритм Node2Vec [43]. Авторы предложили решать задачу максимизации правдоподобия для соседей вершины. Для этого они вводят понятие соседей, как $N_S(u) \subseteq V$ и ставят задачу построения отображения вершин в такое векторное пространство, в кото-

ром вершины-соседи будут также наиболее вероятными соседями для целевых вершин.

$$max_f = \sum_{u \in V} log\, Pr(N_S(u)|f(u)). \tag{1.8}$$

Также авторы алгоритма рассмотрели проблему формирования цепочек из графа. Есть два граничных случая: построение цепочек в ширину (BFS), построение цепочек в глубину (DFS). BFS работает таким образом, что список соседей ограничен вершинами, которые непосредственно имеют связь с изначальной вершиной. DFS же строит цепочки таким образом, чтобы каждый шаг увеличивать расстояние от изначальной вершины. Эти два способа построения цепочек представляют собой экстремальные случаи с точки зрения пространства поиска, которое они рассматривают. Авторами был предложен подход, который позволяет с помощью параметров управлять характером построения цепочек, таким образом обобщая оба подхода. Формально, имея исходную вершину $u$, фиксированную длину цепочки $l$, $i-$ шаг блуждания $c_i$ генерируется, используя следующее распределение:

$$P(c_i = x|c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{если } (v,x) \in E \\ 0 & \text{иначе} \end{cases}, \tag{1.9}$$

где $\pi_{vx}$ не нормализованная вероятность перехода между вершинами $v$ и $x$, а $Z$ константа нормализации.

Авторы определяют свой алгоритм случайного блуждания, как случайное блуждание второго порядка, то есть берущее в рассмотрение не только нынешний шаг, но и прошлый. Для этого вероятность перехода устанавливается как $\pi_{vx} = \alpha_{pq}(t,x) * w_{vx}$, где $w_{vx}$ — это вес ребра, а $\alpha_{pq}(t,x)$:

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{если } d_{tx} = 0 \\ 1 & \text{если } d_{tx} = 1 \\ \frac{1}{q} & \text{если } d_{tx} = 2 \end{cases} \tag{1.10}$$

и $d_{tx}$ определят длину кратчайшего пути между прошлой вершиной $t$ и следующей вершиной $x$.

Параметры $p$ и $q$ контролируют, как быстро случайное блуждание исследует и покидает окрестность исходной вершины $u$.

### 1.2.3 Графовая векторная модель TADW

В работе [44] доказывается, что DeepWalk является на самом деле эквивалентом факторизации матрицы $M \in \mathbb{R}^{||V|| \times ||V||}$, в которой на позиции $M_{ij}$ расположены логарифмы средней вероятности попадания из вершины $v_i$ в вершину $v_j$ за некоторое фиксированное число шагов. В случае с DeepWalk факторизация происходит в произведение двух матриц $W \in \mathbb{R}^{k \times ||V||}$ и $H \in \mathbb{R}^{k \times ||V||}$, где $k \ll ||V||$. Авторы же предложили факторизовать в три матрицы, $W \in \mathbb{R}^{k \times ||V||}$, $H \in \mathbb{R}^{k \times f_t}$ и $T \in \mathbb{R}^{f_t \times ||V||}$ , где $T$ - матрица текстовых признаков. Таким образом, в алгоритме TADW графовая информация дополняется текстовой, что может быть полезно, когда вершиной графа является текстовая сущность по своей природе. Предложенный подход показал повышение качества на задаче мультиклассовой классификации вершин, когда вершинами являлись некоторые текстовые документы.

### 1.2.4 Графовая векторная модель TransE

В алгоритме, TransE [45], ставилась задача представления вершин и отношений таким образом, чтобы для троек $(h, l, t)$ (где $h$ и $t$ некоторые вершины, а $l$ отношение между ними) выучивалось правило $h + l \approx t$. Для этого оптимизировалась разница в расстояниях, между позитивными (реальными) парами $h + l, t$ и между негативными, которые составлялись путем замены $h$ или $t$ на случайную вершину - критерий ранжирования с запасом расстояния (margin-based ranking criterion).

$$loss = \sum_{(h,l,t) \in S} [\gamma + d(h + l, t) - d(h' + l, t')], \tag{1.11}$$

где $S$ - множество всех троек $(h, l, t)$.

Важным отличием TransE подхода от описанных DeepWalk и node2vec является возможность работы с графами, в которых присутствуют разные виды отношений, что может быть важно для задач предсказания типов отношений

или же предсказания вершины, которая связана с другой вершиной некоторым отношением.

### 1.2.5 Графовая векторная модель Poincare Embeddings

В работе [46] был предложен способ обучения иерархических представлений для символьных данных, отображая их в гиперболическое пространство, в н-мерную сферу Пуанкаре. По словам авторов, гиперболические векторные представления более эффективны на иерархических данных и могут превосходить варианты, основанные на евклидовой геометрии. Это, например, показано на задаче пополнения таксономии [47].

Целью перехода от евклидового пространства на н-мерную сферу Пуанкаре в первую очередь является изменение функции расстояния между вершинами, таким образом, чтобы расстояния росли экспоненциально при удалении от центра графа. Таким образом, если вершины расположены далеко по графу друг от друга, они не окажутся близкими с точки зрения функции расстояния.

Более формально, определяется пространство $B^d = \{x \in R^d \mid ||x|| < 1\}$. Тогда расстояние между $v, u \in B^d$:

$$d(u,v) = \text{arcosh}(1 + 2\frac{||u-v||^2}{(1-||u||^2)(1-||v||^2)}). \tag{1.12}$$

На примере задачи векторизации таксономии, используя описанную функцию расстояния, при обучении минимизируется следующая функция потерь:

$$L(\Theta) = \sum_{(u,v)\in D} log\frac{e^{-d(u,v)}}{\sum_{u'\in N(u)} e - d(v,u')}, \tag{1.13}$$

где $D = \{(u,v)\}$ - множество существующих отношений гиперонимии между парами сущностей, $N(u) = \{v \mid (u,v) \notin D\} \cup \{u\}$.

### 1.2.6 Графовая сверточная сеть GCN

Авторы подхода GCN (graph convolutional network) [48] рассматривали задачу классификации вершин графа с частичным обучением, то есть в ситуации, когда для части графа присутствует информация о типах вершин, а для части нет.

Используя матрицу смежности графа и начальную матрицу признаков, алгоритм формирует векторное представление вершин на основе информации о соседях. GCN по своей идее схожа с CNN (сверточными нейронными сетями), но вместо маски свертки и локальной информации о соседних значениях в матрице, использует матрицу смежности и информацию о связанных вершинах.

Более формально, имея граф $G = (V, E)$, GCN принимает на вход матрицу признаков $X$ размерности $N \times f$ и матрицу смежности $A$ размерности $N \times N$, где $f$ - размерность признакового пространства, а $N = ||V||$. Скрытый слой GCN сети представляется как $H^l = f(H^{l-1}, A)$, где $H^0 = X$. $f$ - является некоторой функцией распространения, которая может выглядеть различными способами. Авторами была предложена следующая функция распространения.

$$H^{l+1} = \sigma(\tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}H^l W^l), \tag{1.14}$$

где $\tilde{A}$ - матрица совместной встречаемости с дополнительными единицами на диагонали, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ и $W_l$ - матрица обучаемых весов для слоя $l$, а $\sigma$ - функция активации.

Использование подобной функции распространения, которая оперирует как исходными признаками, так и матрицей смежности, позволяет эффективно распространять локальную графовую информацию, учитывая структуру графа, для получения итоговых векторных представлений вершин. Авторы также отмечают минусы данного подхода: алгоритм умеет работать только с неориентированными графами и нет поддержки разных типов ребер.

## 1.3 Мета-векторные представления

В связи с тем, что векторные представления слов могут быть получены, используя различные алгоритмы и наборы данных, возникает вопрос о методах комбинации различных векторных моделей в одну. Наиболее простые подходы к объединению векторов — это конкатенация или усреднение некоторых исходных векторных представлений.

$$E_{concat}^{f(V_1,V_2)||\times(N_1+N_2)} = E_1^{||V_1||\times N_1} \oplus E_2^{||V_2||\times N_2};$$
$$E_{avg}^{f(V_1,V_2)\times N} = \frac{E_1^{||V_1||\times N} + E_2^{||V_2||\times N}}{2}, \tag{1.15}$$

где $V_i$ - это словарь $i$-ой модели, $N_i$ - размерность векторов $i$-ой модели, $\oplus$ - знак конкатенации, $+$ - знак сложения, f - функция формирования итогового словаря, может быть как объединением, так и пересечением.

Авторы [49] показали, что даже такие простые комбинированные векторные представления могут значительно улучшить общее качество работы алгоритмов для задач аналогии слов [39] и семантической близости [50—54]. Также в работе [55] было показано, что использование сингулярного разложения (SVD) может показать хорошие результаты с возможностью управления конечной размерностью векторов. В данном случае сингулярное разложение применяется для матрицы векторных представлений, вектора в которой сформированы путем конкатенации изначальных векторных моделей. После разложения происходит снижение размерности путем выбора $k$ первых сингулярных значений.

$$E_{concat}^{N\times M} = U^{N\times N} \times \Sigma^{N\times M} \times V^{M\times M};$$
$$E_{svd}^{N\times k} = U_k^{N\times k}, \tag{1.16}$$

где $U_k^{N\times k}$ — это матрица, состоящая из первых $k$ левых сингулярных векторов матрицы $U^{N\times N}$ .

Автокодировщики (Autoencoders) [56], получившие название AutoEncoded Meta-Embeddings (AEME), стали дальнейшим развитием идеи создания мета-векторных представлений слов. Авторы [56] предложили несколько различных алгоритмов (CAEME, AAEME и др.) для объединения различных векторов в

один, путем кодирования исходных векторов в некоторое пространства мета-векторов и последующего декодирования в обратном направлении.

На первом этапе подхода CAEME все векторы слов кодируются в мета-векторы, а затем объединяются путем конкатенации. Каждый вектор кодируется в вектор того же размера, что и изначальный, а общий размер мета-вектора представляет собой сумму размеров исходных векторов. Затем, на этапе декодирования используется конкатенированное представление для декодирования исходных векторных представлений. Для каждой векторной модели при кодировании и декодировании используется свой кодировщик и декодировщик, которые обучаются совместно. И кодировщик и декодировщик представляют собой полносвязный слой с ReLU активацией на выходе. Помимо этого, после кодирования мета-вектора нормализуются.

Подход AAEME схож с CAEME, за исключением того, что каждый вектор кодируется в вектор фиксированного размера, и все закодированные представления усредняются, а не конкатенируются. Очевидным преимуществом этого подхода является возможность управлять размерностью мета-векторов.

Для любого AEME подхода на этапе декодирования могут использоваться различные функции потерь: MSE, KL-дивергенции, косинусное расстояние, а также их комбинации. В [57] авторы исследовали качество работы автокодировщиков в зависимости от функции потерь. Они обнаружили, что среди функций потерь нет очевидного победителя, и что для разных приложений следует подбирать функции потерь.

## 1.4 Контекстуализированные векторные представления

Описанные в прошлых разделах методы формируют векторные представления слов независимо от контекста. Такие векторные представления называют статическими. Особенностью естественных языков является многозначность многих конструкций и слов. В разных контекстах одна и та же текстовая конструкция может означать различные понятия. В связи с этим, в последние годы получило развитие направление по формированию зависящих от контекста векторных представлений.

### 1.4.1 Контекстуализированная векторная модель ELMO

Одним из первых алгоритмов по построению контекстуализированных векторных представлений был ELMO [22]. Авторы представили новый тип контекстуализированного представления слов, который моделирует как сложные характеристики употребления слов (например, синтаксис и семантику), так и то, как это использование различается в зависимости от языкового контекста (например, для моделирования многозначности). Подобные вектора слов представляют собой обученные скрытые состояния глубокой двунаправленной языковой модели, которая предварительно обучается на большом текстовом корпусе. Было показано, что эти представления можно добавить к существующим моделям и значительно улучшить качество на ряде задач, включая вопросно-ответные системы, анализ следования и анализ тональности.

В основе реализации двунаправленной языковой модели лежит рекуррентная сеть LSTM (Long short-term memory) [58]. Рекуррентные сети (RNN) [59] способны обрабатывать последовательность слов, учитывая последовательность прошлых слов. Каждая ячейка RNN принимает на вход текущее слово и скрытое состояние прошлой ячейки, формируя на выходе представление текущего токена и текущее скрытое состояние. Важным моментом также является то, что веса всех матриц ячеек общие. Модель LSTM является продвинутой версией обычных рекуррентных сетей, которая частично решает вопрос затухания градиентов при обучении модели. Слой LSTM выглядит следующим образом:

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f); \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i); \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o); \\ \tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c); \\ c_t &= f_t \cdot c_{t-i} + i_t \cdot \tilde{c}_t; \\ h_t &= o_t \cdot \sigma_h(c_t), \end{aligned} \tag{1.17}$$

где $f_t$ называют фильтром забывания (forget gate), $i_t$ фильтр входа (input gate), $o_t$ фильтр выхода, $c_t$ скрытое состояние, $h_t$ - выход модели, $x_t$ - вход модели, $\sigma$ - различные функции активации и $W, U, b$ - соответственно обучаемые матрицы модели (для каждого фильтра свой набор).

В подходе ELMO используется двунаправленный LSTM слой, который представляет собой конкатенацию двух LSTM слоев, один из которых обрабатывает текст слева-направо, а другой, наоборот. Таким образом, для каждого слоя формируется не статический вектор, а вектор, зависящий как от левого, так и правого контекста.

### 1.4.2 Контекстуализированная векторная модель ULMFiT

Примерно в тоже время, что и ELMO возник другой подход, использующий LSTM для формирования векторных представлений текста - ULMFiT [60]. В постановке задачи авторов получение векторов слов не было самоцелью, разработанная архитектура в первую очередь создавалась для текстовой классификации документов. Но, так как по своей сути в архитектуре модели присутствуют и формируются векторные представления слов, их можно получить при необходимости.

Ключевым отличием подхода ULMFiT от ELMO является подход к обучению модели. Авторы перенесли идеи об обучении с переносом (transfer learning) [61] из задач обработки изображений в задачи обработки естественного языка. Для этого обучение модели ULMFiT разделяется на 3 части:

1. Предварительное обучение языковой модели,
2. Тонкая настройка языковой модели на предметную область,
3. Тонкая настройка языковой модели на целевую задачу.

По аналогии с подходами из обработки изображений, где предварительное обучение проводится на таких крупных наборах данных как ImageNet [62], состоящих из миллионов изображений, предварительное обучение производилось над статьями из Википедии в размере 30 тысяч документов.

Тонкая настройка на предметную область целевой задачи необходима для того, чтобы векторные представления захватили семантику конкретной предметной области, так как она может сильно разниться (например литература и соц. сети). Авторы предложили обучать разные слои с разной скоростью обучения на данном этапе, из-за того, что разные слои хранят разные типы информации [63], а также использовать стратегию изменения скорости обучения в виде треугольника. В подобной стратегии изменения скорости обучения,

на первых шагах скорость обучения будет близка к 0, после некоторого небольшого количества шагов (разогрев) она достигнет своего максимума и начнет линейно уменьшаться до минимума к последнему шагу обучения сети.

### 1.4.3 Контекстуализированная векторная модель BERT

Одним из самых сильных прорывов в области обработки естественного языка за последнее десятилетие стала работа ученых из Google [23]. Авторы объединили подход к переносу обучения с недавно созданной архитектурой трансформер [64]. Помимо этого, была предложена задача маскированного языкового моделирования (MLM) в качестве целевой задачи предварительного обучения.

В основе архитектуры трансформер лежит механизм внимания [65], суть которого в том, чтобы формировать выход слоя в зависимости от всех входов слоя с некоторыми весами, которые зависят от самих входов и скрытых обучаемых матриц. Архитектура трансформера состоит из трансформер-кодировщика и трансформер-декодировщика. Трансформер-кодировщик представляет из себя набор трансформер блоков, каждый из которых формирует свой выход для каждого слова или токена, в зависимости от выхода прошлого слоя по описываемому принципу. Трансформер-декодировщик работает схожим образом, но формирование выхода на шаге $t$ и слое $l$ зависит от всех выходов декодировщика на слое $l-1$ и до шага $t$ и дополнительно от выходов трансформер-кодировщика, если такой присутствует.

Более формально о том, как устроен механизм внимания, а именно, механизм самовнимания (self-attention), который и реализован в трансформер-кодировщике. Для каждого слова под номером $t$ и на слое $l$ входной вектор слоя $i_{t,l-1}$ умножается на три обучаемые матрицы $W_q$, $W_k$ и $W_v$, формируя тем самым три вектора: вектор запроса $q$, вектор ключа $k$ и вектор значения $v$ (они являются некоторыми абстракциями). На следующем этапе, на примере слова под номером $t$, вектор запроса $q_{t,l}$ перемножается со всеми векторами ключа $k_{i,l}$, формируя таким образом набор скалярных значений. После чего, эти значения нормируются некоторым образом (на константу, которая зависит от архитектуры сети) и рассчитывается softmax функция для всех получившихся значений,

формируя веса внимания *score*. На последнем этапе веса внимания перемножаются с соответствующими векторами значения $v$ и получившиеся вектора суммируются. Итоговый вектор и будет выходом после механизма внимания для шага $t$ и слоя $l$. Или в матричном виде:

$$\begin{aligned} Q &= Input \times W_q; \\ K &= Input \times W_k; \\ V &= Input \times W_v; \end{aligned} \tag{1.18}$$

$$Attention(Q, V, K) = softmax(\frac{QK^T}{\sqrt{d}}).$$

Механизмом самовнимания подобный подход называется по той причине, что в операциях участвует также вектор текущего слова. Помимо этого, авторы [64] ввели понятие многоблочного самовнимания, когда обучается не одни набор матриц $W_q$, $W_k$ и $W_v$, а $k$, и выход каждого блока в $k$ раз меньше по размерности, чем у исходного вектора, и, при формировании итогового выхода, получившиеся вектора конкатенируются. В случае многоблочного механизма самовнимания вектора $q$, $k$ и $v$ имеют в $k$ раз меньшую размерность $d$. На примере базовой конфигурации архитектуры, количество блоков $k = 12$, $d = 64$, а размерность входов и выходов слоев $h = 768$.

BERT является трансформер-кодировщиком и в базовой конфигурации состоит из 12 трансформер блоков. Используя подход обучения с переносом, модель изначально обучается предсказывать замаскированное в тексте слово (случайные слова во входной последовательности заменены на специальный токен <MASK>) на основании остальных слов, таким образом, неявно используя дистрибутивную гипотезу. После того, как модель обучена, ее можно использовать без последнего слоя классификатора (который занимался предсказанием замаскированного слоя) для получения контекстуализированных векторных представлений слов.

В отличие от контекстуализированных векторных представлений, построенных с использованием LSTM сетей, подобная сеть намного лучше позволяет настраиваться на предметную область и целевую задачу. Для этого слои трансформер-кодировщика не замораживаются во время тонкой настройки на задачу, а обучаются вместе со слоями, необходимыми для целевой задачи (в отличие от подхода ULMFiT, где веса замораживались).

Подобная архитектура с предварительном обучением на достаточно крупной коллекции текстов (гигабайты и более) позволила достичь наилучших результатов на множестве задач обработки естественного языка, превосходя прошлые подходы.

Помимо этого, особенностью архитектуры BERT является то, что текст сегментируется не на слова, но на некоторые токены, которые могут быть как словами, н-граммами, так и символами. Подобный подход мотивирован тем, что полный словарь слов, особенно без нормализации, в больших коллекциях документах очень быстро разрастается и достигает миллионов слов, что приводит к огромным матрицам векторных представлений.

## 1.5 Векторные представления и нейронные сети в задаче пополнения таксономии

Отношения гиперонимии-гипонимии составляют основу структуры множества онтологий и графов знаний [1]. Поэтому многочисленные исследования посвящены извлечению подобных отношений из текстовых коллекций. Гиперонимы могут быть извлечены с нуля, без каких-либо целевых ресурсов или таксономии, но качество таких подходов обычно достаточно низкое и не позволяет строить качественные таксономии, которые можно было бы использовать в рамках других задач. Также задача извлечения гиперонимов может ставиться как задача поиска гиперонимов для новых слов в существующей таксономии, то есть как задача обогащения или пополнения таксономии. Подобная постановка проще с точки зрения как автоматизированного подхода, так и с точки зрения автоматического подхода.

В 2016 г. задача по обогащению таксономии была организована как соревнование на семинаре SemEval (задача 14) [24]. Участники должны связать слова с определениями для исправления гиперонимов в WordNet [16]. Однако, в реальных приложениях определения новых слов и их значений, скорее всего, отсутствуют. В 2020 году было организовано новое соревнование по обогащению таксономии для русского RuWordNet [26] - RUSSE'2020 [25]. Задача состояла в том, чтобы найти правильные гиперонимы из старой версии RuWordNet для слов, добавленных в новой версии RuWordNet. В работе [27] описаны новые на-

боры данных, называемые диахроническими ворднетами (Diachronic wordnets)[1], созданные на основе английских и русских таксономий типа ворднет (WordNet). Эти наборы данных содержат новые слова, добавленные в более поздние версии ворднетов по сравнению с более ранними версиями, вместе с их гиперонимами в более старых версиях. Таким образом, можно использовать разные версии таксономий типа WordNet в их историческом развитии для оценки методов задачи обогащения таксономии.

### 1.5.1 Подходы к предсказанию гиперонимов

Подходы к предсказанию гиперонимов можно условно разделить на три категории:

1. Лингвистические методы (на основе шаблонов),
2. Методы на основе векторных представлений,
3. Комбинированные методы.

#### Лингвистические методы

Одним из первых лингвистических методов предсказания гиперонимии, является использование шаблонов Херста [17] для расчета совместной встречаемости слов на большом корпусе текстов. Данный подход разрабатывался для английского языка и включает в себя такие шаблоны как X or(other) Y; Y, including X; и другие. Если пара слов часто встречалась в подобных шаблонах, можно использовать это как сигнал о таксономическом отношении между словами. Подобные шаблоны могут быть построены не только над текстовым представлением предложения, но и над синтаксическим деревом [66], что позволяет расширить набор шаблонов и повысить качество похода.

Для русского языка авторы работы [67] предложили набор шаблонов для предсказания отношения гиперонимии. Некоторые из предложенных шаблонов выглядят следующим образом:

[1]https://github.com/skoltech-nlp/diachronic-wordnets

- "Такие/таких/таким X, как Y [, Y] и/или Y". Пример сопоставления:
  - В Индии зародились такие [религии]=HYPER как [индуизм]=HYPO, [буддизм]=HYPO, [сикхизм]=HYPO и [джайнизм]=HYPO.
- "X, такие/таких/таким как Y [, Y] [и/или Y].". Пример сопоставления:
  - ... систем [верований]=HYPER, таких как [шаманизм]=HYPO, [политеизм]=HYPO, [пантеизм]=HYPO, [анимизм]=HYPO.
- "Y — вид/тип/форма/разновидность/сорт X.". Пример сопоставления:
  - [Хобби] = HYPO — вид человеческой [деятельности], некое занятие.

Подходы на основе шаблонов имеют ряд недостатков. Вариативность текстов не позволяет покрыть все случаи шаблонами, помимо этого, многие корректные пары слов могут не встречаться в текстах в описанных шаблонах или даже в одном предложении в принципе. Проблему недостаточного покрытия пытались решать как заменой некоторых слов на соответствующие им части речи [68], так и на автоматическую генерацию шаблонов [69], но, хоть и получалось достичь некоторых улучшений, сама проблема покрытия всегда остается актуальной.

### Методы на основе векторных представлений

Другим популярным направлением к решению задачи предсказания гиперонимии стало использование векторных представлений слов. Для этого каждое слово представляется в виде вектора, и на основании векторных представлений а) нового понятия (слова) и б) концептов таксономии, происходит предсказание гиперонимии, обычно методами машинного обучения.

В работе [70] была использована идея о том, что похожие слова в векторном пространстве группируются в некоторых областях, и между векторами можно производить линейные преобразования таким образом, чтобы это имело некоторый смысл. Подобные подходы называются обучением проекции (projection learning). Например, если вычесть из вектора "Германии" вектор "Берлина" и добавить такой вектор к вектору "Москвы", получившийся вектор окажется близко к вектору "Россия". Таким образом, подобное линей-

ное преобразование можно использовать в качестве выражения отношения страна-столица. В работе была предпринята попытка найти такое линейное преобразование для векторов, которое выражало бы таксономическое отношение гипероним-гипоним. Но авторы в итоге пришли к выводу, что не существует одного такого преобразования, поскольку такие преобразования различаются для разных типов слов. Поэтому в работе предложили кластеризовать слова на основе их векторного представления, и строить подобное линейное преобразование $\Phi_k$ для каждой группы слов отдельно.

$$\Phi_k^* = argmin \frac{1}{N_k} \sum_{(x,y) \in C_k} ||\Phi_k x - y||^2. \tag{1.19}$$

Авторы [71] развили предложенный метод, чтобы а) автоматически определять количество кластеров б) обновлять соответствующую матрицу $\Phi_k$ на основе новых пар и негативного семплирования, таким образом достигнув дальнейшего улучшения качества на англоязычных данных (0.766 F1 мера).

Описанные выше подходы в основном решали задачу в такой постановке, что имеется набор пар слов и нужно определить, имеется ли между парой слов отношение гипоним-гипероним. Соотношение позитивных к негативным парам в подобных наборах данных не отличается на порядки в сторону негативного класса, хотя в реальности, решая задачу расширения таксономии, большинство пар для оценки будут именно негативными. По этой причине в 2018 году было проведено соревнование Semeval-2018 Hypernym discovery task [72]. В предложенной постановке имелись а) набор новых слов, б) большая текстовая коллекция, в) словарь возможных гиперонимов (на основе частотности в корпусе) с) список стоп слов, и было необходимо для новых слов выдать упорядоченный список гиперонимов из списка в) по убыванию вероятности. Способ оценки также изменился, вместо accuracy и f1 меры, которые не применимы для задач ранжирования, организаторы соревнования использовали Mean Average Precision (MAP) и Mean Reciprocal Rank (MRR).

Победителем соревнования стал подход CRIM [73], в котором использовалась комбинация шаблонов и методов обучения проекции. В работе использовалось два вида шаблонов: шаблоны гиперонимов и шаблоны ко-гипонимов, а в качестве метода обучения проекции использовался схожий с [71] подход. В результате предложенный подход победил с существенным отрывом

от второго места, но при этом итоговое качество составило только около 0.36 MRR.

В 2020 году было организовано соревнование RUSSE-2020 [25], которое повторяло по своей сути соревнование Semeval-2018 Hypernym discovery task, но уже для русского языка. Участники использовали различные векторные представления (статические - fastText [74], word2vec [39] и контекстуализированные - BERT [23]), доступную структуру таксономии RuWordNet, шаблоны гиперонимов и ко-гипонимов, определения слов из Викисловаря и результаты глобальных поисковых систем [25; 29; 75; 76].

Современные методы поиска гиперонимов могут также использовать графовое представление структуры таксономии. Liu et al. [77] использовали векторные представления node2vec на основе структуры графов [43] для расширения таксономии. Aly et al. [78] использовали гиперболические представления Пуанкаре [79] для автоматического создания таксономий. Сверточные сети с графами (GCN) [48] применяются к задаче прогнозирования связей для больших баз знаний. В [80] авторы исследовали основанные на графах методы представления слов для набора данных Diachronic wordnets.

## 1.6 Векторные представления и нейронные сети в задаче извлечения именованных сущностей

### 1.6.1 Задача извлечения именованных сущностей

Извлечение именованных сущностей (Named-entity recognition, NER) - важный первый шаг в большинстве систем извлечения информации. В частности, для пополнения графов знаний именованными сущностями, их необходимо предварительно извлечь из текстов. Текущие подходы к извлечению именованных сущностей основаны на методах машинного обучения. Для этого требуются текстовые данные, аннотированные именованными сущностями нескольких типов. Большинство известных наборов данных для задач извлечения именованных сущностей состоят из новостных документов с тремя типами именованных сущностей: персона (имена людей), организация (названия организаций), ло-

кация (места, в основном географические объекты) [81—83]. Для этих типов именованных объектов современные методы обычно дают впечатляющие результаты.

Однако, применение подобных систем к новой области может привести к резкой потере точности. Стоимость подготовки аннотированного корпуса для доменных категорий довольно велика. Необходимо установить принципы аннотации для объектов, зависящих от предметной области, и гарантировать, что эти принципы применяются последовательно. Аннотирование в определенных областях может потребовать специальных знаний. Помимо этого, обработка таких текстовых жанров, как сообщения в социальных сетях (например, в Twitter), комментарии и др., может привести к дальнейшей деградации результатов из-за специфики текстовых данных.

### 1.6.2 Методы в задаче извлечения именованных сущностей

#### Методы до нейронных сетей

Ранние системы для решения задачи извлечения именованных сущностей были основаны на вручную созданных правилах, с использованием лексикона, орфографии и онтологий. Например, в работе [84] использовались только 7 признаков, включая орфографию, присутствие заглавной буквы, контекст сущности и др.

Дальнейшим развитием стало применение методов машинного обучения совместно с методами разработки признаков (feature engineering). В работе [85] авторы предложили использовать две модели SVM [86] для классификации начала и конца именованных сущностей. Они использовали различные орфографические признаки, префиксы слов, информацию о соседних словах с различными окнами. В итоге получилось достичь 88.3% F меры на CoNLL 2003 наборе данных.

Методы машинного обучения показывали наилучшие результаты не только в начале 2000-х годов, но и после прихода нейросетевых подходов. Например,

в работе [87] авторы, используя условные случайные поля (CRF) [88], смогли достичь наилучшего на тот момент качества на наборе данных DrugNER.

Условные случайные поля (CRF) являлся одним из лучших подходов до нейронных сетей для задач извлечения именованных сущностей. Подход также активно применялся для других задач разметки последовательностей (sequence labeling), таких как извлечение частей речи [89], сегментация на слова для китайского языка [90]. CRF является ненаправленной графической моделью, которая обучается максимизировать условную вероятность цепочки [88]. CRF с параметрами $\Lambda = \{\lambda_1, ...\}$ определяет условную вероятность цепочки состояний (или меток) последовательности $y = y_1, ..., y_T$ как:

$$P_\Lambda(y|x) = \frac{1}{Z_x} exp(\sum_{t=1}^{T} \sum_k \lambda_k f_k(y_{t-1}, y_t, x, t)), \tag{1.20}$$

где $Z_x$ - коэффициент нормализации, $f_k$ - признаковая функция, $x$ обозреваемая последовательность на шаге $t$, $\lambda_k$ - обучаемые параметры модели.

Наиболее вероятная цепочка меток $y^* = argmax_y(P_\lambda(y|x))$ определяется с помощью алгоритма Витерби [91].

Иными словами, CRF обучается предсказывать наиболее вероятную метку на основании как текущих признаков, так и на основании прошлых меток, что важно в задачах разметки последовательностей.

### Нейросетевые подходы

В 2011 году в работе [92] было предложено одно из первых решений с использование нейронных сетей и сжатого векторного представлений слов. Их система, названная SENNA, смогла достичь наилучших результатов на ряде задач обработки естественного языка, включая извлечение именованных сущностей (набор данных CoNLL 2003), избегая при этом ручного подбора признаков для обучения. В подходе использовались полносвязные и сверточные слои.

В 2015 году продолжилось дальнейшее развитие методов на основе нейронных сетей для задач извлечения именованных сущностей, только уже с использованием сетей LSTM. В работе [93] авторы использовали LSTM модель на словах с дополнительным CRF слоем над LSTM и смогли достичь 84.26% F1

на CoNLL 2003. Схожие системы показали улучшение качества и на других наборах данных.

В работе [94] описан подход, который долгое время считался лучшим решением для задач извлечения именованных сущностей. В основе подхода лежит также LSTM + CRF, но в дополнение к обычным векторным представлениям слов использовались также символьные векторные представления слов и дополнительные признаки (признак заглавной буквы, лексикон). Символьные векторные представления обучаются с использованием сверточных нейронных сетей (CNN), для этого каждое слово представляется как последовательность символов фиксированной длины (если слово короче, то оно дополняется специальным символом). Подобное представление поступает на вход сверточным слоям, после которых идет слой макс пулинга. В результате векторное представление слова формируется как конкатенация трех векторов: обычного векторного представления слов, символьного векторного представления, вектора дополнительных признаков. Предложенный подход позволил достичь 91.62% F1 меры на CoNLL 2003 наборе данных.

### Подходы на основе модели BERT

Современные модели извлечения именованных сущностей используют различные контекстуализированные векторные представления. Одна из таких популярных моделей - BERT [23]. BERT — это реализация языковой модели, основанной на глубоких нейронных сетях. При обучении модели ставится задача предсказать слово в заданном месте текста. Архитектура BERT состоит из 12-уровневого трансформера-кодировщика, который формирует контекстуализированные представления токенов, таким образом преобразуя последовательность токенов в последовательность векторов.

Для использования BERT в задачах извлечения именованных сущностей, последний слой BERT модели, который отвечает за предсказания маскированного токена, отбрасывается, и от модели используется только сам трансформер-кодировщик, на выходе которого для последовательности токенов длины $N$ получается тензор размерности $R^{N \times d}$, где $d$ - размерность скрытых слоев сети. После этого, контекстуализированное представление каждого токена по-

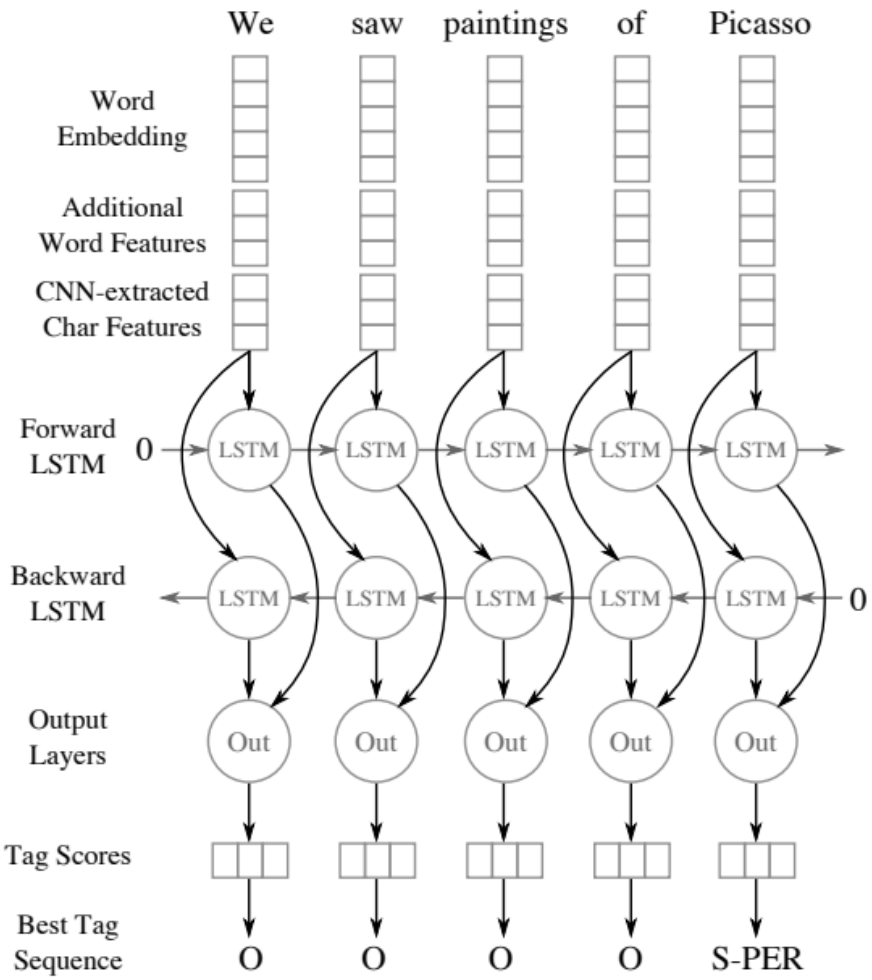


Рисунок 1.5 — Архитектура сети LSTM + CRF + CNN. Заимствовано из [94]

дается на вход слою классификации. Данный слой может быть представлен любой архитектурой для классификации, которая работает с векторными представлениями, но базовым вариантом является простой линейный слой, с softmax функцией активации на выходе.

$$P(c|(x_1, ..., x_n), t) = Softmax(W \times E(x_t, (x_1, ..., x_n)) + b), \tag{1.21}$$

где $P(c|(x_1, ..., x_n), t)$ - вероятностное распределение среди меток $c$ для токена $x_t$, $E(x_t, (x_1, ..., x_n))$ - выход из модели BERT для токена $x_t$, $W$ и $b$ - обучаемые веса классификатора.

В таком варианте сами веса модели BERT обычно не "замораживаются", а обучаются совместно со слоем классификации на целевой задаче. Пример подобной архитектуры проиллюстрирован на Рис. 1.6.

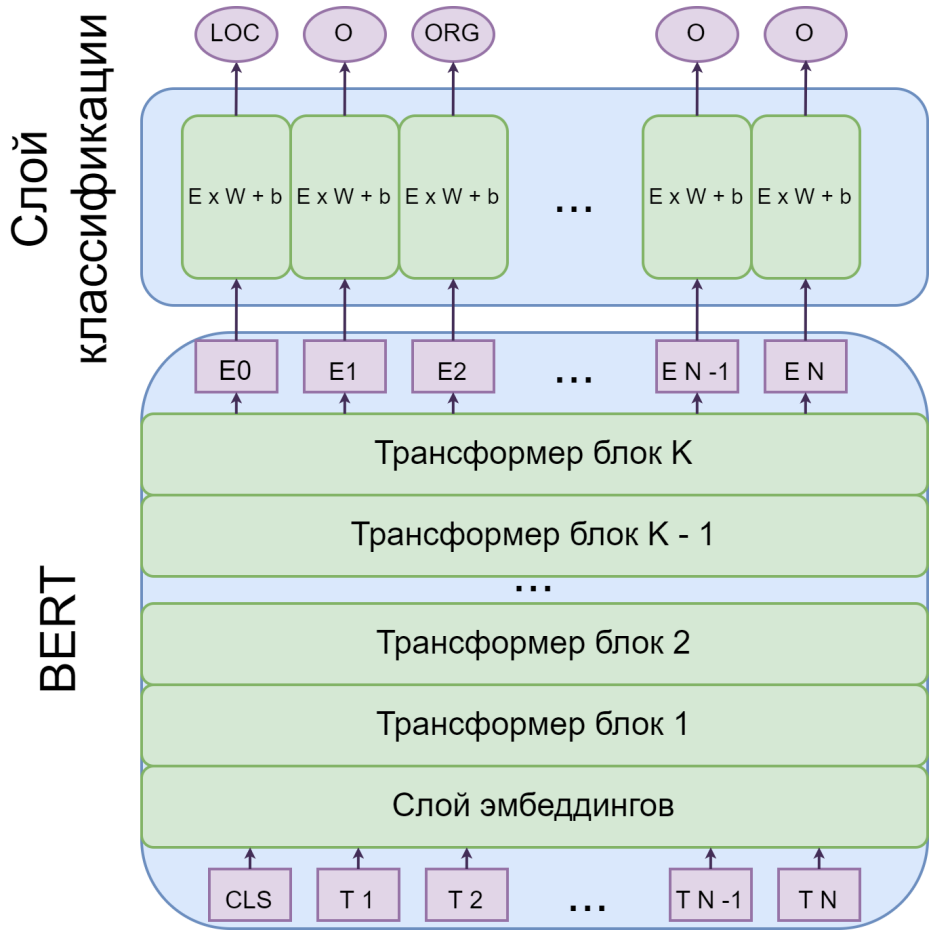


Рисунок 1.6 — Архитектура BERT для извлечения именованных сущностей

Использование BERT позволило достичь лучших результатов в различных задачах обработки естественного языка [23], включая извлечение именованных сущностей (достигая 92.8% F1 на CoNLL 2003). Такие результаты обусловлены высокой информативностью векторных представлений, которые, в отличие от статических векторных представлений, таких как Word2Vec [21], зависят от контекста. Кроме того, немаловажным моментом является использование методик обучения с переносом. BERT предварительно обучается на большом количестве не размеченных данных на задаче языкового моделирования, а затем настраивается для конкретной задачи.

Изначально BERT был многоязычным, обученным на многоязычных данных. В статье [95] описан подход к дальнейшему обучению многоязычной модели на русскоязычных данных (русская Википедия и русскоязычный корпус новостей). Новая модель, получившая название RuBERT [95], показала улучшение качества в трех задачах обработки естественного языка на русском языке по сравнению с предыдущими результатами и многоязычным BERT. Использова-

ние RuBERT в задаче извлечения именованных сущностей для русскоязычного набора данных Collection3 [82] также дало значительное улучшение [96].

В 2019 году была организована совместная дорожка по задаче извлечения именных сущностей для славянских языков [97]. Большинство участников и победитель использовали BERT в качестве основной модели. Следует отметить значительный дисбаланс между типами сущностей в данных. Например, тип сущности "продукт" (PRO) был аннотирован только для 8% всех сущностей в русскоязычных данных. Качество извлечения сущностей этого типа было значительно ниже, чем для других сущностей, что ставит вопрос об улучшении качества извлечения редких сущностей в несбалансированных наборах данных.

### 1.6.3 Извлечение именованных сущностей в области информационной безопасности

Задача извлечения информации в области информационной безопасности обсуждалась в нескольких работах. Однако в большинстве работ извлечение информации рассматривается только из структурированных или частично структурированных английских текстов. Например, Bridges et al. [98] использовали корпуса, состоящие в основном из Microsoft Bulletins и описаний национальных баз данных уязвимостей. Корпус обучения, представленный в [99], действительно содержит неструктурированные сообщения в блогах, но они составляют менее 10% корпуса.

Предлагаемые подходы основаны на таких методах, как принцип максимальной энтропии [98], условные случайные поля (CRF) [88; 99]. Gasmi et al. [100] исследовали два разных подхода к задаче извлечения именованных сущностей: CRF-модель и модель LSTM-CRF на основе нейронной сети (NN) (как было предложено Lample et al. [101]). Модель на основе нейронной сети объединила двунаправленную LSTM, вектора Word2Vec в качестве источника предварительно обученных векторов слов и CRF в качестве выходного слоя.

В [102] был представлен корпус по информационной безопасности Sec_col для извлечения именованных сущностей в русскоязычных текстах. Корпус содержит неструктурированные тексты, он был собран из журнальных статей, новостных сообщений и сообщений на форумах. Все эти данные могут

предоставить дополнительную информацию о проблемах информационной безопасности. Авторы сравнили различные модели для извлечения именованных сущностей на представленном корпусе, такие как CRF и несколько вариантов нейронных сетей.

# Глава 2. Пополнение таксономии графов знаний новыми понятиями

При работе над данной главой диссертации использованы следующие публикации автора, в которых, согласно Положению о присуждении ученых степеней в МГУ, отражены основные результаты, положения и выводы исследования:

1. Tikhomirov M., Loukachevitch N., Parkhomenko E. Combined approach to hypernym detection for thesaurus enrichment // Computational Linguistics and Intellectual Technologies. — 2020. — С. 736—746,
2. Tikhomirov M., Loukachevitch N. V. Domain-specific Taxonomy Enrichment based on Meta-Embeddings // CEUR Workshop Proceedings. Т. 3036. — 2021. — С. 285—298,
3. Tikhomirov M., Loukachevitch N. Meta-Embeddings in Taxonomy Enrichment Task // Computational Linguistics and Intellectual Technologies: papers from the Annual conference Dialogue. — 2021. — С. 681—691,
4. Loukachevitch N., Tikhomirov M., Parkhomenko E. Using Embedding Based Similarities to Improve Lexical Resources // Lobachevskii Journal of Mathematics. — 2021. — Т. 42, No 7. — С. 1532—1546,
5. Nikishina I. [и др.]. Taxonomy Enrichment with Text and Graph Vector Representations // Semantic Web. — 2022. — Т. 13, No 3.

## 2.1 Постановка задачи

Графы знаний могут быть использованы для решения многих задач обработки естественного языка [103; 104], но их польза напрямую зависит от полноты используемой базы знаний и возможностей отражать понятия реального мира. Но, так как существует множество узких предметных областей, не существует баз знаний, которые покрывают понятийное пространство всех областей. Помимо этого, даже существующие базы знаний необходимо поддерживать и пополнять, так как в любой области со временем появляются новые понятия или изменяются старые.

Основой многих онтологий, баз знаний и графов знаний являются таксономии, которые организуют понятия (также называемые концептами) базы знаний, через отношение частичного порядка является или гипероним-гипоним [1]. Гипероним является вышестоящим в таксономии для гипонима понятием, например кошка является также и животным. Таксономии представляют собой ориентированный граф без циклов, а само отношение гиперонимии является при этом транзитивным и антисимметричным [105]. То есть, если класс Б является подклассом класса А, то все подклассы Б тоже являются подклассами класса А. Подобная иерархия необходима для использования наследования в процедуре логического вывода, который в свою очередь необходим при анализе текстов на естественном языке [105—107]. Более формально, отношение $R$ называется отношением частичного порядка на некотором множестве $M$, если оно удовлетворяет следующим свойствам [108]:

1. Рефлексивность. $\forall a \in M : (aRa)$,
2. Транзитивность. $\forall a,b,c \in M : (aRb) \Lambda (bRc) \Rightarrow (aRc)$,
3. Антисимметричность. $\forall a,b \in M : (aRb) \Lambda (bRa) \Rightarrow a = b$.

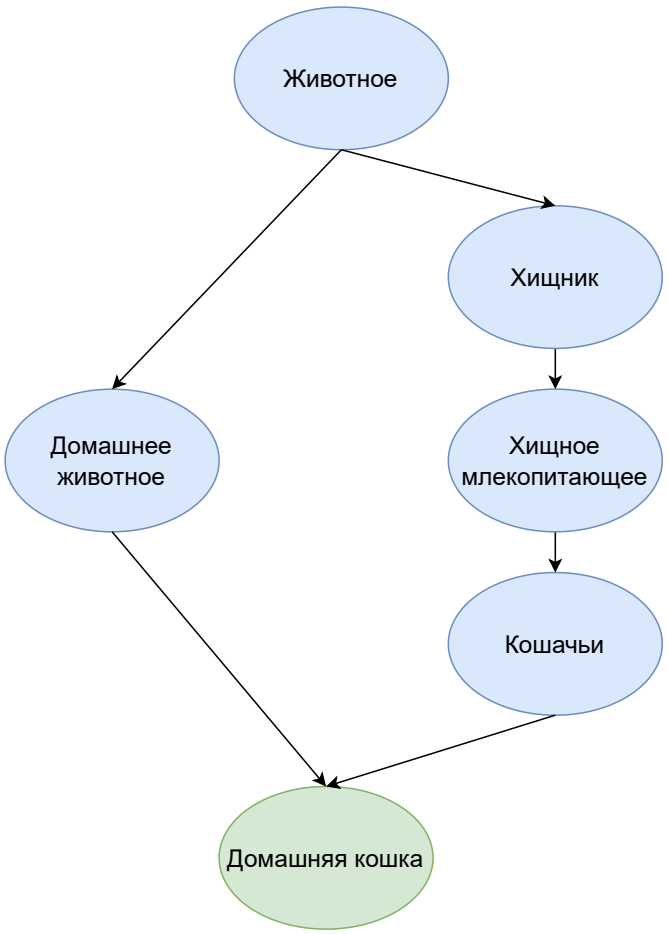


Рисунок 2.1 — Пример части таксономии

Для того, чтобы упростить создание и адаптацию подобных ресурсов на новые предметные области, в работе исследуются методы пополнения существующих таксономий новой терминологией. Задача рассматривается именно в такой постановке, так как построение таксономий с нуля является крайне сложной задачей даже для человека. Более целесообразно рассматривать некоторую таксономию из общей предметной области и пополнять ее новыми понятиями из новой, узкой предметной области. В данной работе рассматриваются методы поиска правильных гиперонимов для новых слов, не ставя задачу извлечения и поиска новых понятий из сырых текстов.

Пусть имеется существующая таксономия и набор новых понятий, представленных в виде слов. Требуется присоединить новые слова к существующей таксономии посредством нахождения гиперонимов для этих слов. Для этого для каждого слова требуется сформировать упорядоченный список фиксированной длины концептов-кандидатов в гиперонимы из таксономии в порядке убывания веса уверенности. В работе рассматриваются две основные постановки: а) пополнение таксономии из общей области новыми словами, которые также из общей области, и б) пополнение таксономии из общей области новыми словами из предметной области информационной безопасности.

Задачу пополнению таксономии, можно также условно разделить на 3 типа: а) пополнение таксономии только абстрактными сущностями (без имен), б) пополнение таксономии только именованными сущностями, в) смешанный вариант. Подобная ситуация сложилась потому, что именованные сущности отличаются от абстрактных сущностей, как по контекстам, так и по статистическим и семантическим характеристикам. Помимо этого, можно классифицировать постановки задач по тому, происходит ли пополнение таксономии на общей предметной области или на конкретной предметной области.

Для решения поставленной задачи могут использоваться следующие ресурсы:

- Таксономия,
- Корпус текстов из конкретной предметной области,
- Корпус текстов из общей области,
- Статическая векторная модель (модели), обученная на общей или конкретной предметной области,
- Контекстуализированная векторная модель, обученная на общей или конкретной предметной области,

– Шаблоны, полученные из текстов на общей или конкретной предметной области (шаблоны гиперонимов, шаблоны ко-гипонимов),
– Внешние словари (Викисловарь или др.).

Описанные ресурсы схематически представлены на Рис. 2.2.


Рисунок 2.2 — Ресурсы в задаче пополнения таксономии

При построении статистических векторных представлений также может быть применена специальная обработка именованных сущностей, чтобы устранить контексты именованных сущностей для многозначных слов.

Алгоритмы, которые используют разные по типу ресурсы для пополнения таксономии, называют комбинированными. В работе реализованы две вариации комбинированного подхода для пополнения таксономии новыми словами с различным набором используемых ресурсов.

## 2.2 Подходы

### 2.2.1 Комбинированный подход на основе шаблонов и векторных представлений слов

#### Описание алгоритма

В рамках работы над задачей пополнения таксономии был разработан комбинированный подход к предсказанию гиперонимов на основе как векторных представлений слов, так и шаблонов. Помимо этого, подход направлен на использование только корпуса текстов, из которого новые слова для пополнения и отбирались. Метод использует следующие ресурсы:

- Дистрибутивное (векторное) представление анализируемых слов и словосочетаний, обученные на общей предметной области,
- Лингвистические шаблоны для гиперонимов и ко-гипонимов,
- Специальная обработка именованных сущностей для исключения их контекстов из рассмотрения,
- Применение классификатора на основе контекстуализированной модели BERT для подтверждения кандидатов в гиперонимы;

Данный подход можно отнести к пополнению таксономии только абстрактными понятиями из общей предметной области.

В качестве алгоритма для обучения векторных представлений использовался PPMI + SVD подход. Для этого:

- На первом шаге рассчитывается матрица совместной встречаемости слов в корпусе (в некотором окне),
- Веса совместной встречаемости слов пересчитываются с использованием меры положительной точечной взаимной информации (PPMI) [18],
- Применяется метод SVD над матрицей PPMI, который позволяет уменьшить размерность матрицы от размера словаря до выбранного значения,

– При обработке корпуса, фразы из таксономии были объединены в единые токены, например рисовое_зерно. Модели были рассчитаны с размером вектора 600 и разными размерами окон: 1, 3, 5.

Подход PPMI + SVD продемонстрировал сравнимое качество с word2vec в ряде экспериментов [18] и показал лучшие результаты в данном случае, по сравнению с другими подходами, к построению векторных моделей на этапе предварительных экспериментов.

На первом шаге алгоритма для всех целевых (новых) слов были рассмотрены 100 наиболее близких слов-кандидатов на основе косинусной близости между векторами слов. Отобранные слова должны присутствовать в таксономии. Затем идет шаг формирования списка концептов-кандидатов. Для каждого слова-кандидата извлекались из таксономии его гиперонимы и гиперонимы второго порядка (со штрафным весом). Каждый концепт мог быть добавлен несколько раз. Поэтому для каждого концепта кандидата фиксировалось, сколько раз он был добавлен (count) и на основе каких значений близости между целевым словом и словом-кандидатом, формируя таким образом список значений близости (cos_sim_list). Кандидаты были ранжированы по следующей формуле:

$$base\_score = mean(cos\_sim\_list) * log_2(1 + count) * \alpha, \tag{2.1}$$

Следующим шагом алгоритма был учет шаблонов. Соответствие шаблонам проверялось для каждой пары "целевое слово" - "слово-кандидат" из списка наиболее близких слов по векторной модели. Появление соответствующей пары в шаблоне увеличивало вес близости слов-кандидатов целевому слову, так как соответствие шаблону является дополнительным доказательством семантического сходства. Например, проверялось, существует ли сопоставление с шаблоном для целевого слова "пион" и слова-кандидата "роза". Если такой случай имел место быть, то это свидетельство того, что роза и пион могут быть синонимами или ко-гипонимами.

Были рассмотрены два типа шаблонов:

– Шаблоны ко-гипонимов, успешное сопоставление с которыми приводит к увеличению веса соответствующих концептов,
– Шаблоны гиперонимов, успешное сопоставление с которыми приводит как к увеличению веса концепта-гиперонима, так и к дополнительному

включению в список кандидатов прямого концепта слова-кандидата, а не только его гиперонимов.

В Таблицах 1, 2 приведены образцы шаблонов, где X - целевое слово, Y - слово-кандидат, а W - любое слово. Также для всех шаблонов допускаются варианты, когда X и Y меняются местами.

Таблица 1 — Примеры шаблонов ко-гипонимов

| Шаблон | Пример |
|---|---|
| X, W, Y | кошка, W, собака |
| X, Y | кошка, собака |
| X и Y | кошка и собака |
| X или Y | кошка или собака |

Таблица 2 — Примеры шаблонов гиперонимов

| Шаблон | Пример |
|---|---|
| Y (X | животное (кошка |
| X - Y | кошка - животное |
| X — это Y | кошка — это животное |
| X, W и еще один Y | кошка, W и еще одно животное |
| X и W — это тип Y | кошка и W — это тип животных |

Информация о сопоставлении с шаблонами учитывается в формуле расчета веса для ранжирования. Также в случае шаблонов ко-гипонимов дополнительно рассчитывается частотность случая, когда целевое слово и слово кандидат содержались в одном предложении в корпусе. Модифицированная формула выглядит следующим образом:

$$upd\_pattern\_score = base\_score * (1 + hyper\_hit) * (1 + \frac{2 * co\_hypo\_count}{one\_sent\_count + 2}) \quad (2.2)$$

, где hyper_hit - факт сопоставления с шаблоном гиперонимов, co_hypo_count - частота сопоставления с шаблоном ко-гипонимов в корпусе текстов, one_sent_count - частота попадания целевого слова и слова кандидата в одно предложение в корпусе текстов.

Контекстуализированная векторная модель BERT [23] может использоваться для задачи предсказания гиперонимов [109] в качестве классификатора пар (слово, концепт-кандидат) на то, имеется ли между парой отношение гиперонимии или нет. Поэтому в данной работе был реализован следующий подход:

– Задача рассматривается как задача бинарной классификации,

- На вход модели BERT подается пара слов (или многословных выражений) в следующем виде:
  - [CLS] word1 [SEP] word2 [SEP],
  - Пример после токенизации:
    [CLS] кош ##ечка [SEP] домашнее животное [SEP];

  Если word2 является гипернимом word1, то класс примера равен 1, в противном случае - 0,
- Данные для обучения могут быть автоматически созданы из пополняемой таксономии. Для каждого положительного примера были добавлены три отрицательных примера. Отрицательные примеры были равномерно выбраны из: случайных концептов, гиперонимов второго порядка, гипонимов и гипонимов гиперонимов.

Модель обучалась 5 эпох со скоростью обучения 2e-4 для слоя классификации и 2e-5 для слоев BERT. Набор обучающих данных состоял из 1.5 миллионов примеров.

Используя обученный классификатор, для каждого концепта-кандидата вычислялась его вероятность быть гиперонимом и максимум из вероятностей того, что гипонимы концепта являются гиперонимом целевому слову. Для вычисления вероятности концепта, вычислялась вероятность быть гиперонимом для каждого текстового входа концепта, затем эти значения усреднялись. Результирующая вероятность для концепта вычислялась следующим образом:

$$bert\_prob = 0.6 * syn\_bert\_prob + 0.4 * max\_hyp\_syn\_bert\_prob \tag{2.3}$$

, где syn_bert_prob - вероятность концепта кандидата быть гиперонимом, max_hyp_syn_bert_prob - максимальная вероятность среди всех гипонимов.

Окончательная модифицированная формула ранжирования выглядит следующим образом:

$$synset\_score = upd\_pattern\_score * (1 + bert\_prob) \tag{2.4}$$

### Специальная обработка именованных сущностей

При расчете векторных моделей была произведена специальная обработка именованных сущностей, с целью исключения их контекстов из рассмотрения

при формировании векторных представлений. В работах других исследователей было обнаружено, что именованные сущности могут включать в себя общие слова и таким образом искажать контекст обычных слов [110; 111]. Например, в работе Лукашевич [111] было обнаружено слово мистраль, что означает «сильный, холодный, северо-западный ветер в северном Средиземноморье» и связано с гиперонимом «ветер» в RuWordNet, но в текущих новостных статьях на русском языке это слово в основном означает класс французских вертолетоносцев.

Поэтому была применена следующая процедура предварительной обработки: если вхождение слова пишется с заглавной буквы, и это не первое слово в предложении, то такое вхождение получает специальный префикс. Таким образом, подобное вхождение и его контексты исключаются из расчета вектора для соответствующего слова.

### 2.2.2 Комбинированный подход на основе мета-векторных представлений слов

#### Описание алгоритма

Недостатками комбинированного подхода на основе шаблонов и векторных представлений слов являются: необходимость ручной разработки шаблонов, высокая вычислительная сложность из-за использования модели BERT и сложность адаптации на новые области, так как веса в формулах требуется подбирать индивидуально под каждую область. По этой причине в работе был разработан комбинированный подход на основе мета-векторных представлений слов, который полагается на векторные представления слов и машинное обучение.

В основе подхода лежит использование векторных представлений слов. При обработке нового слова на первом этапе алгоритма происходит поиск наиболее близких слов на основании косинусной меры близости в векторном пространстве. Затем этот список фильтруется с требованием, чтобы оставшиеся в нем слова присутствовали в таксономии, и из них оставляются $N$ слов. На следующем этапе для каждого слова получившегося списка из таксономии

извлекаются а) концепты б) гиперонимы концептов в) гиперонимы второго порядка. Извлеченные понятия добавляются в новый список, фиксируя при этом каким образом был добавлен концепт (как прямой концепт, его гипероним или гипероним второго порядка). Любой концепт может быть добавлен в список кандидатов концептов несколько раз. Эта информация аккумулируется, а список преобразуется в уникальный список, где каждый концепт встречается единожды. Информация о том, сколько раз концепт был добавлен и какими способами, сохраняется. Получившийся список $C_{cand} \in C$, где $C$ это множество всех концептов, представляет собой усеченное пространство для поиска концептов, которые могут являться гиперонимами рассматриваемого слова.

Подобный подход является пословным, так как поиск начинается с поиска похожих слов. Альтернативным вариантом является поконцептный подход. В поконцептном варианте каждый концепт таксономии отображается в общее со словами векторное пространство, путем усреднения векторов слов в концепте, и они добавляются в векторную модель. Соответственно, на этапе получения близких слов для рассматриваемого слова, убираются из рассмотрения все варианты, которые не являются концептами. Дальнейшие шаги алгоритма остаются прежними.

На следующем этапе алгоритма для кандидатов-концептов вычисляется некоторое количество характеристик-признаков для использования их для классификации методом машинного обучения. Расчетные характеристики следующие:

- Максимальное, минимальное и среднее значение близости между целевым словом и синонимами (то есть текстовыми входами) в концепте кандидате,
- Значения близости между целевым словом и синонимами в концептах-гипонимах концепта-кандидата. Сначала рассчитываются максимальное, минимальное и среднее значения для каждого гипонима, затем вычисляются максимальное, минимальное и среднее значения близости среди всех концептов-гипонимов,
- Позиционный признак (0, 1, 2): если кандидат добавлен как прямой концепт (0) или гипероним (1) или гипероним второго порядка (2). Если концепт был добавлен несколько раз, то берется минимум, максимум и среднее,

– Количество раз, сколько концепт попал в список кандидатов $c$, и его логарифм $log_2(2 + c)$.

После описанной обработки, используя алгоритм логистической регрессии, производится предсказание того, является ли обрабатываемый концепт гиперонимом или нет. Таким образом, каждый концепт-кандидат получает свою вероятность, на основании которой производится ранжирование итогового списка.

## Мета-векторные представления

Так как в основе подхода лежит использование векторных представлений слов, то качество самих векторных представлений имеет существенной значение. Поэтому в работе сравниваются как отдельные векторные модели, так и их комбинации, в виде мета-векторных представлений. Рассматривались как простые мета-векторные представления, такие как конкатенация исходных векторов и SVD над конкатенацией, так и два варианта автокодировщиков: конкатенированные автоматически закодированные мета-вектора (CAEME) и усредненные автоматически закодированные мета-вектора (AAEME), которые показали хорошие результаты на ряде задач обработки естественного языка в предыдущих работах [56]. Отдельно стоит отметить, что AEME подходы ранее не применялись для задач пополнения таксономии.

Предположим, есть два исходных векторных представления $s_1(w)$ и $s_2(w)$, их кодировщики $E_1(w)$ и $E_2(w)$ и их декодировщики $D_1(w)$ и $D_2(w)$. Мета-вектор $m(w)$ в CAEME строится как $L_2$ -нормализованная конкатенация двух закодированных исходных векторов $E_1(s_1(w))$ и $E_2(s_2(w))$:

$$m(w) = \frac{E_1(s_1(w)) \oplus E_2(s_2(w))}{||E_1(s_1(w)) \oplus E_2(s_2(w)||_2} \tag{2.5}$$

,где $\oplus$ — это операция конкатенации.

Кодировщики и декодировщики $E_i(w)$ $D_i(w)$ представляют собой полносвязный слой нейронной сети с ReLU активацией на выходе. В случае с кодировщиками перед полносвязным слоем также расположен dropout слой.

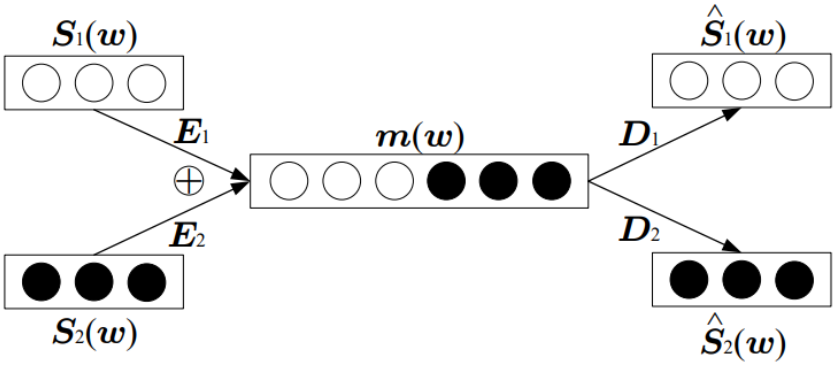


Рисунок 2.3 — Архитектура CAEME. Заимствовано из [56]

В CAEME размерность пространства мета-векторов — это сумма размерностей исходных векторных моделей. Кодировщик AAEME можно рассматривать как случай кодировщика CAEME, где мета-вектор вычисляется путем усреднения закодированных векторов вместо их конкатенации. Усреднение дает возможность избежать увеличения размерности мета-векторов. Кодировщик AAEME вычисляет мета-вектор слова $w$ из двух исходных векторов $s_1(w)$ и $s_2(w)$ как $L_2$ - нормализованную сумму двух закодированных векторных представлений $E_1(s_1(w))$ и $E_2(s_2(w))$.

$$m(w) = \frac{E_1(s_1(w)) + E_2(s_2(w))}{||E_1(s_1(w)) + E_2(s_2(w))||_2} \tag{2.6}$$

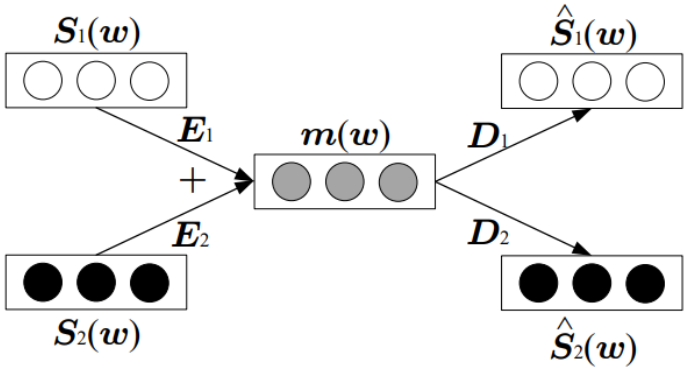


Рисунок 2.4 — Архитектура AAEME. Заимствовано из [56]

Декодировщики CAEME и AAEME восстанавливают исходные вектора из одного и того же мета-пространства $m(w)$, тем самым неявно используя как

общую, так и дополнительную информацию в исходных векторах:

$$\hat{s}_1(w) = D_1(m(w))$$
$$\hat{s}_2(w) = D_2(m(w)) \tag{2.7}$$

Целевая функция обучения автокодировщиков приведена в 2.8. Функция $f$ может быть любым расстоянием или мерой подобия, например MSE, KL-дивергенцией или косинусным расстоянием. Коэффициенты $\lambda_1$ и $\lambda_2$ могут использоваться для взвешивания функций ошибок различных векторных представлений, в случае, когда есть понимание, что какое-либо векторное представление является более значимым.

$$Loss_w(E_1, E_2, D_1, D_2) = \sum_w (\lambda_1 f(s_1(w), \hat{s}_1(w)) + \lambda_2 f(s_2(w), \hat{s}_2(w))) \tag{2.8}$$

Совместное обучение $E_1$, $E_2$, $D_1$, $D_2$ минимизирует общую ошибку реконструкции векторов. Для получения мета-векторных представлений после обучения используются только кодировщики, которые преобразуют входные векторные представления в мета-представление. Далее эти мета-вектора используются как векторные представления слов.

Вместе с обучением модели реконструировать вектора в диссертации предлагается добавить ограничения на обучаемые мета-представления, используя дополнительную функцию потерь - функцию потерь триплетов (triplet loss). Функция потерь триплетов (triplet loss) — это функция потерь для алгоритмов машинного обучения, где некоторый базовый пример (anchor) сравнивается с положительным и отрицательным примерами. Цель - минимизация разницы расстояний между базовым и положительными примерами и базовым и отрицательными. При этом часто присутствует некоторый параметр зазора (margin), который регулирует то, насколько расстояние до отрицательного примера больше, чем до положительного. Одна из первых формулировок подхода эквивалентная потере триплетов, была введена в работе [112] для задачи метрического обучения. Использование подобный функции потерь для модификации алгоритмов также использовалось в задачах близости изображений [113], распознавания лиц [114], классификации текстов [115] и других задачах.

Применительно к задаче обучения мета-векторных представлений для пополнения таксономии идея состоит в том, чтобы поощрять модель за близость

слов, которые семантически связаны по таксономии. Для этого требуется, чтобы схожесть слова к случайному слову из его окружения $C_{neib}$ была выше, чем к случайно выбранному слову из таксономии.

1. Для каждого слова, присутствующего в таксономии, составляется список $C_{neib}$ семантически связанных слов из синонимов, гипонимов и гиперонимов,
2. На каждой эпохе обучения случайным образом выбирается $K$ положительных примеров из $C_{neib}$ и $K$ отрицательных примеров из словаря,
3. Если обрабатываемое слово не представлено в таксономии, то зашумленные вариации исходного вектора рассматриваются как положительные примеры,
4. Затем рассчитывается функция потерь триплетов и итоговая функция потерь выглядит следующим образом: $\alpha * loss + (1 - \alpha) * triplet\_loss$, где $\alpha$ - параметр модели.

Также, описанная модификация AEME алгоритмов с помощью дополнительной функции потерь включает в себя ряд параметров:

– Вид функции потерь: MSE или косинусное расстояние,
– Количество пар (count) положительных и отрицательных примеров при обучении для каждого слова,
– Зазор (margin) между близостью положительных и отрицательных примеров,
– Параметр $\alpha$, регулирующий соотношение между базовой функцией потерь и функций потерь триплетов.

Предложенный подход к пополнению таксономии схематически представлен на Рис. 2.5.

### Добавление признаков из Викисловаря

Некоторые предыдущие подходы [25; 75] для предсказания гиперонимов, помимо векторных представлений, также использовали признаки на основе анализа статей из Викисловаря. По этой причине были предприняты шаги по использованию Викисловаря для улучшения качества работы алгоритма.

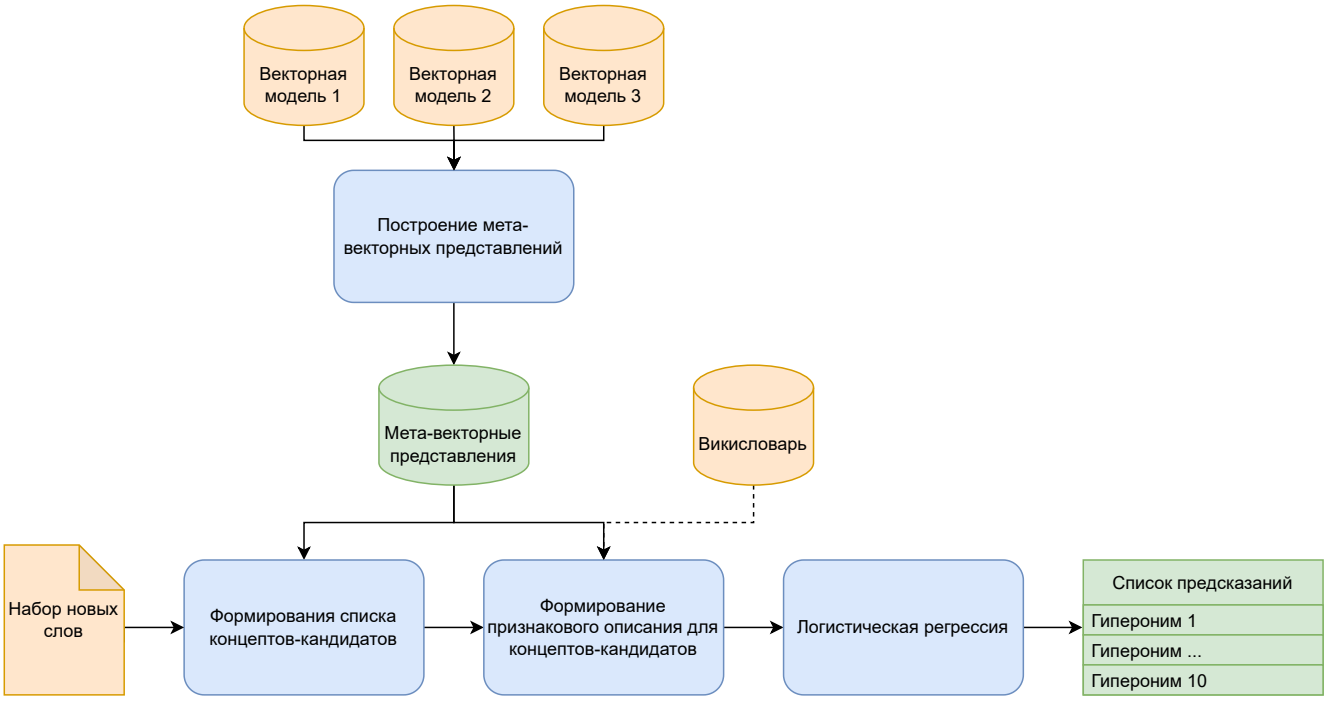


Рисунок 2.5 — Схема работы комбинированного подхода на основе мета-векторных представлений слов

Каждая страница Викисловаря обычно содержит определение и списки гиперонимов, гипонимов и синонимов, которые могут быть полезны для задачи предсказания гиперонимов. В дополнение к уже описанным признакам для работы алгоритма, были реализованы следующие признаки:

– слово-кандидат присутствует в списке гиперонимов Викисловаря для целевого слова (бинарное значение),
– слово-кандидат присутствует в списке синонимов Викисловаря для целевого слова (бинарное значение),
– слово-кандидат присутствует в определении Викисловаря для целевого слова (бинарное значение),

Определения не использовались напрямую, так как их тексты слишком зашумлены. Они часто включают примеры использования слов, которые нельзя отделить от определений и которые могут исказить векторные представления. Поэтому использовался только признак наличия слова в самом определении. Полученные признаки добавляются к описанному ранее классификатору логистической регрессии.

## 2.3 Описание данных и меры оценки

Для тестирования разработанных подходов в работе использовалось несколько разных наборов данных.

### 2.3.1 Набор данных RUSSE'2020

Первый рассматриваемый набор данных — это набор RUSSE'2020, который был создан [25] в рамках соревнования по обогащению таксономии RuWordNet [26]. В представленном наборе данных существует разбиение на существительные и глаголы. Также тестовые данные разбиты на 2 вида: публичные (public) и скрытые (private). Во время соревнования оценка качества на скрытых данных производилась для участников только единожды, в конце события, а публичные данные были доступны все время. Сам же набор данных был построен на разнице между RuWordNet первой версии и RuWordNet второй, расширенной версии. Все слова в наборе данных являются однословными, то есть, многословные выражения в нем не представлены. В качестве гиперонимов золотого стандарта рассматривались не только прямые гиперонимы, но и гиперонимы второго порядка: гиперонимы гиперонимов. Цель создателей набора данных заключалась в том, чтобы сделать оценку менее строгой, но сохранить ее достаточно точной. Статистика по набору данных представлена в Таблице 3.

Таблица 3 — Статистика RUSSE'2020

| **Набор данных** | **Сущ.** | **Глаг.** |
|---|---|---|
| *Public* | 763 | 175 |
| *Private* | 1 525 | 350 |

Также к описанному набору данных прилагался набор текстовых документов News2017, который состоит из новостных статей за 2017 год. Общий объем News2017 составляет около 8 миллионов статей. При составлении набора данных RUSSE'2020 организаторы использовали News2017, фильтруя все слова, которые не входили в News2017 хотя бы 50 раз.

### 2.3.2 Набор данных Diachronic wordnets

Diachronic wordnets [27] состоит из двух наборов диахронических вордне-тов: один для английского, другой для русского, основанный соответственно на таксономиях Princeton WordNet [16] и RuWordNet [26]. Каждый набор данных содержит таксономию и набор новых слов, которые необходимо добавить в этот ресурс. Статистика по количеству слов в наборах данных представлена в Таблице 4.

Таблица 4 — Статистика Diachronic wordnets

| **Набор данных** | **Сущ.** | **Глаг.** |
|---|---|---|
| *WordNet1.6 - WordNet3.0* | 17 043 | 755 |
| *WordNet1.7 - WordNet3.0* | 6 161 | 362 |
| *WordNet2.0 - WordNet3.0* | 2 620 | 193 |
| *RuWordNet1.0 - RuWordNet2.0* | 14 660 | 2 154 |
| *RUSSE'2020* | 2 288 | 525 |

Для составления набора данных на английском языке были взяты пары версий WordNet, а затем выбраны слова, которые появляются только в более новой версии (WordNet 3.0). Для каждого нового слова были извлечены его гиперонимы, которые рассматриваются как "золотой стандарт". Новые слова добавлялись в набор данных, только в том случае, если их гиперонимы присутствовали в обеих версиях вордне-тов. Учитывались только существительные и глаголы. Гиперонимами золотого стандарта, также, как и в RUSSE'2020, являются и прямые гиперонимы и гиперонимы второго порядка.

Для создания аналога английскому набору данных для русского языка, была использована таксономия RuWordNet [26]. Были взяты текущая версия RuWordNet и расширенная версия RuWordNet, и на основании их разницы был составлен набор данных.

Также в Diachronic wordnets включен набор данных RUSSE'2020 в виде объединения публичных и скрытых тестовых данных. RUSSE'2020 можно рассматривать как ограниченное подмножество русскоязычного набора данных с исключением из него нескольких категорий слов (короткие слова, географические и именованные сущности и другие).

### 2.3.3 Набор данных для адаптации таксономии на предметную область информационной безопасности

Все представленные выше наборы данных содержат в себе понятия из общей предметной области. Для того, чтобы изучить качество работы разработанных методов для адаптации таксономии на конкретную предметную область, в работе был создан новый набор данных OENTCyber. Этот набор данных основан на Онтологии по Естественным Наукам и Технологиям (OEHT) [116—118] и текстовом корпусе по информационно безопасности. Онтология представлена как семантическая сеть понятий и отношений между ними, каждое понятие связано с набором слов и словосочетаний, которые могут выражать это понятие в документах. Онтология используется для автоматического анализа документов в информационно-аналитических системах, который включает обеспечение концептуального поиска, расширение запроса с использованием онтологии, категоризацию документов на основе знаний и т.д.

OEHT включает в себя большие объемы концептов и терминологии из нескольких научных дисциплин и технологических областей, представленных в виде единой семантической сети концептов с соответствующими текстовыми записями и отношениями между концепциями [116]. Построение такой онтологии началось с создания текстовых наборов данных (специализированных веб-сайтов, школьных и университетских учебников) по математике, физике, геологии, биологии и химии. В настоящее время эта первоначальная терминология собрана в подмножестве OEHT под названием OEHT-lite.

В дальнейших проектах доступные концептуальные структуры были доведены до более конкретных уровней, а также в OEHT была добавлена терминология технологических областей, таких как нефтегазовая промышленность, энергетика, образовательная политика и методы, компьютерные технологии и информационная безопасность. Полная версия OEHT состоит из 106 тыс. концептов и 308 тыс. различных слов или многословных выражений, тогда как OEHT-lite состоит из 37 тыс. концептов и 133 тыс. слов / фраз. Концепты в OEHT не делятся на существительные, глаголы и т. д., они все представлены в одной большой последовательной семантической сети.

В текущем исследовании используется OEHT-lite и терминология области информационной безопасности для изучения задачи обогащения таксономии

общей онтологии или графа знаний специфической для предметной области терминологией.

В качестве предметной области была выбрана область информационной безопасности, представленная корпусом из 500 тысяч документов. На основании текстового корпуса и таксономий из OEHT и OEHT-lite был построен набор данных OENTCyber следующим образом:

1. Были выбраны все однословные текстовые входы концептов из OEHT таким образом, чтобы они встречались в полной версии, но отсутствовали в OEHT-lite,
2. Из этого списка слов были оставлены только те, для которых гиперонимы присутствуют в сокращенной версии OEHT-lite,
3. Полученный список был также отфильтрован на основании наличия в частотном списке, построенном на корпусе статей по информационной безопасности с частотностью $>= 50$.

В итоге получился набор данных из 4372 слов, которые включают в себя как специфичные для области понятия, так и более общие, но которые еще не включены в OEHT-lite.

## 2.4 Эксперименты

Во всех проведенных экспериментах использовалась единая постановка задачи. Для существующей таксономии и набора новых слов требуется выдать для каждого нового слова список из 10 предсказаний в порядке убывания вероятности.

### 2.4.1 Меры оценки

Задача пополнения таксономии рассматривается как задача ранжирования, в которой правильные ответы должны быть в верхней части списка кандидатов. Поэтому для оценки используется традиционная мера для задач ранжирования - мера средней точности (Mean Average Precision, MAP).

$$MAP = \frac{1}{N}\sum_{i=1}^{N} AP_i;$$
$$AP_i = \frac{1}{M}\sum_{i}^{n} prec_i \times I[y_i = 1], \tag{2.9}$$

где $N$ - количество слов в наборе данных, $n$ и $M$ - количество предсказанных и правильных значений соответственно, $prec_i$ - доля правильных значений в предсказаниях в диапазоне от 1 до $i$, $y_i$ - метка $i$-го ответа в ранжированном списке предсказания, а $I$ - функция индикатор.

Для учета гиперонимов второго порядка используется модифицированная мера MAP. Список гиперонимов золотого стандарта преобразуется в список связанных компонент. Каждая из этих компонент включает гиперонимы (как первого, так и второго порядка), которые образуют связанный компонент в графе таксономии. Чтобы получить наилучшее качество, модель должна угадать гипероним для каждого компонента связности [25].

Также в некоторых экспериментах использовалась другая традиционная мера - среднеобратный ранг (Mean Reciprocal Rank, MRR).

$$MRR = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{rank_i}, \tag{2.10}$$

где $N$ - количество слов в наборе данных, $rank_i$ - позиция первого правильного предсказания.

### 2.4.2 Эксперименты на наборе данных RUSSE'2020

В рамках экспериментов на наборе данных RUSSE'2020 ставилась цель извлечь максимум информации из предоставленного организаторами набора текстовых данных News2017. На данном наборе данных использовался **комбинированный подход** к предсказанию гиперонимов. По этой причине использовались только обученные векторные представления на предоставленном корпусе. Результаты экспериментов можно видеть в Таблице 5 для существительных и в Таблице 6 для глаголов. Конфигурация под названием **base** относится к базовому подходу, ранжирующему по $base\_score$, конфигурация **base-ne** относится к базовой конфигурации, но со специальной обработкой именованных сущностей при построении векторных представлений. Для глаголов

подобная обработка не производилась, потому что глаголы редко выступают как часть именованной сущности.

Таблица 5 — Результаты на существительных

| | public | | private | |
|---|---|---|---|---|
| | MAP | MRR | MAP | MRR |
| base | 0.446 | 0.478 | 0.440 | 0.474 |
| base-ne | 0.449 | 0.482 | 0.444 | 0.478 |
| base-ne, co-hyponym patterns | 0.473 | 0.509 | 0.467 | 0.502 |
| base-ne, hypernym patterns | 0.467 | 0.505 | 0.456 | 0.494 |
| base-ne, both patterns | 0.482 | 0.522 | 0.481 | 0.520 |
| base-ne, bert | 0.471 | 0.507 | 0.455 | 0.490 |
| base-ne, bert, co-hyponym patterns | 0.494 | 0.533 | 0.475 | 0.509 |
| base-ne, bert, hypernym patterns | 0.484 | 0.524 | 0.470 | 0.508 |
| base-ne, bert, both patterns | **0.509** | **0.550** | **0.493** | **0.531** |

Таблица 6 — Результаты на глаголах

| | public | | private | |
|---|---|---|---|---|
| | MAP | MRR | MAP | MRR |
| base | 0.254 | 0.295 | 0.255 | 0.291 |
| base, co-hyponym patterns | 0.285 | 0.327 | 0.246 | 0.286 |
| base, hypernym patterns | 0.260 | 0.303 | 0.254 | 0.290 |
| base, both patterns | 0.288 | 0.332 | 0.244 | 0.284 |
| base, bert | 0.277 | 0.322 | **0.265** | **0.305** |
| base, bert, co-hyponym patterns | 0.314 | 0.358 | 0.259 | 0.302 |
| base, bert, hypernym patterns | 0.288 | 0.334 | 0.264 | 0.304 |
| base, bert, both patterns | **0.324** | **0.371** | 0.254 | 0.296 |

## Анализ результатов

Из результатов следуют следующие выводы:

- Исключение контекстов именованных сущностей немного улучшает результаты,
- Во всех случаях, за исключением private глаголов, использование шаблонов значительно улучшает результаты,
- Использование BERT описанным способом всегда улучшает результаты.

Описанный подход позволил получить четвертый результат на соревновании для существительных без каких-либо дополнительных словарей или дополнительных наборов данных, которые использовались в первых трех подходах победителей (согласно предоставленным описаниям), а отличие результатов от третьего места находятся в пределах погрешности.

Хотя предложенный подход и показал хорошие результаты, у него присутствует несколько проблем:

- Шаблоны необходимо подготовить вручную, а сопоставления насчитать заранее. При переходе на другие данные и языки, этот шаг может быть трудновыполним,
- Использование модели BERT доказало свою эффективность, но вычислительные сложности алгоритма возрастают в несколько раз, что затрудняет процесс улучшения алгоритма и подбора параметров,
- Итоговая формула для ранжирования подбиралась вручную на public данных, подобный подход может привести к неявному переобучению, и адаптация подхода на другие языки и наборы данных будет затруднена.

Исходя из описанных минусов, в дальнейших исследованиях BERT и шаблоны не использовались, а ранжирование по формуле было изменено на использование модели машинного обучения.

### 2.4.3 Эксперименты на наборе данных Diachronic wordnets

На наборе данных Diachronic wordnets использовался **комбинированный подход на основе мета-векторных представлений слов** к пополнению таксономии. Качество подхода оценивалось с использованием различных векторных представлений: fastText[1], word2vec[2], GloVe[3]. Также были исследованы разные подходы к построению мета-векторных представлений: конкатенация, SVD поверх конкатенации, CAEME, AAEME. Косинусное расстояние использовалось в качестве функции потерь для подходов AEME. Проверялись варианты и комбинации между MSE, KL дивергенцией и косинусным расстоянием, и последний вариант показал себя наилучшим образом для данной задачи, в рамках предварительных экспериментов.

Особенностью наборов данных на английском языке является наличие значительного количества многословных выражений. Для получения векторных представлений в таких случаях, были выполнены следующие процедуры:

- Для модели FastText, вектора, которые отсутствовали в словаре модели, были получены естественным образом, путем вычисления векторов самой моделью,
- Для моделей Word2Vec и GloVe, вектора рассчитывались путем усреднения векторов максимальных префиксов для составляющих слов многословных выражений. Также было установлено ограничение на минимальную длину префиксного слова, которая составляет 4 символа,
- Для подходов с мета-векторными представлениями, если в какой-либо исходной модели нет вектора для слова, соответствующий исходный вектор был инициализирован нулями.

Для обучения модели логистической регрессии был создан тренировочный набор данных на основании некоторой базовой версии таксономии. В случае с русским языком использовался RuWordNet1.0, в случае с английским языком WordNet1.6. В качестве тренировочных данных рассматривались понятия в таксономии, которые не имеют гипонимов, и из них выбиралось одно случайное слово или многословное выражение. Из всего множества подобных слов был

[1]Common Crawl английская и русская версии из https://fastText.cc/docs/en/crawl-vectors.html
[2]Araneum для русского и Gigaword для английского языка из http://vectors.nlpl.eu/repository/
[3]Common Crawl 840b токенов из https://nlp.stanford.edu/projects/GloVe/

выбран некоторый небольшой процент случайным образом, который и использовался в качестве обучающего множества. Размеры тренировочных данных представлены в Таблице 7.

Таблица 7 — Размеры тренировочных данных

| Язык | Часть речи | |
|---|---|---|
| | Сущ. | Глаг. |
| Английский | 2931 | 1532 |
| Русский | 2990 | 814 |

Во время обучения модели, из пополняемой таксономии удалялись все концепты, которые содержат тренировочные данные, таким образом воспроизводя ситуацию, когда необходимо добавить новое слово в таксономию, в которой отсутствует добавляемое понятие.

### Подбор параметров для функции потерь триплетов

В модификации AEME алгоритмов с помощью функции потерь триплетов используется ряд параметров, которые необходимо подобрать. Так как полный перебор параметров не представляется возможным из-за вычислительной сложности эксперимента, перебор происходил следующим образом:

1. На первом этапе фиксировался параметр $\alpha$, и происходил жадный перебор по сетке для параметров count (значения 5 и 10) и margin (значения от 0.0 до 0.3 с шагом 0.1), и функции потерь (MSE или косинусная). Целью был выбор функции потерь и параметра count,
2. На втором этапе фиксировалась выбранная функция потерь и параметр count, и происходил подбор параметров $\alpha$ и margin схожим образом.

Эксперименты проводились на английском наборе данных 1.7-3.0 для существительных с использованием признаков из Викисловаря. Само обучение автокодировщика производилось с использованием написанной (автором диссертации) программы на языке python с пакетом для обучения нейронных сетей pytorch. Обучение производилось в течение 20 эпох со скоростью обучения 2e-4, используя оптимизатор Adam. Результаты первого эксперимента представлены в Таблице 8.

Таблица 8 — Эксперименты по count и типу функции потерь

| | MSE | | cosine | |
|---|---|---|---|---|
| margin | count=5 | count=10 | count=5 | count=10 |
| 0.0 | 0.387 | 0.389 | 0.385 | 0.383 |
| 0.1 | 0.389 | 0.385 | 0.386 | 0.389 |
| 0.2 | 0.385 | 0.381 | 0.379 | 0.380 |
| 0.3 | 0.388 | 0.385 | 0.374 | 0.370 |
| Сред. | **0.387** | 0.384 | 0.380 | 0.380 |

Результаты показывают, что MSE в среднем показывает себя лучше, чем косинусное расстояние. Также большой разницы между count равным 5 и 10 не замечено, но вариант с count=5 показал себя чуть лучше. Результаты эксперимента по подбору $\alpha$ и margin представлены в Таблице 9.

Таблица 9 — Эксперименты по margin и $\alpha$ для потерь триплетов

| | margin | | | |
|---|---|---|---|---|
| $\alpha$ | 0.0 | 0.1 | 0.2 | Сред. |
| 0.5 | 0.385 | 0.387 | 0.384 | 0.387 |
| 0.1 | 0.387 | 0.391 | 0.387 | 0.388 |
| 0.02 | 0.388 | 0.389 | 0.387 | 0.388 |
| 0.01 | 0.390 | 0.391 | 0.388 | 0.389 |
| 0.005 | 0.391 | 0.394 | 0.389 | **0.391** |
| 0.002 | 0.387 | 0.389 | 0.381 | 0.385 |
| Сред. | 0.388 | **0.390** | 0.386 | |

По итогам экспериментов, был выбран следующий оптимальный набор параметров, который и использовался в основных экспериментах: функция расстояния для функции потерь триплетов - MSE, $\alpha = 0.005$, margin = 0.1, count = 5. Для случая с CAEME использовалось значение $\alpha = 0.01$.

## Мета-векторные представления в задаче пополнения таксономии

Результаты основных экспериментов приведены в Таблице 10 и Таблице 11. Видно, что предсказание гиперонимов для глаголов намного хуже, чем

для существительных на всех наборах данных. SVD, применяемый к конкатенации исходных векторных моделей, всегда улучшает результаты по сравнению с конкатенацией. SVD дает лучшие результаты по сравнению с исходными векторными моделями на большинстве наборов данных, за исключением английских глаголов. SVD обеспечивает лучшее качество предсказания гиперонимов среди всех рассмотренных мета-векторных представлений для глаголов на русскоязычных наборах данных.

Таблица 10 — MAP для методов обогащения таксономии для наборов данных на английском языке

| Метод | Существительные | | | Глаголы | | | Размер |
|---|---|---|---|---|---|---|---|
| | 1.6-3.0 | 1.7-3.0 | 2.0-3.0 | 1.6-3.0 | 1.7-3.0 | 2.0-3.0 | вектора |
| fastText | 0.300 | 0.346 | 0.396 | **0.290** | 0.224 | **0.280** | 300 |
| word2vec | 0.226 | 0.242 | 0.265 | 0.091 | 0.114 | 0.150 | 300 |
| GloVe | 0.261 | 0.290 | 0.326 | 0.182 | 0.145 | 0.175 | 300 |
| concat | 0.308 | 0.344 | 0.387 | 0.273 | 0.206 | 0.247 | 900 |
| SVD | 0.308 | 0.358 | 0.406 | 0.286 | 0.222 | 0.271 | 600 |
| CAEME | 0.309 | 0.347 | 0.395 | 0.252 | 0.189 | 0.260 | 900 |
| CAEME triplet loss | 0.322 | 0.367 | **0.416** | 0.287 | 0.218 | 0.270 | 900 |
| AAEME | 0.318 | 0.354 | 0.401 | 0.283 | 0.218 | 0.254 | 600 |
| AAEME triplet loss | **0.333** | **0.373** | **0.416** | 0.**289** | **0.227** | 0.274 | 600 |
| Предыдущие результаты: | | | | | | | |
| fastText [27] | – | – | 0.339 | – | – | 0.213 | – |
| Poincaré embeddings [78; 80] | 0.059 | 0.066 | 0.101 | 0.126 | 0.066 | 0.109 | – |
| node2vec [43; 80] | 0.194 | 0.219 | 0.155 | 0.151 | 0.109 | 0.147 | – |
| GCN autoencoder [48; 80] | 0.157 | 0.175 | 0.168 | 0.109 | 0.094 | 0.117 | – |

Автокодировщик AAEME сопоставим с автокодировщиком CAEME на русскоязычных и лучше на англоязычных наборах данных. Среди мета-векторных представлений, основанных на автокодировщиках, лучшие результаты предсказания гиперонимов дает автокодировщик AAEME с функцией потерь триплетов. Этот подход улучшает результаты предсказания гиперонимов почти во всех случаях по сравнению с исходными векторами fastText, за исключением английских глаголов. Автокодировщики с функцией потерь триплетов в большинстве случаев достигают лучших результатов, чем соответствующие автокодировщики без дополнительной функцией потерь.

По сравнению с другими подходами, основанными только на различных векторных представлениях, применяемые в работе методы дали лучшие резуль-

Таблица 11 — MAP для методов обогащения таксономии для наборов данных на русском языке

| Метод | Существительные | | Глаголы | | Размер |
|---|---|---|---|---|---|
| | non-restricted | restricted | non-restricted | restricted | вектора |
| fastText | 0.416 | 0.537 | 0.318 | 0.418 | 300 |
| word2vec | 0.276 | 0.526 | 0.231 | 0.272 | 600 |
| concat | 0.401 | 0.563 | 0.337 | 0.423 | 900 |
| SVD | 0.435 | 0.579 | **0.382** | 0.442 | 600 |
| CAEME | 0.468 | 0.579 | 0.352 | 0.433 | 900 |
| CAEME triplet loss | 0.470 | **0.580** | 0.350 | 0.420 | 900 |
| AAEME | 0.466 | 0.577 | 0.350 | 0.432 | 600 |
| AAEME triplet loss | **0.474** | **0.581** | 0.375 | 0.439 | 600 |
| Предыдущие результаты: | | | | | |
| RUSSE Top-1 for verbs: [76] | 0.288 | 0.418 | 0.340 | **0.448** | – |
| Poincaré embeddings [78; 80] | 0.143 | 0.252 | 0.105 | 0.140 | – |
| node2vec [43; 80] | 0.266 | 0.366 | 0.168 | 0.252 | – |
| GCN autoencoder [48; 80] | 0.183 | 0.261 | 0.095 | 0.141 | – |

таты на всех наборах данных, за исключением глаголов для русского языка, где результаты немного хуже.

### Улучшение результатов работы алгоритма с помощью Викисловаря

В Таблицах 12 и 13 показаны результаты с добавлением признаков на основе Викисловара. Добавление Викисловаря улучшило результаты по всем наборам данных. По сравнению с предыдущими подходами, использующими Викисловарь, были получены лучшие результаты на наборе данных Диахронических ворднетов.

### Использование графовых векторных представлений

Другим направлением экспериментов была проверка подхода с комбинированием векторных представлений слов с графовыми векторными представ-

Таблица 12 — MAP для методов обогащения таксономии для наборов данных на английском языке с признаками из Викисловаря

| Метод | Существительные | | | Глаголы | | |
|---|---|---|---|---|---|---|
| | 1.6-3.0 | 1.7-3.0 | 2.0-3.0 | 1.6-3.0 | 1.7-3.0 | 2.0-3.0 |
| fastText | 0.319 | 0.373 | 0.424 | **0.296** | 0.231 | **0.288** |
| word2vec | 0.236 | 0.259 | 0.288 | 0.090 | 0.12 | 0.149 |
| GloVe | 0.277 | 0.315 | 0.357 | 0.189 | 0.153 | 0.190 |
| concat | 0.326 | 0.370 | 0.425 | 0.275 | 0.213 | 0.259 |
| SVD | 0.324 | 0.380 | 0.437 | 0.289 | 0.228 | 0.276 |
| CAEME | 0.327 | 0.373 | 0.430 | 0.262 | 0.205 | 0.264 |
| CAEME triplet loss | 0.339 | 0.39 | 0.44 | 0.292 | 0.225 | 0.277 |
| AAEME | 0.334 | 0.379 | 0.432 | 0.286 | 0.225 | 0.267 |
| AAEME triplet loss | **0.345** | **0.394** | **0.445** | 0.289 | **0.239** | 0.272 |
| Предыдущие результаты: | | | | | | |
| fastText [27] | 0.337 | 0.380 | 0.344 | 0.267 | 0.200 | 0.237 |
| fastText + node2vec [80] | 0.313 | 0.380 | 0.340 | 0.259 | 0.195 | 0.200 |
| fastText+node2vec + Poincaré [80] | 0.311 | 0.350 | 0.300 | 0.251 | 0.177 | 0.248 |

лениями. Для этого были взяты обученные в работе [80] графовые векторные представления. Эксперимент состоял из нескольких этапов: а) оценка качества графовых векторных представлений отдельно, и б) оценка качества комбинации графовых векторных представлений и векторных представлений слов. В качестве основного алгоритма использовался описанный ранее подход, но с поиском по концептам, а не по словам. Дополнительные признаки из Викисловаря также использовались.

Комбинирование производилось с использованием алгоритма построения мета-векторных представлений, но к обычным векторным моделям добавлялись графовые векторные представления. Исследовались следующие графовые подходы: node2vec, TADW, GCN, GraphSAGE . Результаты представлены в Таблицах 14, 15, 16, 17.

### Анализ результатов

Проведенная серия экспериментов показала, что:

Таблица 13 — MAP для методов обогащения таксономии для наборов данных на русском языке с признаками из Викисловаря

| method | nouns | | verbs | |
|---|---|---|---|---|
| | non-restricted | restricted | non-restricted | restricted |
| fastText | 0.436 | 0.579 | 0.366 | 0.445 |
| word2vec | 0.292 | 0.574 | 0.258 | 0.295 |
| concat | 0.421 | 0.598 | 0.388 | 0.474 |
| SVD | 0.452 | **0.612** | 0.423 | 0.458 |
| CAEME | 0.485 | **0.614** | 0.400 | 0.472 |
| CAEME triplet loss | 0.486 | 0.607 | 0.402 | 0.463 |
| AAEME | 0.484 | **0.611** | 0.403 | **0.486** |
| AAEME triplet loss | **0.490** | **0.611** | **0.427** | 0.471 |
| Предыдущие результаты: | | | | |
| RUSSE Top-1 for nouns [25; 27] | 0.393 | 0.552 | 0.293 | 0.436 |
| fastText+wiki [27] | 0.413 | 0.551 | 0.297 | 0.389 |
| fastText + node2vec + Poincaré+wiki [80] | 0.414 | 0.560 | 0.306 | 0.391 |

– При достаточно крупной таксономии предложенный метод способен предсказывать правильные гиперонимы для новых слов в среднем в пределах трех первых позиций в ранжированном списке кандидатов,
– Для русскоязычного набора данных правильные предсказания расположены в среднем среди двух лучших кандидатов,
– Разработанный подход превосходит другие методы, которые оценивались на данном наборе данных,
– Использование мета-векторных представлений повышает качество работы подхода,
– Использование дополнительной функции потерь триплетов повышает качество работы подхода,
– Комбинирование обычных векторных представлений вместе с графовыми может привести к улучшению работы подхода при условии, если качество графовых векторных представлений достаточно высокое.

Чтобы проанализировать качество полученных предсказаний, были вычислены доли слов, имеющих хотя бы один правильный гипероним в N первых позициях списка кандидатов, см. Таблицу 18. Видно, что текущие результаты не позволяют использовать полностью автоматическое обогащение таксономии, потому что качество предсказания правильных ответов на первой позиции все

еще довольно низкое. Но предсказанные результаты могут значительно облегчить работу лексикографов или инженеров.

Также были проанализированы типы ошибок, которые допускает разработанная модель. Для этого рассматривались слова, для которых не было обнаружено правильного гиперонима в топ-10 списке предсказаний. Обнаружены следующие случаи:

- Предсказанные гиперонимы соответствуют смыслам, упущенным в таксономии. Например, слово *вечерница* описывается в RuWordNet только в смысле «ученица вечернего образования», но также это слово может означать «летучая мышь» (ночная летучая мышь) или «цветущее растение». Прогнозируемые гиперонимы включают концепты, соответствующий концепту цветов в 3-й позиции списка кандидатов,
- Во многих случаях прогнозы семантически близки, но неверны. Например, для слова «гугл» (Google) правильными ответами являются концепты «поисковая машина» и «глобальная поисковая система». Предсказанные гиперонимы: «компьютерная программа», «интернет-технология», «интернет-сайт», «IT-технология» и т.д. Для слова «датацентр», который представляет собой специализированное здание, прогноз на первая позиция — это «компьютер»,
- Также есть множество примеров, когда предсказываются слишком общие гиперонимы,
- В некоторых случаях предсказания очень далеки от разумных ответов, и их трудно объяснить.

### 2.4.4 Эксперименты на наборе данных OENTCyber

С целью изучения качества работы подхода в случае конкретной предметной области были проведены эксперименты на наборе данных OENTCyber. В качестве подхода использовался **комбинированный подход на основе мета-векторных представлений слов** с поиском кандидатов по словам, но, помимо использованных ранее двух "внешних" векторных моделей word2vec и fasttext, дополнительно были обучены две векторные модели (word2vec и fasttext соответственно) на корпусе текстов по информационной безопасности

(далее ”внутренние”). Параметры обучения были следующими: окно = 3, размер вектора = 300, количество эпох = 10, Skip-Gram. Помимо этого, в данном эксперименте не использовались признаки из Викисловаря, так как предварительные эксперименты показали, что на данной узкой предметной области, они не дают никакого эффекта в итоговое качество.

Исследовались возможности работы обученных моделей на предметной области, и также возможность совмещения их с более ”мощными” моделями, так как 500 тысяч текстов является небольшим числом, по сравнению с тем количеством документов, на которых обучались внешние модели word2vec и fasttext.

В случае использования подходов AEME для всех четырех векторных моделей было необходимо определить индивидуальные вклады каждой векторной модели при расчете функции потерь. Экспериментально были получены следующие веса: вес 1.0 для внутренних моделей, обученных на корпусе, вес 5.0 для внешней модели word2vec и вес 2.0 для внешней модели fasttext.

Ставилась серия из 3 экспериментов:

– Использование только внешних векторных моделей,
– Использование только внутренних векторных моделей,
– Комбинирование внешних и внутренних векторных моделей.

Результаты можно видеть в Таблицах 19, 20, 21.

### Анализ результатов

Полученные результаты позволяют сделать следующие выводы:

– Векторные модели, обученные на небольшой коллекции документов из предметной области, дают весьма низкие результаты, по сравнению с более крупными моделям, которые обучались на корпусах на порядки больших по размеру,
– Методы по построению мета-векторных представлений повышают качество работы подхода,
– Комбинирование слабых предметных векторных моделей с внешними векторными моделями не работает, если использовать такие простые подходы как конкатенацию и SVD, так как их качество сильно уступает,

– Итоговое качество на основе векторных моделей может быть улучшено за счет более слабых моделей, обученных на корпусе по информационной безопасности, методами AEME (также и с функцией потерь триплетов), если правильно подобрать способ комбинирования векторных моделей, путем задания весов.

## 2.5 Выводы

В главе исследовалась задача пополнения таксономии графа знаний новыми понятиями. Такая процедура необходима для поддержки подобных ресурсов и адаптации их на новые предметные области. Было предложено два подхода:

– Комбинированный подход на основе шаблонов и векторных представлений слов,
– Комбинированный подход на основе мета-векторных представлений слов.

Подходы были исследованы на нескольких наборах данных, таких как RUSSE'2020, Diachronic wordnets, а также на новом наборе данных OENTCyber. Методы построения мета-векторных представлений включали в себя конкатенацию, SVD поверх конкатенации, два варианта автокодировщиков (AAEME и CAEME). Также была предложена модификация алгоритмов AEME с целью внедрить явные знания и структуру существующей таксономии в получаемые векторные представления, путем использования дополнительной функции потерь (функция потерь триплетов).

Было показано, что использование мета-векторных представлений улучшает качество работы подхода практически для всех рассмотренных наборов данных, кроме английских глаголов. SVD всегда улучшает результаты по сравнению с конкатенацией. Мета-вектора на основе автокодировщика в большинстве случаев дают наилучшие результаты. Использование дополнительной функции потерь также улучшает результаты работы. Использование дополнительных признаков из Викисловаря помогает достичь лучших результатов для наборов данных из общей предметной области, но в случае узкой предметной области, подобный подход не дает прироста качества.

Также было показано, что адаптация на более узкую предметную область информационной безопасности представляет из себя более сложную задачу. Внешние словари перестают улучшать качество, также и общее качество оказалось ниже, чем для русскоязычных наборов данных для общей предметной области. Был предложен способ комбинирования векторных моделей, обученных на текстах из предметной области с внешними векторными моделями, который обучались на существенно больших объемах данных. Данных подход показал прирост качества.

Таблица 14 — MAP для методов обогащения таксономии для наборов данных на английском языке для существительных. Цифры, выделенные **жирным шрифтом**, показывают лучшую модель в категории, подчеркнутые числа обозначают лучший результат среди всех моделей

| Метод | Существительные | | |
|---|---|---|---|
| | 1.6-3.0 | 1.7-3.0 | 2.0-3.0 |
| Векторные модели слов | | | |
| fastText | **0.314±0.001** | **0.373±0.003** | **0.418±0.004** |
| word2vec | 0.244±0.001 | 0.271±0.003 | 0.298±0.004 |
| GloVe | 0.283±0.001 | 0.329±0.003 | 0.377±0.004 |
| Мета-векторные представления на векторных моделях слов | | | |
| concat (*words*) | **0.335±0.001** | 0.386±0.003 | 0.386±0.003 |
| SVD (*words*) | 0.333±0.001 | **0.399±0.003** | **0.456±0.004** |
| CAEME (*words*) | 0.321±0.001 | 0.386±0.003 | 0.448±0.005 |
| AAEME (*words*) | 0.322±0.001 | 0.384±0.003 | 0.453±0.004 |
| CAEME triplet loss (*words*) | 0.332±0.001 | 0.394±0.003 | 0.451±0.004 |
| AAEME triplet loss (*words*) | **0.335±0.001** | 0.391±0.003 | 0.453±0.004 |
| Графовые векторные модели | | | |
| GCN [119] | 0.175±0.001 | 0.249±0.002 | 0.267±0.002 |
| GraphSAGE [120] | 0.214±0.001 | 0.282±0.002 | 0.224±0.003 |
| TADW [44] (on fastText) | **0.350±0.001** | **0.392±0.002** | **0.435±0.004** |
| node2vec [121] (top-5 fastText associates) | 0.270±0.001 | 0.312±0.002 | 0.341±0.004 |
| Мета-векторный представления на комбинации векторных моделей слов и графовых векторных моделях | | | |
| SVD (*words* + node2vec) | 0.343±0.001 | 0.383±0.003 | 0.434±0.005 |
| CAEME (*words* + node2vec) | 0.335±0.001 | 0.379±0.003 | 0.426±0.0042 |
| AAEME (*words* + node2vec) | 0.350±0.001 | 0.394±0.003 | 0.446±0.004 |
| SVD (*words* + TADW) | 0.355±0.001 | 0.414±0.003 | 0.472±0.004 |
| CAEME (*words* + TADW) | 0.350±0.001 | 0.404±0.003 | 0.458±0.004 |
| AAEME (*words* + TADW) | **0.367±0.001** | **0.418±0.002** | **0.480±0.004** |
| SVD (*words* + GCN) | 0.323±0.001 | 0.385±0.003 | 0.443±0.004 |
| CAEME (*words* + GCN) | 0.331±0.001 | 0.395±0.003 | 0.457±0.004 |
| AAEME (*words* + GCN) | 0.331±0.001 | 0.392±0.003 | 0.456±0.004 |
| SVD (*words* + GraphSAGE) | 0.338±0.001 | 0.401±0.003 | 0.464±0.004 |
| CAEME (*words* + GraphSAGE) | 0.323±0.001 | 0.382±0.003 | 0.435±0.004 |
| AAEME (*words* + GraphSAGE) | 0.343±0.001 | 0.406±0.003 | 0.468±0.004 |
| Предыдущие подходы | | | |
| WBSR (Top-1 RUSSE'2020 для существительных) [25] | 0.333±0.002 | **0.393±0.003** | **0.436±0.003** |
| WBSR, без использования поисковых технологий [25] | 0.251±0.001 | 0.309±0.003 | 0.344±0.004 |
| hypo2path rev [122] | 0.264±0.001 | 0.283±0.003 | 0.238±0.007 |
| hypo2path [122] | 0.252±0.002 | 0.261±0.002 | 0.208±0.006 |
| hypo2path transformer [122] | 0.218±0.002 | 0.229±0.002 | 0.057±0.002 |
| *TaxoExpan* [123] | *0.004±0.000* | *0.003±0.000* | *0.054±0.002* |

Таблица 15 — MAP для методов обогащения таксономии для наборов данных на английском языке для глаголов. Цифры, выделенные **жирным шрифтом**, показывают лучшую модель в категории, подчеркнутые числа обозначают лучший результат среди всех моделей

| Метод | Глаголы | | |
|---|---|---|---|
| | 1.6-3.0 | 1.7-3.0 | 2.0-3.0 |
| Векторные модели слов | | | |
| fastText | **0.286±0.007** | **0.218±0.008** | **0.254±0.012** |
| word2vec | 0.099±0.005 | 0.118±0.008 | 0.141±0.010 |
| GloVe | 0.182±0.007 | 0.159±0.008 | 0.203±0.011 |
| Мета-векторные представления на векторных моделях слов | | | |
| concat (*words*) | 0.270±0.007 | 0.194±0.009 | 0.226±0.011 |
| SVD (*words*) | 0.277±0.007 | 0.209±0.010 | 0.264±0.012 |
| CAEME (*words*) | 0.278±0.007 | 0.205±0.008 | 0.266±0.015 |
| AAEME (*words*) | 0.271±0.007 | **0.218±0.008** | **0.273±0.012** |
| CAEME triplet loss (*words*) | 0.273±0.007 | 0.205±0.007 | 0.256±0.013 |
| AAEME triplet loss (*words*) | **0.280±0.008** | 0.212±0.007 | 0.262±0.014 |
| Графовые векторные модели | | | |
| GCN [119] | 0.162±0.006 | 0.113±0.005 | 0.149±0.010 |
| GraphSAGE [120] | 0.127±0.004 | 0.114±0.004 | 0.090±0.008 |
| TADW [44] (on fastText) | **0.268±0.007** | **0.201±0.007** | **0.217±0.010** |
| node2vec [121] (top-5 fastText associates) | 0.175±0.006 | 0.128±0.007 | 0.118±0.012 |
| Мета-векторный представления на комбинации векторных моделей слов и графовых векторных моделях | | | |
| SVD (*words* + node2vec) | 0.272±0.006 | 0.194±0.009 | 0.239±0.011 |
| CAEME (*words* + node2vec) | 0.242±0.005 | 0.184±0.009 | 0.221±0.012 |
| AAEME (*words* + node2vec) | 0.252±0.007 | 0.184±0.008 | 0.208±0.012 |
| SVD (*words* + TADW) | **0.288±0.007** | 0.222±0.009 | **0.280±0.013** |
| CAEME (*words* + TADW) | 0.267±0.007 | 0.212±0.007 | 0.247±0.011 |
| AAEME (*words* + TADW) | 0.283±0.007 | **0.227±0.007** | 0.260±0.012 |
| SVD (*words* + GCN) | 0.260±0.005 | 0.209±0.009 | 0.249±0.011 |
| CAEME (*words* + GCN) | 0.251±0.006 | 0.207±0.009 | 0.235±0.012 |
| AAEME (*words* + GCN) | 0.243±0.006 | 0.200±0.008 | 0.228±0.012 |
| SVD (*words* + GraphSAGE) | 0.239±0.006 | 0.194±0.009 | 0.221±0.011 |
| CAEME (*words* + GraphSAGE) | 0.200±0.006 | 0.170±0.007 | 0.202±0.01 |
| AAEME (*words* + GraphSAGE) | 0.238±0.007 | 0.178±0.008 | 0.209±0.011 |
| Предыдущие подходы | | | |
| WBSR (Top-1 RUSSE'2020 для существительных) [25] | **0.252±0.006** | **0.206±0.011** | **0.252±0.013** |
| WBSR, без использования поисковых технологий [25] | 0.231±0.006 | 0.180±0.008 | 0.222±0.009 |
| hypo2path rev [122] | 0.173±0.005 | 0.104±0.008 | 0.118±0.009 |
| hypo2path [122] | 0.162±0.005 | 0.093±0.006 | 0.067±0.008 |
| hypo2path transformer [122] | 0.140±0.003 | 0.120±0.006 | 0.100±0.008 |
| *TaxoExpan* [123] | *0.001±0.000* | *0.000±0.000* | *0.000±0.000* |

Таблица 16 — MAP для методов обогащения таксономии для наборов данных на русском языке для существительных. Цифры, выделенные **жирным шрифтом**, показывают лучшую модель в категории, подчеркнутые числа обозначают лучший результат среди всех моделей

| Метод | Существительные | |
|---|---|---|
| | Не ограниченный набор | Ограниченный набор |
| Векторные модели слов | | |
| fastText | **0.419±0.001** | **0.572±0.005** |
| word2vec | 0.296±0.002 | 0.569±0.005 |
| Мета-векторные представления на векторных моделях слов | | |
| concat (*words*) | 0.422±0.001 | 0.589±0.005 |
| SVD (*words*) | 0.461±0.001 | **0.600±0.005** |
| CAEME (*words*) | 0.400±0.001 | 0.561±0.005 |
| AAEME (*words*) | 0.456±0.001 | 0.582±0.005 |
| CAEME triplet loss (*words*) | 0.449±0.001 | 0.581±0.005 |
| AAEME triplet loss (*words*) | **0.474±0.001** | 0.593±0.006 |
| Графовые векторные модели | | |
| GCN [48] | 0.183±0.001 | 0.306±0.005 |
| GraphSAGE [120] | 0.176±0.001 | 0.348±0.005 |
| TADW [44] | **0.417±0.001** | **0.562±0.005** |
| node2vec [121] (top-5 fastText associates) | 0.343±0.002 | 0.477±0.005 |
| Мета-векторный представления на комбинации векторных моделей слов и графовых векторных моделях | | |
| SVD (*words* + node2vec) | 0.367±0.001 | 0.521±0.005 |
| CAEME (*words* + node2vec) | 0.370±0.001 | 0.533±0.005 |
| AAEME (*words* + node2vec) | 0.373±0.001 | 0.529±0.005 |
| SVD (*words* + TADW) | **0.469±0.001** | **0.604±0.006** |
| CAEME (*words* + TADW) | 0.429±0.001 | 0.571±0.005 |
| AAEME (*words* + TADW) | 0.461±0.001 | 0.584±0.005 |
| SVD (*words* + GCN) | 0.395±0.001 | 0.554±0.005 |
| CAEME (*words* + GCN) | 0.389±0.001 | 0.544±0.005 |
| AAEME (*words* + GCN) | 0.386±0.001 | 0.545±0.006 |
| SVD (*words* + GraphSAGE) | 0.410±0.001 | 0.603±0.005 |
| CAEME (*words* + GraphSAGE) | 0.321±0.001 | 0.541±0.005 |
| AAEME (*words* + GraphSAGE) | 0.409±0.001 | 0.577±0.006 |
| Предыдущие подходы | | |
| WBSR (Top-1 RUSSE'2020 для существительных) [25] | **0.393±0.002** | **0.552±0.005** |
| WBSR, без использования поисковых технологий [25] | 0.369±0.002 | 0.497±0.005 |
| Top-1 RUSSE'2020 для глаголов: [76] | 0.288±0.001 | 0.418±0.006 |
| hypo2path [122] | 0.061±0.000 | 0.097±0.002 |
| hypo2path rev [122] | 0.246±0.001 | 0.342±0.006 |
| hypo2path rev transformer [122] | 0.234±0.001 | 0.331±0.004 |
| *TaxoExpan* [123] | *0.007±0.000* | *0.006±0.001* |

Таблица 17 — MAP для методов обогащения таксономии для наборов данных на русском языке для глаголов. Цифры, выделенные **жирным шрифтом**, показывают лучшую модель в категории, подчеркнутые числа обозначают лучший результат среди всех моделей

| Метод | Глаголы | |
|---|---|---|
| | Не ограниченный набор | Ограниченный набор |
| Векторные модели слов | | |
| fastText | **0.337±0.003** | **0.428±0.007** |
| word2vec | 0.250±0.003 | 0.284±0.011 |
| Мета-векторные представления на векторных моделях слов | | |
| concat (*words*) | 0.351±0.004 | 0.426±0.009 |
| SVD (*words*) | **0.426±0.005** | **0.475±0.010** |
| CAEME (*words*) | 0.342±0.003 | 0.416±0.008 |
| AAEME (*words*) | 0.368±0.004 | 0.442±0.009 |
| CAEME triplet loss (*words*) | 0.374±0.003 | 0.427±0.010 |
| AAEME triplet loss (*words*) | 0.399±0.004 | 0.449±0.010 |
| Графовые векторные модели | | |
| GCN [48] | 0.220±0.003 | 0.287±0.009 |
| GraphSAGE [120] | 0.181±0.003 | 0.226±0.008 |
| TADW [44] | **0.328±0.003** | **0.423±0.008** |
| node2vec [121] (top-5 fastText associates) | 0.226±0.003 | 0.322±0.010 |
| Мета-векторный представления на комбинации векторных моделей слов и графовых векторных моделях | | |
| SVD (*words* + node2vec) | 0.252±0.003 | 0.351±0.010 |
| CAEME (*words* + node2vec) | 0.267±0.003 | 0.362±0.010 |
| AAEME (*words* + node2vec) | 0.272±0.003 | 0.358±0.010 |
| SVD (*words* + TADW) | **0.394±0.005** | **0.455±0.010** |
| CAEME (*words* + TADW) | 0.349±0.003 | 0.437±0.009 |
| AAEME (*words* + TADW) | 0.362±0.004 | 0.439±0.009 |
| SVD (*words* + GCN) | 0.291±0.004 | 0.356±0.009 |
| CAEME (*words* + GCN) | 0.302±0.003 | 0.381±0.008 |
| AAEME (*words* + GCN) | 0.295±0.004 | 0.365±0.008 |
| SVD (*words* + GraphSAGE) | 0.336±0.004 | 0.426±0.009 |
| CAEME (*words* + GraphSAGE) | 0.266±0.004 | 0.345±0.007 |
| AAEME (*words* + GraphSAGE) | 0.323±0.004 | 0.419±0.009 |
| Предыдущие подходы | | |
| WBSR (Top-1 RUSSE'2020 для существительных) [25] | 0.293±0.004 | 0.428±0.010 |
| WBSR, без использования поисковых технологий [25] | 0.267±0.004 | 0.387±0.009 |
| Top-1 RUSSE'2020 для глаголов: [76] | **0.341±0.004** | **0.452±0.012** |
| hypo2path [122] | 0.137±0.003 | 0.174±0.009 |
| hypo2path rev [122] | 0.151±0.003 | 0.194±0.008 |
| hypo2path rev transformer [122] | 0.152±0.003 | 0.201±0.008 |
| *TaxoExpan* [123] | *0.009±0.001* | *0.008±0.002* |

Таблица 18 — Первый правильный ответ среди первых N позиций. Результаты приведены для существительных, основанных с AAEME triplet loss с Викисловарем

| | Английский % | | | Русский % | |
|---|---|---|---|---|---|
| | 1.6-3.0 | 1.7-3.0 | 2.0-3.0 | non-restricted | restricted |
| Top-1 | 28 | 32 | 38 | 42 | 52 |
| Top-3 | 41 | 45 | 51 | 57 | 75 |
| Top-5 | 47 | 52 | 56 | 64 | 81 |
| Top-10 | 54 | 59 | 63 | 70 | 87 |

Таблица 19 — Расширение OEHT-lite: внешние векторные модели

| Метод | MAP | MRR |
|---|---|---|
| fastText | 0.362 | 0.407 |
| word2vec | 0.375 | 0.421 |
| concat | 0.397 | 0.446 |
| SVD | 0.400 | 0.447 |
| CAEME | 0.391 | 0.439 |
| CAEME triplet | 0.398 | 0.448 |
| AAEME | 0.404 | 0.453 |
| AAEME triplet | **0.412** | **0.464** |

Таблица 20 — Расширение OEHT-lite: внутренние векторные модели

| Метод | MAP | MRR |
|---|---|---|
| fastText | 0.277 | 0.317 |
| word2vec | 0.277 | 0.316 |
| concat | 0.287 | 0.327 |
| SVD | 0.283 | 0.324 |
| CAEME | 0.286 | 0.325 |
| CAEME triplet | **0.298** | **0.339** |
| AAEME | 0.280 | 0.319 |
| AAEME triplet | 0.295 | 0.335 |

Таблица 21 — Расширение OEHT-lite: комбинация внутренних и внешних моделей

| Метод | MAP | MRR |
|---|---|---|
| fastText внутр. | 0.277 | 0.317 |
| word2vec внутр. | 0.277 | 0.316 |
| fastText внешн. | 0.362 | 0.407 |
| word2vec внешн. | 0.375 | 0.421 |
| concat | 0.386 | 0.434 |
| SVD | 0.387 | 0.433 |
| CAEME | 0.362 | 0.407 |
| CAEME triplet | 0.408 | 0.456 |
| AAEME | 0.414 | 0.463 |
| AAEME triplet | **0.427** | **0.479** |

# Глава 3. Методы пополнения графов знаний именованными сущностями в конкретной предметной области

При работе над данной главой диссертации использованы следующие публикации автора, в которых, согласно Положению о присуждении ученых степеней в МГУ, отражены основные результаты, положения и выводы исследования:

1. Tikhomirov M. [и др.]. Pretraining and augmentation in named entity recognition task for cybersecurity domain in Russian // Computational Linguistics and Intellectual Technologies. — 2020. — С. 724—735,
2. Tikhomirov M. [и др.]. Using bert and augmentation in named entity recognition for cybersecurity domain // International Conference on Applications of Natural Language to Information Systems. — Springer. 2020. — С. 16—24,
3. Tikhomirov M., Loukachevitch N., Dobrov B. Recognizing Named Entities in Specific Domain // Lobachevskii Journal of Mathematics. — 2020. — Т. 41, No 8. — С. 1591—1602.

## 3.1 Задача извлечения именованных сущностей

Графы знаний, помимо концептов, связанных отношением класс-подкласс, также содержат концепты, связанные отношением экземпляр-класс. Такие объекты часто представляют собой именованные сущности, например, WannaCry — это экземпляр для класса компьютерный вирус. Особенностями именованных сущностей являются а) они очень разнообразны, все время появляются новые, что приводит к тому, что в статических векторных моделях могут отсутствовать их векторные представления, и б) в тексте они могут выражаться как отдельными словами, так и многословными выражениями, и даже регистр таких слов имеет значение, из-за чего нормализация слов может привести к существенной потери информации.

По описанным причинам, для того чтобы пополнять графы знаний новыми объектами, которые связаны отношением экземпляр-класс, необходимо

использовать контекстуализированные векторные представления и методы извлечения именованных сущностей.

## 3.2 Постановка задачи

В данной работе рассматривается задача пополнения графов знаний именованными сущностями в области информационной безопасности. Имеется:

- Размеченный набор текстовых данных, в котором каждое слово относится к одной из $K$ категорий (или же к пустой категории),
- Не размеченный набор текстовых данных из предметной области, но существенно большего размера,
- Существующая таксономия графа знаний,
- Также могут быть использованы дополнительные размеченные коллекции из общей предметной области.

Требуется реализовать такой алгоритм $A$, который будет выделять из текстов из конкретной предметной области сущности описанных категорий, после чего их можно будет добавить в существующий граф знаний.

## 3.3 Используемый подход и модели

Описанная задача имеет следующие сложности:

- Размеченных данных по некоторым сущностям может быть недостаточно из-за сложности разметки и особенностей текстов,
- Векторные представления специфичных слов могут быть некорректными из-за отсутствия подобных слов и смыслов в корпусах общей области.

В работе исследовались следующие методы для борьбы с описанными сложностями: а) дополнение тренировочных данных данными из общей области, б) дополнение тренировочных данных за счет большой не размеченной коллекции и в) настройка контекстуализированной векторной модели BERT на предметную область.

### 3.3.1 Методы дополнения данных

Методы дополнения данных для обработки естественного языка в основном обсуждаются в рамках таких задач, как машинный перевод и автоматическая классификация текстов. Самый простой метод дополнения - заменить исходные слова их синонимами из построенных вручную таксономий (например, WordNet [16]) или словами, близкими к исходным словам в соответствии с векторной моделью, обученной на большом текстовом наборе [124]. В [125] утверждалось, что синонимы могут не вписываться в контекст, поэтому заменяемые слова должны быть теми, которые наиболее вероятны согласно языковой модели.

Авторы [126] использовали четыре простых метода дополнения данных для задач классификации: замена слов их синонимами (WordNet), случайная вставка слов, случайное удаление слов и случайное изменение порядка слов. Этот метод был применен к пяти наборам данных для задачи классификации текстов. Оценка качества была представлена для моделей на основе нейронных сетей RNN и CNN. Было достигнуто среднее улучшение 0.8% по F-мере. Исследование показало, что все четыре операции способствовали полученному улучшению.

В диссертации было исследовано: а) использование дополнительного размеченного набора данных для извлечения именованных сущностей в общей области, и б) метод автоматического построения псевдоразметки для задачи извлечения именованных сущностей в области информационной безопасности.

#### Порождение псевдоразметки на основе дескрипторов

Был предложен новый подход к дополнению данных для задачи извлечения именованных сущностей. Основная идея метода заключается в следующем: в большинстве контекстов, где упоминается дескриптор объекта, возможны разные варианты упоминания именованной сущности. Дескриптор — это некоторый сигнал/маркер, выраженный в виде слова или многословного выражения. Вариантами упоминания могут быть: 1) дескриптор, за которым следует имя

или 2) только имя. Первый из указанных выше вариантов упоминания сущностей зависит от языка и правил грамматики. Следовательно, можно расширить коллекцию, добавляя имена после дескрипторов или заменяя дескрипторы именами.

Для процедуры дополнения данных осуществляется отбор предложений с соответствующими дескрипторами в коллекции не аннотированных текстов по информационной безопасности. Сам список дескрипторов формировался вручную с использованием онтологии ОЕНТ. Поиск осуществляется с помощью списка дескрипторов и определенных ограничений на предложения. Применяемые ограничения были следующие:

– В предложении не должно быть явных именованных сущностей (слов, начинающихся с заглавной буквы),
– Требуется отсутствие прилагательных или других слов, согласованных с дескриптором и стоящих перед ним.

Если предложение соответствует ограничениям, то с равной вероятностью: (1) дескриптор заменяется именем, или (2) имя добавляется после дескриптора, или (3) предложение сохраняется как есть.

Именованные сущности для операции вставки или замены были получены следующим образом. Была взята большая коллекция текстов по информационной безопасность и из нее извлекались все имена, следующие за дескрипторами. Был создан частотный список извлеченных имен, и выбраны те имена, для которых частота была выше определенного порога. Затем были исключены имена, которые появились в аннотированной обучающей коллекции или принадлежали классам, отличным от целевого класса. Разработанный подход к построению псевдоразмеченных данных для задачи извлечения именованных сущностей изображен на Рис 3.1.

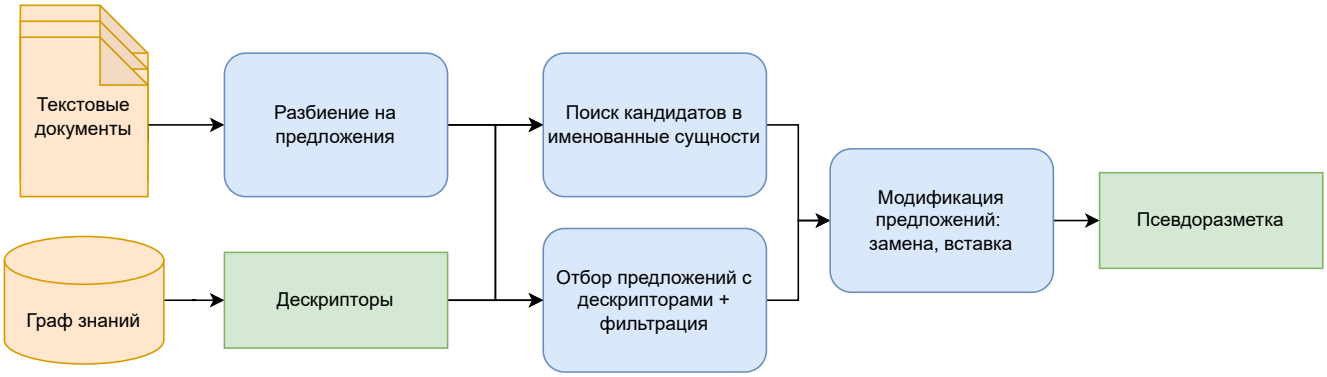


Рисунок 3.1 — Схема работы модуля порождения псевдоразметки для обучения

Объем псевдоразмеченных данных может быть неограниченного размера. Поэтому было исследовано, как объем дополненных данных влияет на качество на целевой задаче.

Таблицы 22 и 23 показывают примеры предлагаемой процедуры построения псевдоразметки. В первой паре предложений дескрипторы были заменены именами, во второй паре предложений имена были вставлены после дескрипторов ”хакер” и ”зловред”.

Таблица 22 — Примеры псевдоразметки для HACKER

| Исходное | Измененное |
|---|---|
| Замена | |
| Отсутствие уязвимостей на сайте и его готовность противостоять атакам **хакеров** - важный вопрос, который часто упорно игнорируется владельцами сайтов. | Отсутствие уязвимостей на сайте и его готовность противостоять атакам **Pwn2Own** - важный вопрос, который часто упорно игнорируется владельцами сайтов. |
| Вставка | |
| А количество установленных программных средств защиты от **хакеров** меньше - 71% от тех, кто установил межсетевой экран. | А количество установленных программных средств защиты от **хакеров Sandworm** меньше - 71% от тех, кто установил межсетевой экран. |

Таблица 23 — Примеры псевдоразметки для VIRUS

| Исходное | Измененное |
|---|---|
| Замена | |
| Почти 30% серьезно обеспокоены этой проблемой, еще 25% считают, что опасность **шпионского ПО** преувеличена, а более 15% вообще не считают этот тип угроз проблемой. | Почти 30% серьезно обеспокоены этой проблемой, еще 25% считают, что опасность **Remcos** преувеличена, а более 15% вообще не считают этот тип угроз проблемой. |
| Insertion | |
| Описанный выше **вредонос** уникален и может создать большие проблемы как для отдельного человека, так и для всей компании. | Описанный выше **вредонос Locker** уникален и может создать большие проблемы как для отдельного человека, так и для всей компании. |

### 3.3.2 Подход на основе контекстуализированной векторной модели BERT

В работе исследуется использование модели BERT [23] для задачи извлечения именованных сущностей в области информационной безопасности. Модель BERT получает последовательность токенов, полученных путем токенизации текста с использованием метода WordPiece [127], и генерирует последовательность контекстуализированных векторных представлений, соответствующих этим токенам.

Одной из основных особенностей модели BERT является механизм внимания [65]. Его основная идея заключается в том, что, при формировании представлений токенов на каждом уровне, учитываются представления всех токенов с предыдущего уровня. Еще одна важная особенность BERT - способ обучения этой модели. Обучение делится на два этапа: предварительное обучение (pretraining) и тонкая настройка (fine-tuning) [60]. На этапе предварительного обучения модель обучается на задаче маскированного языкового моделирования. Для этого фрагменты текста рассматриваются как входные данные модели, в которой некоторые из токенов замаскированы специальным токеном <MASK>, и задача состоит в том, чтобы предсказать каждый из этих замаскированных токенов. Этап предварительного обучения требует достаточно много времени и ресурсов. На этапе тонкой настройки слои для конкретных задач строятся поверх слоев модели BERT, а слои BERT инициализируются предварительно обученными весами. После чего происходит дальнейшее обучение для соответствующей задачи. Этап тонкой настройки обычно занимает относительно мало времени.

Из-за того, что предварительное обучение является дорогостоящей процедурой, использование предварительно обученных весов более эффективно. Важными вопросами являются, какие предварительно обученные веса использовать, и на каких данных эти веса были обучены. Первая опубликованная модель многоязычной модели BERT (multilingual-bert-base) была обучена на многоязычных текстовых коллекциях, включая русскоязычные данные [23]. Позже было показано, что модель BERT, специализированная для конкретного языка, работает лучше, чем многоязычная, когда она обучалась на сопоставимом количестве данных [95].

По этой причине исследователи из DeepPavlov [96] обучили модель RuBERT на русской Википедии и новостном корпусе [95]. Для этого они:

– Инициализировали матрицу весов модели весами многоязычного BERT,
– Построили новый словарь токенов аналогичного размера, более подходящий для обработки русскоязычных текстов, тем самым уменьшив среднюю длину токенизированных последовательностей в 1.6 раза, что критично для производительности модели,
– Инициализировали векторные представления новых токенов с использованием векторов из многоязычной модели,
– Обучили полученную модель с новым словарем на русскоязычных данных.

Полученная модель показала лучшие результаты на нескольких задачах обработки естественного языка по сравнению с многоязычной версией [95].

В диссертации было оценено качество моделей архитектуры BERT в задаче извлечения именованных сущностей в области информационной безопасности со следующими предварительно обученными весами:

– RuBERT, модель, обученная на русскоязычных данных [95],
– RuCyBERT, модель, полученная путем дообучения RuBERT на текстах по информационной безопасности.

Все описанные модели имеют одинаковую архитектуру: трансформер-кодировщик [64] с 12 трансформер блоками, блоками самовнимания и размером скрытых слоев H = 768. Модели обучались для задачи извлечения именованных сущностей 6 эпох, с размером батча B = 16, со скоростью обучения 2e-5 и максимальной длиной последовательности T = 128. При формировании входных данных для модели только первый токен слова получал реальную метку слова (тип сущности), остальные лексемы получают специальную метку X. На шаге предсказания метка всего слова выбиралась исходя из предсказанной метки первого токена. Схема работы нейронной модели BERT для извлечения именованных сущностей представлена на Рис. 3.2.

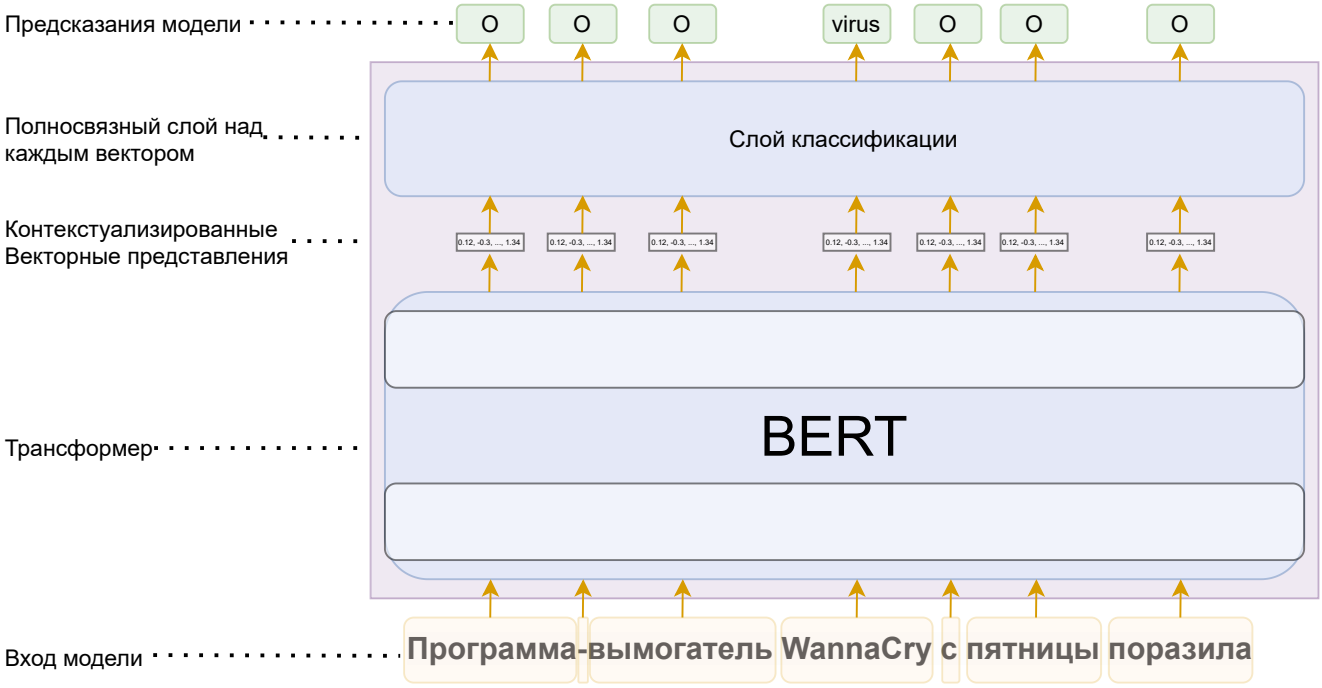


Рисунок 3.2 — Схема работы BERT для извлечения именованных сущностей

## 3.4 Описание данных для задачи извлечения именованных сущностей в области информационной безопасности

В данной работе использовалась обновленная версия Sec_col [102] в качестве обучающего набора данных для задачи извлечения именованных сущностей. Корпус содержит 861 неструктурированных текстов (более 400 тыс. токенов), которые представляют собой сообщения и комментарии, извлеченные из нескольких источников по информационной безопасности.

Набор сущностей корпуса включает как четыре основных типа: PER (для персон, исключая хакеров), ORG (для организаций, исключая группы хакеров), LOC и EVENT, так и пять типов, зависящих от предметной области, таких как PROGRAM (для компьютерных программ, исключая вредоносное ПО), DEVICE (для различных электронных устройств), TECH (для технологий с собственными именами), VIRUS (вредоносное ПО и уязвимости) и HACKER (для отдельных хакеров и групп хакеров). Корпус был предварительно аннотирован автоматически, затем было выполнено многопроходное ручное аннотирование. Принципы аннотации подробно описаны в [102]. Частотные характеристики сущностей в корпусе представлены в Таблице 24.

Таблица 24 — Распределение сущностей в Sec_col

| Тип сущности | Описание | Количество |
|---|---|---|
| ORG | организации (для организаций, исключая группы хакеров) | 3791 |
| PROGRAM | программы (программные продукты и их составляющие: коды, процедуры) | 3497 |
| TECH | технологии (именованные методы и подходы) | 2962 |
| LOC | локации (географические локации) | 1376 |
| PER | персоны (имена людей, которые не являются хакерами) | 1015 |
| DEVICE | устройства (различные электронные и компьютерные устройства) | 539 |
| VIRUS | вирусы (вредоносное ПО и уязвимости) | 480 |
| EVENT | события | 301 |
| HACKER | хакеры (отдельные хакеры и хакерские группы) | 60 |

Таблица 25 — Распределение сущностей в Collection3

| Тип сущности | Описание | Количество |
|---|---|---|
| PER | персоны (имена людей) | 10623 |
| ORG | организации | 5683 |
| LOC | локации (географические локации) | 7244 |

Согласно таблице, большинство типов сущностей, относящихся к предметной области, имеют относительно небольшое количество аннотированных объектов. Например, сущность HACKER, сильно недопредставлена в наборе данных. Одна из причин этого может заключаться в том, что в момент атаки хакеры неизвестны, поэтому их имена не упоминаются. Другой важный тип меток, VIRUS, представлен лучше, чем HACKER, но его частота все же ниже, чем у других типов.

В качестве набора данных из общей предметной области использовался набор данных Collection3 [82]. Collection3 содержит 1000 новостных текстов, помеченных тремя типами именованных сущностей: людьми, организациями и локациями. В Таблице 25 представлено количество размеченных объектов в соответствии с различными типами сущностей. Из таблицы видно, что набор данных содержит значительное количество общих именованных сущностей, и, следовательно, важно понять, как использовать доступные общие данные для улучшения результатов извлечения именованных сущностей в определенной области.

### 3.4.1 Дополнение Sec_col тренировочными данными (порождение псевдоразметки)

Важными типами именованных сущностей в области информационной безопасности являются имена вирусов и хакеров (включая группы хакеров). Однако коллекция Sec_col включает в себя довольно небольшое количество имен хакеров. Это может быть связано с тем, что имена многих хакеров и групп хакеров неизвестны, поэтому многие тексты, связанные с информационной безопасностью, включают только не именованные дескрипторы (например, хакер, группа хакеров, сообщество хакеров и тд).

Порождение псевдоразметки реализовано для всех типов именованных сущностей, размеченных в Sec_col и отсутствующих в Collection3, включая вредоносное ПО (метка VIRUS), хакеров (метка HACKER), технологии (метка TECH), компьютерные программы (метка PROGRAM), событие (метка EVENT) и электронные устройства (метка DEVICE). В Таблице 26 показаны количество и примеры дескрипторов, используемых в процедуре.

Таблица 26 — Общая информация о дескрипторах

| Тип сущности | Количество | Примеры |
|---|---|---|
| HACKER | 6 | сообщество хакеров<br>организация хакеров |
| VIRUS | 28 | вредоносный файл<br>вредонос |
| DEVICE | 52 | аппаратная платформа<br>жесткий диск |
| PROGRAM | 65 | системная программа<br>текстовый редактор |
| TECH | 19 | кэш<br>алгоритм |
| EVENT | 20 | выставка<br>конференция |

В работе предлагается двухэтапный подход к обучению модели BERT для задачи извлечения именованных сущностей при использовании псевдоразметки:

– На первом этапе модель обучается только на дополнительных данных (на псевдоразметке и на дополнительных размеченных данных из общей предметной области),

– На втором этапе модель дообучается уже на целевых тренировочных данных.

## 3.5 Эксперименты

В рамках экспериментов было произведено сравнение нескольких вариантов модели архитектуры BERT для задачи извлечения именованных сущностей для области информационной безопасности: RuBERT и RuCyBERT. Также были проведены эксперименты с методами порождения псевдоразметки.

Обучение RuCyBERT в работе происходило схожим с RuBERT образом, но без создания нового словаря. Для этого была запущена процедура предварительного обучения на 500 тысячах текстов по информационной безопасности с инициализацией всех весов из RuBERT. Обучение длилось 70 эпох с размером батча 32 и максимальной длиной последовательности 128. Для обучения использовался суперкомпьютер DGX-2 с 16 графическими процессорами NVIDIA Tesla V100, поэтому общий размер батча равен 512. Обучение заняло немногим более двух дней, что эквивалентно месяцу обучения на обычном ЭВМ с современным графическим процессором.

Поскольку модели, основанные на нейронных сетях, из-за случайной инициализации могут показывать несколько разные результаты от запуска к запуску, результаты в таблицах для всех моделей BERT представлены как среднее значение четырех запусков. Последняя строка таблиц указывает (F-macro std) стандартное отклонение результатов от среднего.

Схема проведенных экспериментов приведена на Рис. 3.3. Каждый эксперимент исследовался в двух вариантах: RuBERT и RuCyBERT для того, чтобы выяснить вклад в итоговое качество как методов порождения псевдоразметки, так и дообучения модели BERT на предметную область.

Первый эксперимент направлен на то, чтобы выявить, может ли добавление размеченных данных, но из общей области (в которой присутствует только 3 типа именованных сущностей: PER, LOC, ORG), улучшить качество извлечения на целевом наборе данных. В Таблице 27 отражены результаты для двух моделей BERT в двух вариантах: а) базовая постановка обучения модели только на Sec_col и б) с добавлением Collection3 к Sec_col. Как видно из результатов,

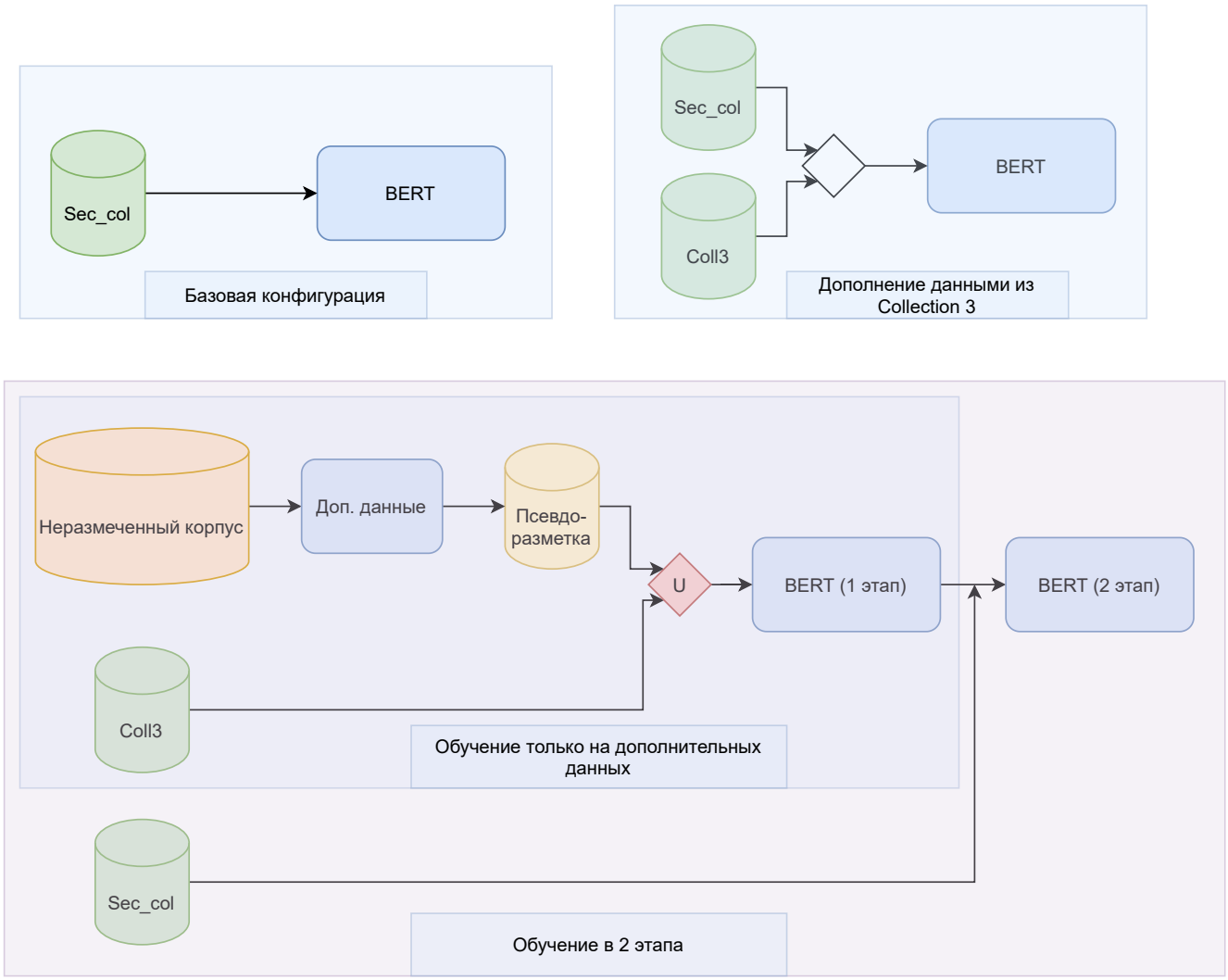


Рисунок 3.3 — Схема экспериментов по извлечению именованных сущностей

особенно по микро и макро метрикам, такая процедура не дает значимых улучшений. Показатели по одним сущностям улучшаются, по другим ухудшаются.

В Таблицах 28 и 29 показаны результаты экспериментов по обучению только на объединении псевдоразметки и Collection3. Целью эксперимента было выяснить, насколько получившееся псевдоразметка соответствует эталонной разметке в Sec_col.

В Таблице 30 отражен основной эксперимент. В нем сравниваются результаты базовой конфигурации и конфигурации, когда модель обучается в двухэтапном режиме.

Исходя из приведенных результатов, можно заключить, что на оцененных наборах данных:

– RuCyBERT всегда значительно лучше, чем RuBERT,
– Двухэтапный режим обучения с использованием псевдоразметки и Collection3 может улучшить конечные результаты для целевой задачи.

Таблица 27 — Качество базовых моделей

| | RuBERT | | RuCyBERT | |
|---|---|---|---|---|
| | seccoll | + coll3 | seccoll | + coll3 |
| DEVICE | 42.96 | **44.94** | 53.52 | **54.73** |
| EVENT | **66.19** | 64.78 | **68.82** | 68.60 |
| HACKER | **58.89** | 54.73 | **68.43** | 62.70 |
| LOC | **91.09** | 90.30 | **91.10** | 90.66 |
| ORG | **79.27** | 78.60 | **80.87** | 80.08 |
| PER | 83.85 | **85.51** | 86.38 | **87.77** |
| PROGRAM | **65.45** | 65.06 | 68.31 | **69.43** |
| TECH | 67.34 | **67.49** | **71.21** | 71.04 |
| VIRUS | 45.94 | **48.56** | 61.39 | **61.68** |
| F-micro | 71.79 | 71.79 | 75.21 | 75.15 |
| F-macro | 66.77 | 66.66 | **72.34** | 71.85 |
| F-macro std | **0.68** | 0.81 | 0.84 | 0.84 |

Таблица 28 — Качество RuBERT при обучении только на псевдоразметке и Collection3

| | доп. 200 | доп. 400 | доп. 800 | доп. 1600 |
|---|---|---|---|---|
| DEVICE | **14.65** | 13.35 | 12.34 | 13.76 |
| EVENT | 31.16 | 32.31 | 33.72 | **33.78** |
| HACKER | 14.73 | **14.77** | 14.15 | 13.88 |
| LOC | 89.18 | 89.27 | 89.30 | **89.46** |
| ORG | 68.93 | 68.84 | **68.98** | 67.75 |
| PER | 82.19 | **82.70** | 82.54 | 82.07 |
| PROGRAM | 25.07 | 23.61 | 23.73 | **26.05** |
| TECH | **13.10** | 12.52 | 12.65 | 12.09 |
| VIRUS | 25.80 | **29.07** | 26.29 | 27.11 |
| F-micro | **48.76** | 48.36 | 48.39 | 48.03 |
| F-macro | 40.17 | **40.71** | 40.30 | 40.66 |
| F-macro std | 1.39 | 0.25 | 0.30 | **0.22** |

Таблица 29 — Качество RuCyBERT при обучении только на псевдоразметке и Collection3

| | доп. 200 | доп. 400 | доп. 800 | доп. 1600 |
|---|---|---|---|---|
| DEVICE | 17.97 | **22.09** | 18.42 | 19.68 |
| EVENT | 36.54 | **38.20** | 37.70 | 36.15 |
| HACKER | 21.91 | 21.50 | **23.91** | 18.18 |
| LOC | 89.03 | 89.34 | **89.78** | 89.74 |
| ORG | **72.78** | 72.37 | 72.06 | 72.06 |
| PER | **85.88** | 85.15 | 85.04 | 85.03 |
| PROGRAM | 26.45 | 27.82 | 29.40 | **29.44** |
| TECH | 9.29 | 11.40 | **17.76** | 11.01 |
| VIRUS | 33.62 | **36.49** | 32.81 | 32.03 |
| F-micro | 52.36 | **52.59** | 52.46 | 51.59 |
| F-macro | 43.72 | **44.93** | 44.76 | 43.80 |
| F-macro std | 0.83 | **0.26** | 0.60 | 0.46 |

Таблица 30 — Качество при последовательном обучении модели

| | RuBERT | | | RuCyBERT | | |
|---|---|---|---|---|---|---|
| | база | доп. 400 | доп. 1600 | база | доп. 400 | доп. 1600 |
| DEVICE | 42.96 | **43.64** | 43.28 | 53.52 | **54.16** | 52.58 |
| EVENT | 66.19 | **67.35** | 65.69 | 68.82 | **72.34** | 70.39 |
| HACKER | 58.89 | **60.36** | 60.29 | **68.43** | 66.03 | 66.52 |
| LOC | 91.09 | **91.37** | 91.29 | 91.10 | 91.60 | **91.70** |
| ORG | 79.27 | **79.84** | 79.67 | 80.87 | 81.07 | **81.45** |
| PER | 83.85 | **85.59** | 85.55 | 86.38 | **88.31** | 88.05 |
| PROGRAM | 65.45 | **66.22** | 66.20 | 68.31 | 69.10 | **69.65** |
| TECH | 67.34 | 67.20 | 67.33 | 71.21 | 71.24 | 71.02 |
| VIRUS | 45.94 | 50.61 | **51.42** | 61.39 | 61.51 | **62.58** |
| F-micro | 71.79 | **72.51** | 72.48 | 75.21 | 75.50 | **75.58** |
| F-macro | 66.77 | **68.05** | 67.86 | 72.34 | **72.82** | 72.66 |
| F-macro std | 0.68 | 0.64 | **0.54** | 0.84 | 1.02 | **0.50** |

### 3.5.1 Оценка производительности

В данном разделе приводятся исследования по вычислительной производительности обучения модели в зависимости от размера батча (batch size) и

использования смешанной точности. Для всех экспериментов в данном исследовании использовался суперкомпьютер NVIDIA DGX-2 с 16 графическими процессорами NVIDIA Tesla V100. Одной из особенностей этих графических процессоров NVIDIA является более эффективное вычисление операций с плавающей запятой с 16 битами (float16) по сравнению с 32 битами (float32). Операции с более низкой точностью должны использоваться на графических процессорах NVIDIA, где это возможно без существенной потери качества.

Однако, некоторые матрицы все равно должны оставаться в float32 по вычислительным причинам, чтобы модель обучалась со сравнимым качеством. Смешанная точность означает возможность использования float16 для некоторых вычислений и float32 для остальных, где это необходимо. В экспериментах по вычислительной производительности RuCyBERT был выбран в качестве инициализации модели, а оценка проводилась на наборе данных Sec_coll с 4 разбиениями и 12 эпохами.

В Таблице 31 представлены результаты эксперимента. Для экспериментов со смешанной точностью использовалась библиотека apex, разработанная компанией NVIDIA, которая позволяет обучать нейронные сети через библиотеку pytorch используя смешанную точность. Apex предоставляет несколько вариантов использования смешанной точности (O1, O2, O3). O1 и O2 представляют собой реализацию смешанной точности разным образом, а O3 случай, когда все матрицы и вычисления проходят в float16 режиме. В данных исследованиях использовалась опция O2.

Таблица 31 — Характеристики вычислительного эксперимента в зависимости от размера батча и использования смешанной точности

| | fp32 | | | fp16 O2 | | |
|---|---|---|---|---|---|---|
| | F1 micro | время(с) | память | F1 micro | время(с) | память |
| размер батча 16 | 75.20 | 1742 | 6000 | 75.16 | 849 | 5500 |
| размер батча 32 | 75.15 | 1241 | 8400 | 75.16 | 525 | 6700 |
| размер батча 64 | 75.53 | 903 | 13000 | 74.54 | 340 | 8400 |

Как видно из результатов, использование смешанной точности ускоряет вычисления более чем в два раза без потери качества, а также снижает потребление памяти графического процессора. Увеличение размера батча ускоряет обучение, но результирующее качество размером батча 64 ниже, чем для 32

и 16. Вероятно причина в том, что с подобным размером батча было сделано недостаточно шагов оптимизации нейронной сети. На рисунке 3.4 показаны результаты во время процесса обучения для разных размеров батча (использовался fp16), где ось x - номер шага оптимизации. На рисунке 3.5 отображено тоже самое, но с относительным временем обучения на оси x.

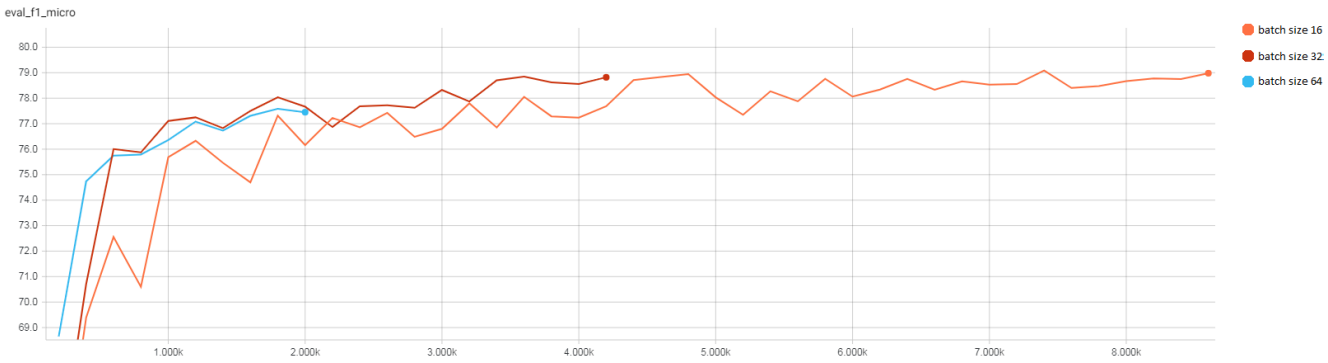


Рисунок 3.4 — Качество во время обучения, по оси X номер шага

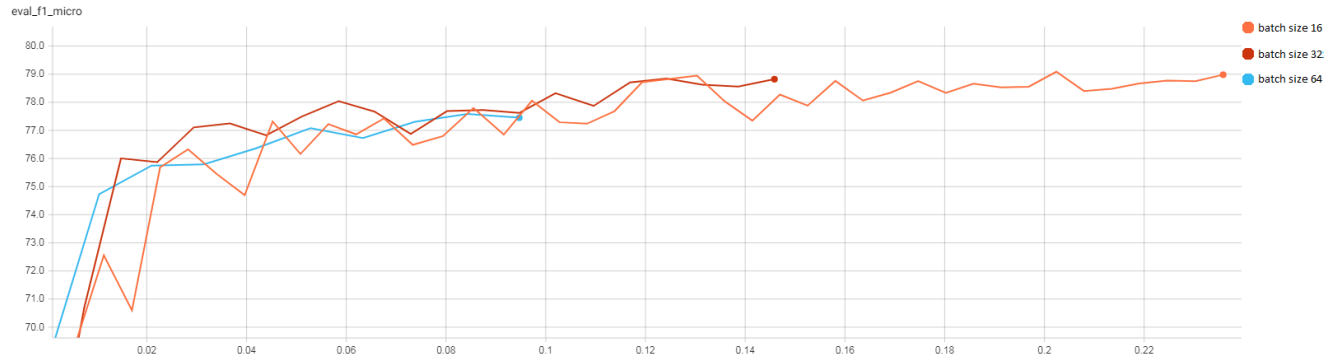


Рисунок 3.5 — Качество во время обучения, по оси X время обучения

Из графиков видно, что чем больше размер батча, тем стабильнее обучение (линия обучения более плавная). При больших размерах батча конкретные значения результата получается быстрее с точки зрения количества шагов, но с точки зрения времени ускорения не наблюдается.

Можно сделать следующий вывод: при работе с графическими процессорами подобного типа необходимо использование смешанной точности. Также необходимо выбрать правильный размер батча под задачу и настроить нужным образом количество эпох, чтобы процесс обучения сошелся. Чем больше размер батча, тем больше может понадобится эпох. В случае данной задачи, размер батча больше влияет на стабильность обучения, чем на скорость сходимости или конечное качество.

## 3.6 Выводы

В данной главе была исследована задача извлечения именованных сущностей из предметной области информационной безопасности для русского языка с целью пополнения графа знаний именованными сущностями.

Было показано, что настройка модели BERT на конкретную область значительно улучшает качество работы подхода на этой области.

Был предложен метод дополнения тренировочных данных (порождение псевдоразметки) для задачи извлечения именованных сущностей. Описанный подход представляет собой добавление имен после или вместо дескриптора определенного типа. Кроме того, предложенный подход может быть использован для автоматического создания предметно-ориентированного набора аннотированных данных для задачи извлечения именованных сущностей из общих наборов данных, таких как Collection3.

Также был предложен двухэтапный подход к обучению модели BERT для извлечения именованных сущностей с использованием псевдоразметки. Благодаря этому, было достигнуто улучшение качества работы модели.

Было показано, что эффективное использование возможностей графических процессоров, таких как вычисления со смешанной точностью и правильный выбор размера батча, необходимы и существенно влияют на процесс обучения. При увеличении размера батча также необходимо увеличивать количество эпох, чтобы достигалась сходимость модели. Увеличение размера батча увеличивает стабильность обучения, но в целом не влияет на итоговую скорость обучения.

# Глава 4. Программный комплекс автоматизированного пополнения графов знаний

При работе над данной главой диссертации использованы следующие публикации автора, в которых, согласно Положению о присуждении ученых степеней в МГУ, отражены основные результаты, положения и выводы исследования:

Тихомиров М. Разработка автоматизированной системы пополнения таксономии на текстах конкретной предметной области // Интеллектуальные системы. Теория и приложения. — 2021. — Т. 25, No 4. — С. 250—254.

## 4.1 Схема программного комплекса

Полученные в Главе 2 результаты не позволяют использовать реализованные методы для пополнения графов знаний в автоматическом режиме, но позволяют использовать их в автоматизированном режиме. Такое заключение было сделано после анализа позиций первых правильных ответов в предсказаниях, показанные в Таблице 18. Автоматизированный режим пополнения графа знаний — это способ пополнения графа знаний, с использованием человеческого труда. Человек (аннотатор), анализирует выдаваемые моделью результаты и на основании них производит разметку. Под разметкой подразумевается, что аннотатор принимает решение о том, какие предсказания правильные и фиксирует их в системе.

Также был проведен анализ того, в скольких шагах от полученных предсказаний содержатся правильные гиперонимы. Под шагом подразумевается переход к гипероними или гипониму для любого из анализируемых концептов из предсказания. Анализ проводился для результатов по набору данных OENTCyber и содержит в себе информацию о минимальных, средних и максимальных дистанциях. Например, если первая группа гиперонимов для слова доступна сразу из списка предсказаний модели, а вторая (в случае наличия двух) доступна за 4 шага, то тогда минимальная дистанция равна 0, средняя

равна 2, а максимальная 4. Усредненные результаты по набору данных представлены в Таблице 32.

Таблица 32 — Анализ дистанции до правильных гиперонимов

| | Мин. | Сред. | Макс. |
|---|---|---|---|
| 0 шаг | 71.7% | 61.5% | 61.5% |
| 0-1 шаг | 78.7% | 73.2% | 70.8% |
| 0-2 шаг | 83.4% | 80.4% | 77.8% |

Из таблицы можно сделать вывод, что хотя бы один корректный гипероним доступен практически 72% случаях сразу в списке из 10 предсказаний, и практически в 78% случаях правильный гипероним доступен всего лишь за 2 шага.

На основании проведенных исследований был реализован программный комплекс, использование которого облегчает задачу пополнения графа знаний для аннотаторов. Схематически программный комплекс представлен на Рис. 4.1. Построенная система позволяет:

– Получать ранжированный список предсказаний для введенного пользователем слова (сервис предсказаний),
– Обработать набор слов, сформировать для каждого слова ранжированный список концептов-кандидатов, и представить результат в удобном для разметки виде (сервис разметки),
– Обучать новые модели на основании существующих таксономий графов знаний в формате WordNet и векторных моделей слов (модуль обучения).

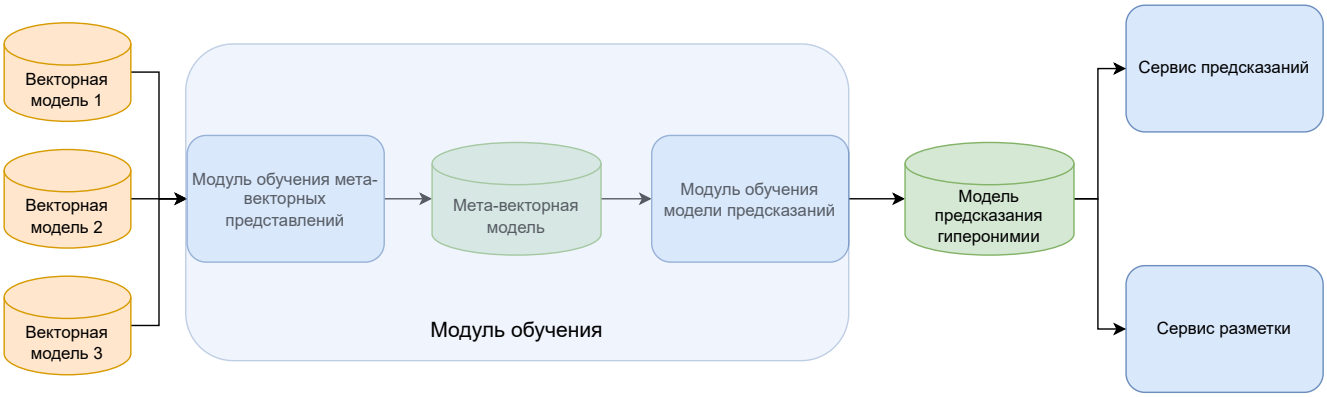


Рисунок 4.1 — Схема программной системы

Программная система реализована на языке Python версии 3.8 с использованием таких основных библиотек как: gensim, pytorch, sklearn, flask. Пользовательский интерфейс реализован с использованием HTML, css и javascript.

## 4.2 Сервис предсказаний

Через сервис предсказаний гиперонимии у пользователя есть возможность изучить предсказания модели в режиме ”онлайн” для любого слова или многословного выражения, которые присутствуют в словаре векторной модели. Для этого пользователь должен ввести интересующее его слово и количество предсказаний для отображения, затем нажать кнопку ”predict” в веб-сервисе. Система в течение 1-2 секунд обрабатывает запрос пользователя и выдает список предсказанных гиперонимов с весами, которые отражают уверенность модели в предсказании. Помимо этого, пользователю отображается граф, который представляет собой подмножество вершин графа знаний, которое включено в предсказания и отношения между вершинами. Первые 3 предсказания связываются с искомым словом ребрами. Анализ подобного графа позволяет быстро ознакомится с какими группами концептов связано целевое слово, что позволяет лучше понять его семантические характеристики.

На Рис. 4.2 представлен интерфейс пользователя для работы с сервисом на примере слова css3. Из графа видно, что данное понятие тесно связано с программированием и веб-разработкой, что является корректным.

## 4.3 Сервис разметки

Для более удобного взаимодействия с обученными моделями был реализован сервис разметки. Его цель в том, чтобы упростить работу аннотаторов при решении задачи пополнения графа знаний новой лексикой. Реализованный сервис позволяет навигацию по списку предсказаний, разметку каждого предсказания на один из нескольких классов, навигацию по ближайшим гиперонимам и гипонимам предсказаний.

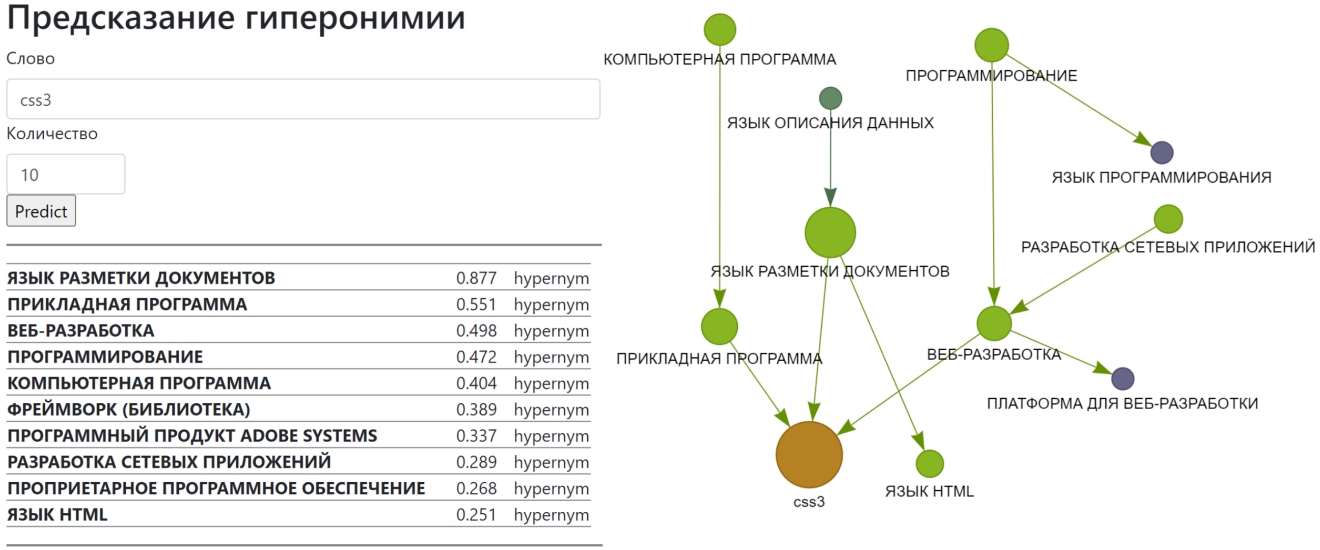


Рисунок 4.2 — Интерфейс пользователя для работы с сервисом предсказаний

Работа с системой происходит следующим образом:

1. Администратор системы запускает метод предсказания гиперонимов для интересующего набора слов,
2. Результат предсказания загружается в веб-сервис,
3. Аннотаторы получают доступ к системе, где для каждого слова есть набор из 10 предсказаний, которые необходимо разметить,
4. При необходимости в список предсказаний можно добавить связанное понятие и разметить его.

Каждое предсказание, которое отображается пользователю, также содержит инструменты отображения дополнительной информации о гиперонимах и гипонимах предсказанного концепта. Помимо этого, выводится информация о весе предсказания, и сам список упорядочен в соответствии с этим весом. От аннотаторов требуется: просмотреть список, пополнить его при необходимости близкими концептами и разметить, связан ли каждый концепт с целевым словом некоторым отношением.

## 4.4 Модуль обучения

Модуль обучения является реализацией разработанных алгоритмов, который позволяет:

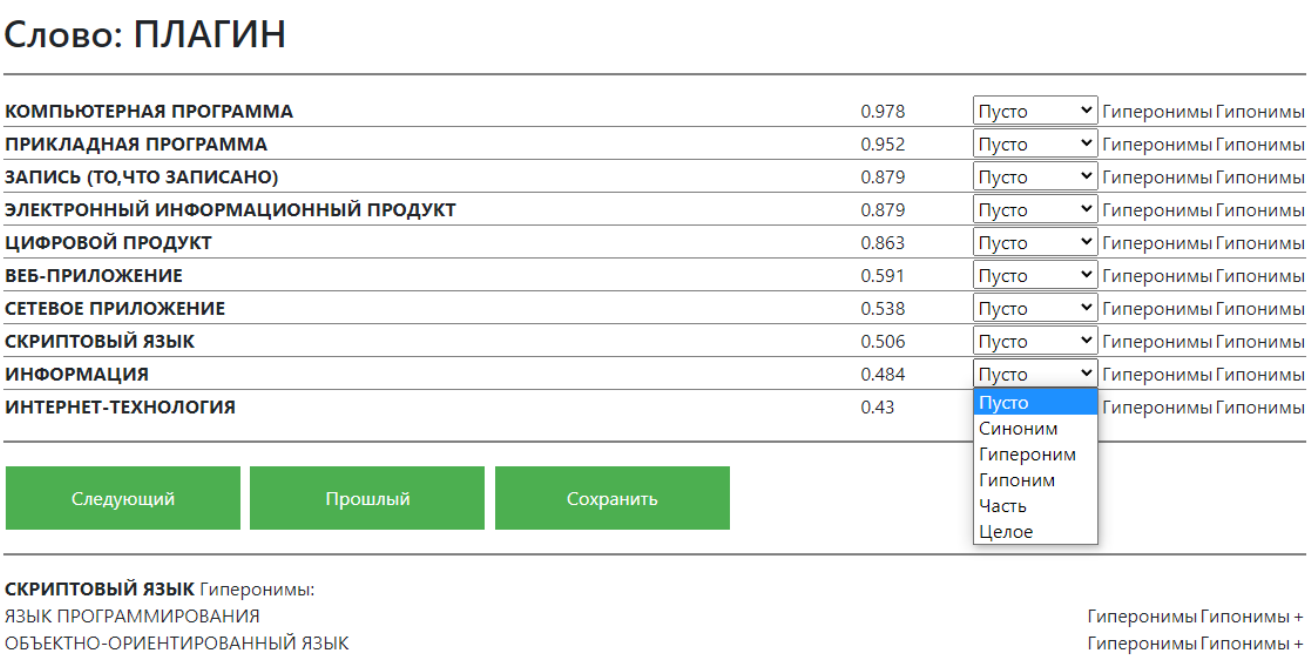


Рисунок 4.3 — Пример подготовленного списка предсказаний для разметки

– Проводить обучение мета-векторных представлений (конкатенация, SVD, AEME вариации),
– Проводить обучение модели предсказания гиперонимии на тройке: а) таксономия графа знаний б) векторная модель в) тренировочный набор,
– Сохранять модель предсказания в виде, необходимом для других модулей,
– Производить тестирование обученных моделей, в случае наличия тестовых размеченных данных.

# Заключение

Основные результаты работы заключаются в следующем. В работе исследованы методы пополнения графов знаний новыми понятиями и именованными сущностями.

Разработан и реализован метод пополнения таксономии графа знаний с использованием мета-векторных представлений. Исследована применимость разработанного метода на английском и русском языках, в общей области и конкретной предметной области информационной безопасности.

Разработан новый подход к задаче извлечения именованных сущностей в области информационной безопасности для русского языка с использованием псевдоразметки, двухэтапного обучения и специализированной языковой модели в области компьютерной безопасности RuCyBERT.

Реализована автоматизированная программная система для пополнения таксономии новыми словами.

## Список литературы


1. Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем. — Питер, 2000.
2. Зацман И. Концептуальный поиск и качество информации. — Федеральное государственное унитарное предприятие Академический научно-издательский, производственно-полиграфический и книгораспространительский центр Наука, 2003.
3. Hitzler P. A review of the semantic web field // Communications of the ACM. — 2021. — Т. 64, № 2. — С. 76—83.
4. Vrandečić D., Krötzsch M. Wikidata: a free collaborative knowledgebase // Communications of the ACM. — 2014. — Т. 57, № 10. — С. 78—85.
5. Bollacker K. [и др.]. Freebase: a collaboratively created graph database for structuring human knowledge // Proceedings of the 2008 ACM SIGMOD international conference on Management of data. — 2008. — С. 1247—1250.
6. Liu H., Singh P. ConceptNet—a practical commonsense reasoning tool-kit // BT technology journal. — 2004. — Т. 22, № 4. — С. 211—226.
7. Speer R., Lowry-Duda J. Conceptnet at semeval-2017 task 2: Extending word embeddings with multilingual relational knowledge // arXiv preprint arXiv:1704.03560. — 2017.
8. Paulheim H. Knowledge graph refinement: A survey of approaches and evaluation methods // Semantic web. — 2017. — Т. 8, № 3. — С. 489—508.
9. Dietz L., Kotov A., Meij E. Utilizing knowledge graphs for text-centric information retrieval // The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. — 2018. — С. 1387—1390.
10. Huang X. [и др.]. Knowledge graph embedding based question answering // Proceedings of the twelfth ACM international conference on web search and data mining. — 2019. — С. 105—113.
11. Ait-Mlouk A., Jiang L. KBot: a Knowledge graph based chatBot for natural language understanding over linked data // IEEE Access. — 2020. — Т. 8. — С. 149220—149230.

12. Zhou R. [и др.]. WCL-BBCD: A Contrastive Learning and Knowledge Graph Approach to Named Entity Recognition // arXiv preprint arXiv:2203.06925. — 2022.

13. Petasis G. [и др.]. Ontology population and enrichment: State of the art // Knowledge-driven multimedia information extraction and ontology evolution. — 2011. — С. 134—166.

14. Meijer K., Frasincar F., Hogenboom F. A semantic approach for extracting domain taxonomies from text // Decision Support Systems. — 2014. — Т. 62. — С. 78—93.

15. Roller S., Kiela D., Nickel M. Hearst Patterns Revisited: Automatic Hypernym Detection from Large Text Corpora // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). — 2018. — С. 358—363.

16. Fellbaum C. WordNet: An electronic lexical database and some of its applications. — 1998.

17. Hearst M. A. Automatic acquisition of hyponyms from large text corpora // Coling 1992 volume 2: The 15th international conference on computational linguistics. — 1992.

18. Levy O., Goldberg Y., Dagan I. Improving distributional similarity with lessons learned from word embeddings // Transactions of the Association for Computational Linguistics. — 2015. — Т. 3. — С. 211—225.

19. Bullinaria J. A., Levy J. P. Extracting semantic representations from word co-occurrence statistics: A computational study // Behavior research methods. — 2007. — Т. 39, № 3. — С. 510—526.

20. Turney P. D., Pantel P. From frequency to meaning: Vector space models of semantics // Journal of artificial intelligence research. — 2010. — Т. 37. — С. 141—188.

21. Mikolov T. [и др.]. Efficient estimation of word representations in vector space // arXiv preprint arXiv:1301.3781. — 2013.

22. Peters M. [и др.]. Deep Contextualized Word Representations // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). — Association for Computational Linguistics, 2018. — С. 2227—2237.

23. Devlin J. [и др.]. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). — Association for Computational Linguistics, 06.2019. — С. 4171—4186.

24. Jurgens D., Pilehvar M. T. SemEval-2016 Task 14: Semantic Taxonomy Enrichment // Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016). — Association for Computational Linguistics, 06.2016. — С. 1092—1102.

25. Nikishina I. [и др.]. RUSSE'2020: Findings of the First Taxonomy Enrichment Task for the Russian Language // Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue". — 2020.

26. Loukachevitch N. V. [и др.]. Creating Russian wordnet by conversion // Computational Linguistics and Intellectual Technologies: papers from the Annual conference "Dialogue. — 2016. — С. 405—415.

27. Nikishina I. [и др.]. Studying Taxonomy Enrichment on Diachronic WordNet Versions // Proceedings of the 28th International Conference on Computational Linguistics. — Barcelona, Spain : Association for Computational Linguistics, 12.2020.

28. Luo X. [и др.]. AliCoCo: Alibaba e-commerce cognitive concept net // Proceedings of the 2020 ACM SIGMOD international conference on management of data. — 2020. — С. 313—327.

29. Tikhomirov M., Loukachevitch N., Parkhomenko E. Combined approach to hypernym detection for thesaurus enrichment // Computational Linguistics and Intellectual Technologies. — 2020. — С. 736—746. — [Scopus: Impact Factor 0.427].

30. Tikhomirov M., Loukachevitch N. V. Domain-specific Taxonomy Enrichment based on Meta-Embeddings // CEUR Workshop Proceedings. Т. 3036. — 2021. — С. 285—298. — [Scopus: Impact Factor 0.551].

31. Tikhomirov M., Loukachevitch N. Meta-Embeddings in Taxonomy Enrichment Task // Computational Linguistics and Intellectual Technologies: papers from the Annual conference Dialogue. — 2021. — С. 681—691. — [Scopus: Impact Factor 0.427].

32. Tikhomirov M., Loukachevitch N., Dobrov B. Recognizing Named Entities in Specific Domain // Lobachevskii Journal of Mathematics. — 2020. — Т. 41, № 8. — С. 1591—1602. — [Scopus: Impact Factor 0.969].

33. Tikhomirov M. [и др.]. Using bert and augmentation in named entity recognition for cybersecurity domain // International Conference on Applications of Natural Language to Information Systems. — Springer. 2020. — С. 16—24. — [Scopus: Impact Factor 1.363].

34. Tikhomirov M. [и др.]. Pretraining and augmentation in named entity recognition task for cybersecurity domain in Russian // Computational Linguistics and Intellectual Technologies. — 2020. — С. 724—735. — [Scopus: Impact Factor 0.427].

35. Loukachevitch N., Tikhomirov M., Parkhomenko E. Using Embedding-Based Similarities to Improve Lexical Resources // Lobachevskii Journal of Mathematics. — 2021. — Т. 42, № 7. — С. 1532—1546. — [Scopus: Impact Factor 0.969].

36. Nikishina I. [и др.]. Taxonomy Enrichment with Text and Graph Vector Representations // Semantic Web. — 2022. — Т. 13, № 3. — С. 441—475. — [WoS: Impact Factor 2.214].

37. Тихомиров М. Разработка автоматизированной системы пополнения таксономии на текстах конкретной предметной области // Интеллектуальные системы. Теория и приложения. — 2021. — Т. 25, № 4. — С. 250—254. — [RINC: Impact Factor 0.192].

38. Firth J. A Synopsis of Linguistic Theory, 1930-1955. Studies in Linguistic Analysis (Volume of the Philological Society). — 1957.

39. Mikolov T. [и др.]. Distributed representations of words and phrases and their compositionality // Advances in neural information processing systems. — 2013. — С. 3111—3119.

40. Bojanowski P. [и др.]. Enriching word vectors with subword information // Transactions of the Association for Computational Linguistics. — 2017. — Т. 5. — С. 135—146.

41. Pennington J., Socher R., Manning C. GloVe: Global Vectors for Word Representation // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). — Association for Computational Linguistics, 2014. — С. 1532—1543.

42. Perozzi B., Al-Rfou R., Skiena S. Deepwalk: Online learning of social representations // Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. — 2014. — С. 701—710.

43. Grover A., Leskovec J. node2vec: Scalable feature learning for networks // Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. — 2016. — С. 855—864.

44. Yang C. [и др.]. Network representation learning with rich text information // Twenty-fourth international joint conference on artificial intelligence. — 2015.

45. Bordes A. [и др.]. Translating embeddings for modeling multi-relational data // Advances in neural information processing systems. — 2013. — Т. 26.

46. Nickel M., Kiela D. Poincaré Embeddings for Learning Hierarchical Representations // Advances in Neural Information Processing Systems 30 / под ред. I. Guyon [и др.]. — Curran Associates, Inc., 2017. — С. 6341—6350.

47. Aly R. [и др.]. Every child should have parents: a taxonomy refinement algorithm based on hyperbolic term embeddings // arXiv preprint arXiv:1906.02002. — 2019.

48. Kipf T. N., Welling M. Semi-supervised classification with graph convolutional networks // arXiv preprint arXiv:1609.02907. — 2016.

49. Coates J., Bollegala D. Frustratingly Easy Meta-Embedding–Computing Meta-Embeddings by Averaging Source Word Embeddings // arXiv preprint arXiv:1804.05262. — 2018.

50. Rubenstein H., Goodenough J. B. Contextual correlates of synonymy // Communications of the ACM. — 1965. — Т. 8, № 10. — С. 627—633.

51. Miller G. A., Charles W. G. Contextual correlates of semantic similarity // Language and cognitive processes. — 1991. — Т. 6, № 1. — С. 1—28.

52. Finkelstein L. [и др.]. Placing search in context: The concept revisited // Proceedings of the 10th international conference on World Wide Web. — 2001. — С. 406—414.

53. Luong M.-T., Socher R., Manning C. D. Better word representations with recursive neural networks for morphology // Proceedings of the seventeenth conference on computational natural language learning. — 2013. — С. 104—113.

54. Hill F., Reichart R., Korhonen A. Simlex-999: Evaluating semantic models with (genuine) similarity estimation // Computational Linguistics. — 2015. — Т. 41, № 4. — С. 665—695.

55. Yin W., Schütze H. Learning word meta-embeddings // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — 2016. — С. 1351—1360.

56. Bollegala D., Bao C. Learning word meta-embeddings by autoencoding // Proceedings of the 27th international conference on computational linguistics. — 2018. — С. 1650—1661.

57. Neill J. O., Bollegala D. Meta-embedding as auxiliary task regularization // arXiv preprint arXiv:1809.05886. — 2018.

58. Hochreiter S., Schmidhuber J. Long short-term memory // Neural computation. — 1997. — Т. 9, № 8. — С. 1735—1780.

59. Mikolov T. [и др.]. Recurrent neural network based language model. // Interspeech. Т. 2. — Makuhari. 2010. — С. 1045—1048.

60. Howard J., Ruder S. Universal language model fine-tuning for text classification // arXiv preprint arXiv:1801.06146. — 2018.

61. Pan S. J., Yang Q. A survey on transfer learning // IEEE Transactions on knowledge and data engineering. — 2009. — Т. 22, № 10. — С. 1345—1359.

62. Deng J. [и др.]. Imagenet: A large-scale hierarchical image database // 2009 IEEE conference on computer vision and pattern recognition. — Ieee. 2009. — С. 248—255.

63. Yosinski J. [и др.]. How transferable are features in deep neural networks? // arXiv preprint arXiv:1411.1792. — 2014.

64. Vaswani A. [и др.]. Attention is all you need // arXiv preprint arXiv:1706.03762. — 2017.

65. Bahdanau D., Cho K., Bengio Y. Neural machine translation by jointly learning to align and translate // arXiv preprint arXiv:1409.0473. — 2014.

66. Aldine A. I. A. [и др.]. Redefining Hearst Patterns by using Dependency Relations. // KEOD. — 2018. — С. 146—153.

67. Sabirova K., Lukanin A. Automatic Extraction of Hypernyms and Hyponyms from Russian Texts. // AIST (Supplement). — 2014. — С. 35—40.

68. Nakashole N., Weikum G., Suchanek F. PATTY: A taxonomy of relational patterns with semantic types // Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. — 2012. — С. 1135—1145.

69. Snow R., Jurafsky D., Ng A. Y. Learning syntactic patterns for automatic hypernym discovery // Advances in Neural Information Processing Systems 17. — 2004.

70. Fu R. [и др.]. Learning semantic hierarchies via word embeddings // Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — 2014. — С. 1199—1209.

71. Yamane J. [и др.]. Distributional hypernym generation by jointly learning clusters and projections // Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. — 2016. — С. 1871—1879.

72. Camacho-Collados J. [и др.]. SemEval-2018 task 9: Hypernym discovery // Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018); 2018 Jun 5-6; New Orleans, LA. Stroudsburg (PA): ACL; 2018. p. 712–24. — ACL (Association for Computational Linguistics). 2018.

73. Bernier-Colborne G., Barriere C. Crim at semeval-2018 task 9: A hybrid approach to hypernym discovery // Proceedings of the 12th international workshop on semantic evaluation. — 2018. — С. 725—731.

74. Bojanowski P. [и др.]. Enriching Word Vectors with Subword Information // Transactions of the Association for Computational Linguistics. — 2017. — Т. 5. — С. 135—146.

75. Arefyev N. [и др.]. Word2vec not dead: predicting hypernyms of co-hyponyms is better than reading definitions // Computational Linguistics and Intellectual Technologies: papers from the Annual conference “Dialogue”. — 2020.

76. Dale D. A simple solution for the Taxonomy enrichment task: Discovering hypernyms using nearest neighbor search // Computational Linguistics and Intellectual Technologies: papers from the Annual conference “Dialogue”. — 2020.

77. Liu N. [и др.]. On interpretation of network embedding via taxonomy induction // Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. — 2018. — С. 1812—1820.

78. Aly R. [и др.]. Every Child Should Have Parents: A Taxonomy Refinement Algorithm Based on Hyperbolic Term Embeddings // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. — Florence, Italy : Association for Computational Linguistics, 2019. — С. 4811—4817.

79. Nickel M., Kiela D. Poincar\’e embeddings for learning hierarchical representations // arXiv preprint arXiv:1705.08039. — 2017.

80. Nikishina I. [и др.]. Evaluation of Taxonomy Enrichment on Diachronic WordNet Versions . // Proceedings of the 11th Global WordNet conference GWC-2021. — 2021.

81. Sang E. F., De Meulder F. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition // arXiv preprint cs/0306050. — 2003.

82. Mozharova V. A., Loukachevitch N. V. Combining knowledge and CRF-based approach to named entity recognition in Russian // International Conference on Analysis of Images, Social Networks and Texts. — Springer. 2016. — С. 185—195.

83. Starostin A. [и др.]. FactRuEval 2016: evaluation of named entity recognition and fact extraction systems for Russian. — 2016.

84. Collins M., Singer Y. Unsupervised models for named entity classification // 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora. — 1999.

85. Li Y., Bontcheva K., Cunningham H. SVM based learning system for information extraction // International Workshop on Deterministic and Statistical Methods in Machine Learning. — Springer. 2004. — С. 319—339.

86. Boser B. E., Guyon I. M., Vapnik V. N. A training algorithm for optimal margin classifiers // Proceedings of the fifth annual workshop on Computational learning theory. — 1992. — С. 144—152.

87. Liu S. [и др.]. Effects of semantic features on machine learning-based drug name recognition systems: word embeddings vs. manually constructed dictionaries // Information. — 2015. — Т. 6, № 4. — С. 848—865.

88. Lafferty J., McCallum A., Pereira F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. — 2001.

89. Awasthi P., Rao D., Ravindran B. Part of speech tagging and chunking with hmm and crf // Proceedings of NLP Association of India (NLPAI) Machine Learning Contest 2006. — 2006.

90. Peng F., Feng F., McCallum A. Chinese segmentation and new word detection using conditional random fields // COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics. — 2004. — С. 562—568.

91. Rabiner L. R. A tutorial on hidden Markov models and selected applications in speech recognition // Proceedings of the IEEE. — 1989. — Т. 77, № 2. — С. 257—286.

92. Collobert R. [и др.]. Natural language processing (almost) from scratch // Journal of machine learning research. — 2011. — Т. 12, ARTICLE. — С. 2493—2537.

93. Huang Z., Xu W., Yu K. Bidirectional LSTM-CRF models for sequence tagging // arXiv preprint arXiv:1508.01991. — 2015.

94. Chiu J. P., Nichols E. Named entity recognition with bidirectional LSTM-CNNs // Transactions of the Association for Computational Linguistics. — 2016. — Т. 4. — С. 357—370.

95. Kuratov Y., Arkhipov M. Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language // arXiv preprint arXiv:1905.07213. — 2019.

96. DeepPavlov-documentation. http://docs.deeppavlov.ai/en/master/. — (Дата обр. 25.12.2019).

97. Piskorski J. [и др.]. The second cross-lingual challenge on recognition, normalization, classification, and linking of named entities across Slavic languages // Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing. — 2019. — С. 63—74.

98. Bridges R. A. [и др.]. Automatic labeling for entity extraction in cyber security // arXiv preprint arXiv:1308.4941. — 2013.

99. Joshi A. [и др.]. Extracting cybersecurity related linked data from text // 2013 IEEE Seventh International Conference on Semantic Computing. — IEEE. 2013. — С. 252—259.

100. Gasmi H., Bouras A., Laval J. LSTM recurrent neural networks for cybersecurity named entity recognition // ICSEA. — 2018. — Т. 11. — С. 2018.

101. Lample G. [и др.]. Neural architectures for named entity recognition // arXiv preprint arXiv:1603.01360. — 2016.

102. Sirotina A., Loukachevitch N. Named entity recognition in information security domain for Russian // Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019). — 2019. — С. 1114—1120.

103. Berners-Lee T., Hendler J., Lassila O. The semantic web // Scientific american. — 2001. — Т. 284, № 5. — С. 34—43.

104. Gómez-Pérez A., Corcho O. Ontology languages for the semantic web // IEEE Intelligent systems. — 2002. — Т. 17, № 1. — С. 54—60.

105. Лукашевич Н. В. Тезаурусы в задачах информационного поиска. — 2010.

106. Леонтьева Н. Автоматическое понимание текстов: системы, модели // Ресурсы: учеб. пособие для студентов лингвистических факультетов вузов. М.: Издательский центр"Академия. — 2006.

107. Chavez N., Pfeiffer H., Hartley R. Using and Interfacing Background Knowledge in Story Understanding // Proceedings of SENSE-09 Workshop on Conceptual Structures for Extracting Natural Language Semantics–2009. — 2007.

108. Колмогоров А., Фомин С. Элементы теории функций и функционального анализа. — Litres, 2018.

109. Yao L., Mao C., Luo Y. KG-BERT: BERT for knowledge graph completion // arXiv preprint arXiv:1909.03193. — 2019.

110. Bernier-Colborne G., Barrière C. CRIM at SemEval-2018 Task 9: A Hybrid Approach to Hypernym Discovery // Proceedings of The 12th International Workshop on Semantic Evaluation. — New Orleans, Louisiana : Association for Computational Linguistics, 06.2018. — С. 725—731. — URL: https://www.aclweb.org/anthology/S18-1116.

111. Loukachevitch N. Corpus-based Check-up for Thesaurus // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. — 2019. — С. 5773—5779.

112. Schultz M., Joachims T. Learning a distance metric from relative comparisons // Advances in neural information processing systems. — 2004. — Т. 16. — С. 41—48.

113. Wang J. [и др.]. Learning fine-grained image similarity with deep ranking // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2014. — С. 1386—1393.

114. Schroff F., Kalenichenko D., Philbin J. Facenet: A unified embedding for face recognition and clustering // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2015. — С. 815—823.

115. Wei J. [и др.]. Few-Shot Text Classification with Triplet Networks, Data Augmentation, and Curriculum Learning // arXiv preprint arXiv:2103.07552. — 2021.

116. Dobrov B. V., Loukachevitch N. V. Development of Linguistic Ontology on Natural Sciences and Technology. // LREC. — Citeseer. 2006. — С. 1077—1082.

117. Добров Б., Лукашевич Н. Онтология по естественным наукам и технологиям ОЕНТ: структура, состав и современное состояние // Электронные библиотеки. — 2008. — Т. 11, № 1.

118. Tikhomirov M., Loukachevitch N., Dobrov B. Methods for Assessing Theme Adherence in Student Thesis // International Conference on Text, Speech, and Dialogue. — Springer. 2019. — С. 69—81. — [Scopus: Impact Factor 1.363].

119. Kipf T. N., Welling M. Semi-Supervised Classification with Graph Convolutional Networks // International Conference on Learning Representations (ICLR). — 2017.

120. Hamilton W. L., Ying R., Leskovec J. Inductive representation learning on large graphs // Proceedings of the 31st International Conference on Neural Information Processing Systems. — 2017. — С. 1025—1035.

121. Grover A., Leskovec J. node2vec: Scalable Feature Learning for Networks // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. — 2016.

122. Cho Y. [и др.]. Leveraging WordNet Paths for Neural Hypernym Prediction // Proceedings of the 28th International Conference on Computational Linguistics. — Barcelona, Spain (Online) : International Committee on Computational Linguistics, 12.2020. — С. 3007—3018. — URL: https://www.aclweb.org/anthology/2020.coling-main.268.

123. Shen J. [и др.]. TaxoExpan: Self-supervised Taxonomy Expansion with Position-Enhanced Graph Neural Network // Proceedings of The Web Conference 2020. — 2020. — С. 486—497.

124. Wang W. Y., Yang D. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets // Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. — 2015. — С. 2557—2563.

125. Kobayashi S. Contextual augmentation: Data augmentation by words with paradigmatic relations // arXiv preprint arXiv:1805.06201. — 2018.

126. Wei J., Zou K. Eda: Easy data augmentation techniques for boosting performance on text classification tasks // arXiv preprint arXiv:1901.11196. — 2019.

127. Wu Y. [и др.]. Google's neural machine translation system: Bridging the gap between human and machine translation // arXiv preprint arXiv:1609.08144. — 2016.

## Список рисунков

## Список таблиц