Санкт-Петербургский государственный университет

ПРЕДЕЛИНА Анастасия Игоревна

Выпускная квалификационная работа

Нейросетевые методы выделения сочинительных связей

Уровень образования: бакалавриат
Направление 02.03.01 «Математика и компьютерные науки»
Основная образовательная программа:
СВ.5152.2019 «Математика, алгоритмы и анализ данных»

Научный руководитель:
Доцент,
Факультет математики
и компьютерных наук СПбГУ,
Авдюшенко Александр Юрьевич

Рецензент:
Инженер ключевых проектов,
Общество с ограниченной
ответственностью
«Техкомпания Хуавей»,
Малых Валентин Андреевич

Санкт-Петербург
2023 год

Содержание

1. Введение

1.1. **Задача выделения сочинительных связей.** Задача выделения сочинительных связей (Coordination Analysis, CA) состоит в том, чтобы научиться находить внутри предложений синтаксические структуры, соединяющие грамматически равноправные части предложений (такие структуры и называются в лингвистике “coordinations”) [31]. Например, в предложении “Susan works [slowly] and [carefully]” такой структурой является “[slowly] and [carefully]”. В ней союз “and” связывают два независимых обстоятельства: “slowly” и “carefully” (см. Рис. 1).

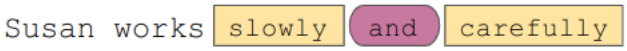

Рис. 1. Пример предложения с сочинительной связью.

Решение этой задачи позволяет устанавливать потенциально ценные связи и отношения между определёнными частями предложения. В том числе поэтому выделение сочинительных связей — важный инструмент предобработки текстов. Так, в работе [14] предложение с сочинительными связями преобразовывается в несколько «простых», и метод решения целевой задачи (OpenIE) применяется уже к ним, что даёт значительный прирост в качестве. В работах [14, 28] показано, что этот этап подготовки значительно влияет на итог работы соответствующих методов.

1.1.1. *Решение задачи CA методами машинного обучения.* Одна из первых статей, в которой задача выделения сочинительных связей решалась с помощью нейронных сетей, была опубликована **Ficler и Goldberg в 2016 году** [7]. Идея основана на двух лингвистических свойствах сочинительных структур: (1) отдельные части таких структур «похожи», выполняют сходную фунцию в предложении; (2) при замене всей структуры на любую из её составных частей предложения остаются осмысленными.

Для входного предложения строится дерево разбора (см. пример на Рис. 2).

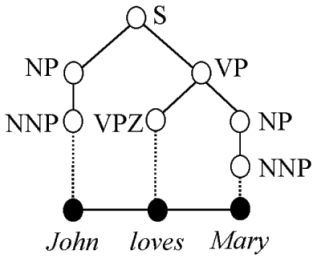

Рис. 2. Построение дерева разбора входного предложения.

Подход к решению задачи, предложенный в статье, состоит в том, чтобы для каждого сочинительного союза (к ним авторы относят слова из списка: “and”, “nor”, “or” и “but”) определить, задаёт ли он какую-либо сочинительную структуру, и, если задаёт, найти индексы начала и конца первой и второй частей данной структуры.

Алгоритм состоит из 3 шагов: сначала для каждого сочинительного союза с помощью бинарного классификатора предсказывается, задаёт ли он некоторую сочинительную структуру. Если да, то на следующем шаге извлекается ранжированный список возможных структур, где кандидатом является пара промежутков вида $((i, j), (l, m))$. Затем оцениваются кандидаты, и пара, набравшая наибольшее количество очков, возвращается в качестве ответа.

Модель для подсчёта значений ранжирующей функции на парах промежутков состоит из 3 компонентов: компонент, связанный с первым свойством, вычисляет расстояние между выходами модели LSTM (Long Short-Term Memory network) [10] по последовательностям частеречных меток ("POS-тегов") по пути от конъюнкта до вершины в дереве разбора. Компонент, учитывающий второе свойство, получает 2 вектора из выходов LSTM по последовательностям "POS-тегов" токенов предложения с предварительно удалёнными из него первым и вторым промежутком соответственно. Третий компонент подсчитывает дополнительные характеристики, получаемые из синтаксического парсера. Обучение данной модели представляет собой минимизацию попарной функции потерь для задачи ранжирования на парах извлечённых связей-кандидатов.

Несмотря на то, что подход Ficler и Goldberg дал существенный прирост качества решения данной задачи, у него были недостатки, связанные с тем, что он сильно зависит от синтаксического парсера: зависимость от внешней модели, замедление скорости работы и сохранение ошибок парсера.

В 2017 году появился новый метод, направленный на решение проблем выше. Он был описан в статье [26] (здесь и далее будем обозначать его как **"Teranishi-17"**). Данный подход не строит **синтаксических деревьев**, использует только токены предложения и их частеречные метки ("POS-теги"). Модель состоит из 4 основных частей.

(1) Построение векторных представлений по токенам и их "POS-тегам".
(2) Двунаправленная рекуррентная сеть, строящая по входной последовательности векторные представления на уровне предложения.
(3) Подсчёт векторных представлений возможных сочинительных структур и векторов признаков 2 соединяемых данной структурой частей (вектора признаков основаны на свойствах сочинительных структур, сформулированных выше 1.1.1).
(4) Подсчёт очков по всем возможным парам соединяемых частей с помощью Многослойного перцептрона. Выбор пары с наибольшим количеством очков.

В 2019 году теми же авторами была опубликована статья [27] (здесь и далее "**Teranishi-19**"), в которой описанные выше идеи обобщались и улучшались. Во-первых, вместо одной модели, генерирующей пары соединяемых сочинительными структурами частей появляется 2: внутренняя и внешняя. Внутренняя модель генерирует индексы начала и конца ближайших слева и справа соединяемых частей, а внешняя — индексы начала и конца первой и последней соединяемых частей структуры. Таким образом данный подход способен предсказывать не

только структуры, соединяющие 2 части, но и больше. Также внутренняя и внешняя модели подсчитывают очки для каждой из пар-кандидатов. Во-вторых, поверх описанных моделей запускается алгоритм Кока-Янгера-Касами [24] для построения дерева разбора в рамках особой грамматики, задающей правила сочинительных структур. Это сделано для того, чтобы найти «наилучший» вариант сочинительной связи, исходя из структуры предложения.

В последнее время наиболее перспективные подходы к СА состоят в использовании **моделей, решающих задачи, сходные с извлечением информации (Information extraction, IE)**. Так, одна из разновидностей задачи, Open Information Extraction, заключается в извлечении троек вида «субъект, предикат, объект». Например, в предложении "John managed to open the door" такой тройкой является *(John, managed to open, the door)*. Модель OpenIE для входного предложения должна выдавать набор «масок», где каждый токен будет помечен одним из 4 классов: «субъект», «объект», «отношение» (предикат) или «фон» (т.е. ни один из предыдущих классов). Аналогично можно сформулировать и задачу СА: для входного предложения модель должна научиться находить набор масок, где каждый токен помечен одним из 6 классов: `CP_START`, `CP`, `CC`, `SEP`, `OTHERS` или `NONE` (`CP_START` — токен, с которого начинается структура; `CP` — элементы, соединяемые союзом, `CC` — союз; `SEP` — разделители разных частей структуры, например, запятые, `OTHERS` — всё, что не относится к категориям выше, но присутствует в структуре; `NONE` — фоновые слова, не относящиеся к структуре).

Так, пример, приведённый выше, будет размечен как на Рис. 3:

Рис. 3. Пример предложения с сочинительной связью и соответствующей разметкой.

1.2. **Современные нейросетевые подходы к обработке текста как последовательности.** В этом разделе будет приведён обзор нейросетевых методов, связанных с предлагаемым исследованием.

1.2.1. *Архитектура Transformer.* Для того, чтобы перейти к описанию последующих моделей, решающих задачу СА, рассмотрим подробнее важную часть их архитектуры — модель BERT [5], основанную на элементах архитектуры Transformer [29].

Изначально Transformer — нейросетевая архитектура «кодировщик-декодировщик» (encoder-decoder, см. Рис. 4), использующая механизм self-attention («само-внимание»), помогающий кодировщику посмотреть на другие токены во входном предложении во время кодирования текущего токена. Причём архитектура данной модели устроена так, что все вычисления для отдельных токенов происходят параллельно, что делает обучение Transformer и предсказания с их помощью очень производительными.

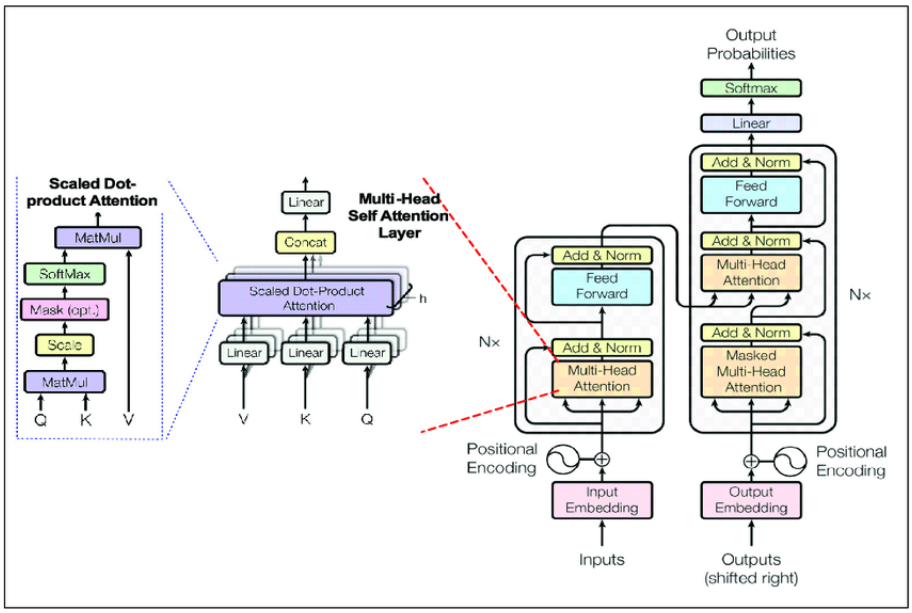

Рис. 4. Архитектура Transformer.

До публикации статьи [29], в которой впервые была представлена архитектура Transformer, достигающими лучшего качества в области обработки естественного языка моделями в задачах, рассматривающих текст как последовательность, как правило, были рекуррентные нейронные сети с механизмом внимания [2]. Однако у этих моделей был существенный недостаток: вычисления в них авторегрессивны, то есть токены обрабатываются последовательно, и для того, чтобы вычислить вектор состояния текущего токена, нужно проделать эту процедуру для всех предыдущих токенов. Архитектура из последовательных блоков рекуррентной сети приводит к нескольким проблемам: во-первых, к затуханию градиентов при обработке длинных последовательностей, а во-вторых, к невозможности сделать вычисления параллельными, поэтому обучение данной модели получается достаточно долгим.

Трансформеры дали сразу ряд преимуществ: вычислительная производительность (качество работы превзошло качество других моделей, то же верно для скорости работы), сокращение «максимальных путей» между словами (в трансформерах, в отличие от рекуррентных моделей, для формирования векторных представлений (эмбеддингов) необязательно учитывать абсолютно все токены последовательности, веса внимания позволяют убрать слишком далёкие слова, не влияющие на текущий токен [29]; в рекуррентных сетях каждый предыдущий токен влияет на вектор состояния следующего), возможность построения контекстозависимых векторных представлений, а также потенциальная интерпретируемость модели с помощью карт внимания.

Архитектура Transformer стала основой очень большого количества новых моделей. Например, появились модели, строящие позволившие добиться улучшения качества в большом числе задач контексто-зависимые векторные представления слов, основанные

на кодировщике «трансформера»: Bidirectional Encoder Representations from Transformers (BERT) [5], Enhanced Representation through Knowledge Integration (ERNIE) [30], Lite BERT (ALBERT) [16], Robustly Optimized BERT (RoBERTa) [17] и т.п.

Также на основе декодировщика Transformer были предложены порождающие языковые модели — например, семейство моделей Generative Pre-trained Transformer (GPT) [21], которые, в числе прочего, авторегрессионным способом могут генерировать новый текст.

1.2.2. *Метод BERT.* Как было сказано выше, метод BERT используется для построения хороших контексто-зависимых векторных представлений слов и основан на кодировщике Transformer.

Более ранние «популярные» архитектуры, решающие данную задачу, учились предсказывать следующий токен, опираясь только на предыдущие токены. Однако интуитивно контекст должен определяться не только предыдущими токенами, но и последующими. Для решения этой проблемы были придуманы двунаправленные модели, которые учитывают контекст и слева, и справа: например, модель Embeddings from Language Models, ELMo [20] обучает параллельно две вспомогательные модели, одна из которых предсказывает следующее слово по предыдущим, а вторая — текущее слово по последующим. Итоговое векторное представление представляет собой конкатенацию двух векторов левого и правого контекстов соответственно.

BERT [5] также можно отнести к «двунаправленным» моделям: она учитывает два направления контекста, но делает это по-другому.

На вход подаются токенизированные пары предложений с некоторыми скрытыми (маскированными) токенами (см. Рис. 5). Одна из двух задач, на которых одновременно обучается модель, — это задача генерации пропущенного токена (Cloze task). Модель обучается предсказывать намеренно «пропущенное» в тексте слово. Вторая задача, на которой обучается данная модель, — это задача бинарной классификации. Она состоит в определении того, является ли второе предложение на входе продолжением первого. Решая эту задачу, модель учится различать наличие связи между предложениями в тексте[1].

1.2.3. *OpenIE6.* Достигающая в настоящий момент лучшего качества модель IGL-CA из работы, посвящённой OpenIE6 [14], решает задачу CA, используя подход, описанный выше. Авторы подготовили новую модель для решения задачи OpenIE, а затем изменили количество классов с 4 («объект», «субъект», «предикат», «пропуск») до 6 (`CP_START`, `CP`, `CC`, `SEP`, `OTHERS`, `NONE`; см. выше) и применили точно такой же подход к задаче CA.

Основная идея заключается в итеративном извлечении троек, Iterative Grid Labeling («разметка решётки»). Эта идея основана на архитектуре Iterative Memory-Based Joint Open Information Extraction (IMoJIE) [8], которая на момент 2020 года была лучшей по качеству

[1]Ряд нюансов схемы обучения BERT опущен, подробнее описано в исходной работе [5].

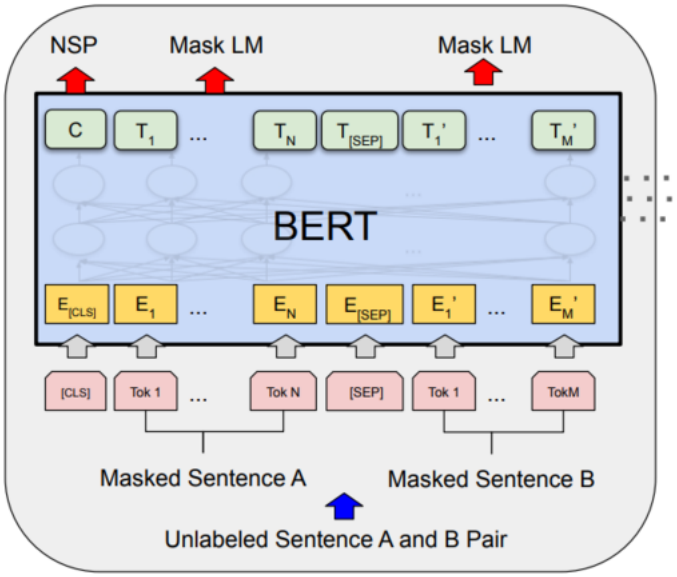

Рис. 5. Обучение модели BERT.

моделью для решения задачи OpenIE. В IMoJIE тройки (триплеты) извлекаются последовательно, маска за маской, причём полученные моделью эмбеддинги на предыдущей итерации передаются на вход следующей. Это сделано для того, чтобы не повторять извлечения с предыдущих шагов. Для извлечения векторных представлений исходных токенов входного предложения используется модель BERT [5]. Существенным недостатком IMoJIE является низкая скоростью работы. Поэтому для нахождения баланса между скоростью работы и качеством предсказаний была разработана модель OpenIE6. В ней триплеты извлекаются также итеративно, а для ускорения работы первоначальные эмбеддинги слов последовательности рассчитываются с помощью BERT только один раз (в IMoJIE они считаются на каждом шаге извлечения маски).

Также для улучшения качества модели OpenIE добавляются регуляризаторы, учитывающие различные лингвистические ограничения и включаемые как слагаемые в функцию потерь (невязку).

Например, одно из лингвистических ограничений в OpenIE6 — добавка POSC — отвечает за то, чтобы каждое слово, относящееся к части речи из заданного авторами списка (существительное, глагол, прилагательное и наречие) было хотя бы в одном из извлечённых триплетов для данного входного предложения.

Формула регуляризатора POSC следующая:

$$J_{posc} = \sum_{t=1}^{T} x_t^{imp} \cdot Pr_t$$

$$Pr_t = 1 - \max_{n=1}^{N} \max_{k \in \{S,R,O\}} p_{k,t,n},$$

где $p_{k,t,n}$ — предсказанная моделью вероятность принадлежности токена t в маске n к классу k (в данном примере S — “subject”, R — “relation”, O — “object”); T — число токенов в маске, N — число извлекаемых масок; x_t^{imp} — индикатор токенов с интересующими частями речи, он равен 1, если токен t относится к части речи из списка, в ином случае равен 0. J_{posc} входит с определённымв весом как слагаемое в итоговую функцию потерь.

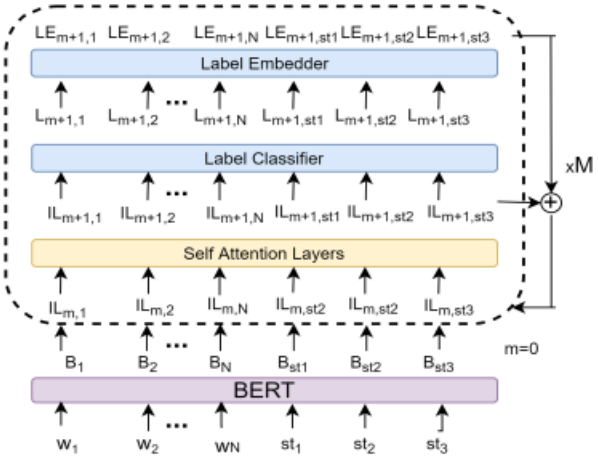

Рис. 6. Архитектура модели OpenIE6 [14].

1.2.4. *DetIE как базовая модель.* В феврале 2022 года была опубликована статья [28], представляющая новую модель для решения задачи OpenIE, вдохновлённую идеей одностадийных детекторов (one-stage) из компьютерного зрения.

В области компьютерного зрения детекторами называются модели, которые учатся находить определённые объекты на изображениях и выделять их, например, ограничивающими рамками (bounding boxes, bboxes; см. Рис. 7). Одностадийные детекторы за один проход одновременно формируют признаки, предсказывают класс объекта внутри текущего bbox и «координаты» этого bbox на изображении.

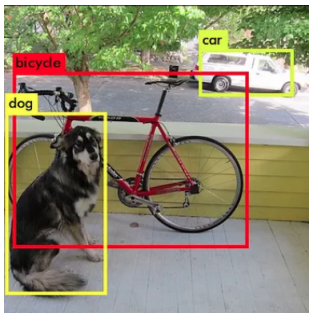

Рис. 7. Пример предсказания одностадийного детектора YOLO [23].

Архитектура модели DetIE состоит из 3 основных частей: токенизация входного текста; извлечение эмбеддингов при помощи BERT, последние слои которого «размораживаются» (то

есть часть весов нейронной сети начинает обновляться в ходе обучения в рамках обратного распространения ошибок лишь начиная с некоторого шага/итерации) для того, чтобы иметь возможность «подкорректировать» (дообучить; fine-tuning) модель под задачу OpenIE; затем идёт полносвязный слой. На выходе моделью выдаётся фиксированное количество *масок* найденных структур N. Маски представляют собой тензор размера (T, N, C), в каждом элементе (t, n, c) которого записана вероятность p того, что токен t в маске n относится к классу c. Финальная маска структуры получается при помощи взятия argmax по всем классам.

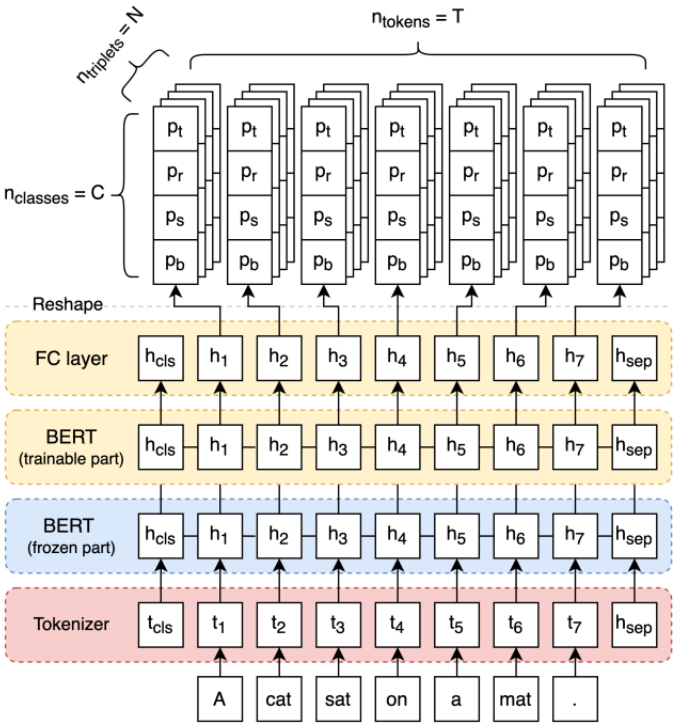

Рис. 8. Архитектура модели DetIE. Изображение из работы [28].

Функция потерь в DetIE аналогична той, что используется one-stage-детекторами: вводится оценка **IoU** (Intersection over Union — отношение пересечения ограничивающих прямоугольников к площади их объединения) между триплетами, извлечёнными моделью, и реальными метками. IoU рассчитывается следующим образом: пусть n — это номер извлечённой маски, m — номер истинной бинарной маски, тогда значение оценки IoU между ними равно:

$$\mathbf{IoU}_{nm} = \frac{I_{nm}}{U_{nm}}$$

$$I_{nm} = \sum_{t,c} p_{tnc} \cdot l_{tmc},$$

$$U_{nm} = \sum_{t,c} (p_{tnc} + l_{tmc}) - I_{nm},$$

где p_{tnc} — предсказанная моделью вероятность того, что токен t в маске n относится к классу c, l_{tmc} — 0 или 1 в зависимости от того, относится ли токен t к классу c в истинной маске m.

На данном этапе есть матрица размера $N \times M$ со значениями **IoU** между истинными метками и предсказанными моделью вероятностями. Для того, чтобы определить, каким именно истинным маскам соответствует набор извлечённых вероятностей, используется Венгерский алгоритм [15], предназначенный для решения линейной задачи о назначениях (Linear Assignment problem, LAP). Эта задача заключается в максимизации или минимизации $\mathrm{Tr}\,[P \cdot C]$ для некоторой фиксированной неотрицательной квадратной матрицы стоимостей C по множеству её матриц перестановок P. Данную задачу можно обобщить и на случай неквадратной матрицы с помощью добавления фиктивных столбцов или строчек, состоящих из максимально возможных значений в случае максимизации и из нулей в случае минимизации. Для неотрицательной неквадратной матрицы **IoU** из DetIE с помощью решения LAP находится набор сопоставлений между истинными масками и извлечёнными, максимизирующий сумму **IoU** по соответствиям из набора. Так как $M \leq N$, то в матрицу **IoU** предварительно добавляются фиктивные столбцы, а в итоговом решении $(N - M)$ предсказаний модели не будут сопоставлены ни одной из истинных масок. В коде DetIE используется реализация lapsolver [4].

В качестве функции потерь используется кросс-энтропия, вычисленная по полученному набору соответствий.

Преимуществами модели DetIE являются быстрый inference (предсказание), а также данный подход на нескольких наборах данных для оценки качества OpenIE продемонстрировал качество более высокое, чем OpenIE6.

1.3. **Структура работы.** Последующее содержание состоит из 3 основных глав: «Постановка задачи», «Методология» и «Эксперименты и результаты».

В первой из них сформулирована цель квалификационной работы. В главе «Методология» 3 описаны используемые данные, оценка качества и более подробно рассказано об основных этапах работы модели DetIE, которую было решено использовать в качестве базового подхода к решению задачи выделения сочинительных связей. В главе «Эксперименты и результаты» 4 проведён анализ ошибок базовой модели, описаны все апробированные нововведения в процедурах предобработки и пост-обработки данных и в архитектуре модели (они предложены на основе анализа ошибок, поэтому описаны в данной главе, а не в «Методологии»), а также степень их влияния на качество решения. В разделе «Заключение» 5 подведены итоги и предложены варианты дальнейшей работы над рассмотренной задачей.

2. Постановка задачи

Цель настоящей работы — разработать нейросетевую модель для выделения сочинительных связей из предложений на английском языке, имеющую высокую скорость предсказаний и сопоставимую по качеству с лучшим из современных методов решения.

В частности, таким методом является IGL-CA [14] (см. раздел 1.2.3); для честности сопоставления в работе производится сравнение с методами, в которых предобученный кодировщик имеет сходное число параметров (то есть, в частности, `bert-base-*`).

Для достижения цели в ходе работы было решено адаптировать модель DetIE [28] для решения задачи выделения сочинительных связей, провести анализ ошибок, в том числе на основе которого выдвинуть предположения, что можно изменить в исходных архитектуре и процедуре подготовки данных для улучшения результатов, затем последовательно улучшать модель, следуя разработанному плану.

3. Методология

3.1. **Данные.** В качестве данных для обучения и тестирования используется набор данных Penn Treebank CA (PTB-CA; он же использовался для обучения и оценки качества выделения сочинительных связей в работе OpenIE6 [14]). Он был получен в ходе улучшения исходной «сырой» разметки сочинительных связей в PTB [19]. Подходы к улучшению были описаны в статье Ficler и Goldberg [6].

В датасете PTB-CA тренировочная часть состоит из 17282 предложений, валидационная — из 742, а тестовая — из 985.

Для каждого предложения в этом датасете выделено по три маски сочинительных связей. Маски также могут быть полностью пустыми: в них каждый токен размечен как фоновый класс. Так, в примере на Рис. 9 первая маска описывает сочинительную связь на уровне соединения двух простых предложений в составе сложного, вторая — связь, соединяющую однородные подлежащие, а третья является пустой.

```
|   Mr.  Guber and  Mr. Peters are  n't universally loved  in Hollywood but  they are  well connected  .
CP_START CP   CP   CP   CP     CP   CP     CP       CP   CP     CP     CC   CP   CP   CP       CP     NONE
CP_START CP   CC   CP   CP    NONE NONE   NONE      NONE NONE   NONE   NONE NONE NONE NONE     NONE   NONE
  NONE   NONE NONE NONE NONE   NONE NONE   NONE      NONE NONE   NONE   NONE NONE NONE NONE     NONE   NONE
```

Рис. 9. Пример размеченного предложения из набора данных PTB-CA.

3.2. **Оценка качества.** Был подготовлен и отлажен программный код для оценки решений, аналогичный использованному в OpenIE6 [14], чтобы иметь возможность сравнивать качество работы моделей. Качество работы модели оценивается на тестовой части описанного выше набора данных PTB-CA путём вычисления точности (precision), полноты (recall) и F1-меры (F1 score).

Опишем более подробно процедуру подсчёта оценок качества. Изначально инициализируется нулями следующий набор счётчиков: `tp_t, tp_f, fn, fp` («True Positive корректные», «True Positive с ошибкой», False Negative, False Positive). Для каждого предложения есть размеченные маски сочинительных связей из датасета и предсказанные моделью маски. Заметим, что каждый сочинительный союз предложения может быть соединяющим элементом в ≤ 1 сочинительной структуре. Из всех истинных и всех предсказанных масок предложения извлекаются непрерывные отрезки из токенов нефонового класса и сохраняются вместе с разметкой в соответствующий словарь с ключом, равным индексу сочинительного союза, размеченного как `CC` внутри текущего отрезка. Затем для каждой истинной сочинительной структуры в словаре с предсказанными структурами ищется соответствующая по индексу соединяющего её сочинительного союза. Если предсказанная структура с данным союзом в качестве соединяющего элемента действительно существует, то производится сравнение предсказанных и истинных меток внутри структуры: если все метки и координаты начала и конца непрерывных отрезков, задающих структуры, совпадают, то к счётчику `tp_t` добавляется 1; если же хотя бы одна метка не совпала или координаты начала и конца отрезков не равны между собой, то к счётчику `tp_f` прибавляется 1. Если для текущей истинной структуры не найдено соответствия в словаре с предсказанными масками, то к счётчику `fn` добавляется 1. Затем, если в словаре предсказанных структур существуют такие, что к ним нельзя найти соответствующую в словаре с истинными структурами, то к счётчику `fp` добавляется количество таких структур.

Полученные после прохода по всем предложениям значения счётчиков используются далее для подсчёта точности и полноты, а именно:

$$Precision = P = \frac{\texttt{tp\_t}}{\texttt{tp\_t} + \texttt{tp\_f} + \texttt{fp}}$$

$$Recall = R = \frac{\texttt{tp\_t}}{\texttt{tp\_t} + \texttt{tp\_f} + \texttt{fn}}$$

F1-мера является средним гармоническим точности и полноты:

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

3.3. **Модель.** Исходная модель DetIE-CA (лишь с изменением числа классов с 4 на 6) принимает на вход предложение с добавленными токенами `CLS` и `SEP` в начало и конец соответственно.

Далее идёт несколько этапов, связанных с собственно архитектурой модели, описанной в разделе 1.2.4. А именно — сначала предложение разделяется на токены, затем извлекаются эмбеддинги с помощью предобученной модели BERT (с несколькими последними «размороженными» слоями, что будет важно в обучении). Далее эмбеддинги подаются на вход завершающему полносвязному слою. На выходе последнего слоя получается тензор размера

(T, N, C), где T — количество токенов во входном предложении (добавленные токены `CLS` и `SEP` не учитываются), N — предварительно заданное количество извлекаемых масок ($N = 5$), C — количество классов ($C = 6$). После применения к данному тензору преобразования `softmax` будет получен тензор такого же размера (T, N, C), в котором в позиции (t, n, c) будет записана величина, которую можно интерпретировать как вероятность p_{tnc} того, что токен t в маске n относится к классу c. Именно по этому тензору оценок вероятностей вычисляется функция потерь, подробно описанная выше в параграфе 1.2.4.

Следующий этап — агрегация тензора вероятностей в итоговый набор масок. В DetIE итоговая метка класса для каждого токена t в каждой маске n вычисляется «жадно», то есть выбирается класс c, для которого вероятность p_{tnc} наибольшая. Таким образом, после «жадной» агрегации получается тензор, заполненный номерами классов, размера (T, N).

Этапы, связанные с архитектурой модели и агрегацией меток, отражены на Рис. 10.

Количество токенов в предложении может не совпадать с количеством слов в нём. В датасете PTB-CA разметка сочинительных структур сделана по отдельным словам предложения, а модель BERT работает с токенами, полученными способом Byte Pair Encoding (BPE) [25]. По этой причине, кроме упомянутых выше шагов работы, также необходимы этапы предобработки и пост-обработки. Предобработка заключается в присвоении каждому токену в качестве истинного класса метки того слова, внутри которого находится данный токен (этот этап нужен для вычисления функции потерь). Пост-обработка представляет собой обратную операцию: по полученным после «жадной» агрегации меткам для токенов нужно получить метки по отдельным словам предложения (это необходимо для подсчёта оценки качества). В качестве метки слова выбирается метка последнего токена, содержащегося внутри данного слова.

Поскольку переосмысление изначального подхода, предлагаемое в данной квалификационной работе, как и предполагалось, основано на анализе ошибок описанного в этой главе **базового подхода**, все нововведения и оценка их влияния будут описаны в Главе 4.

4. Эксперименты и результаты

4.1. **Изначальное качество работы и процедура предобработки данных.** После обучения изначальной модели DetIE-CA на датасете PTB-CA, значение полноты на тестовой части было достаточно низким (см. первую строку Таблицы 1). Затем была найдена ошибка в подсчёте оценок качества, после её исправления значение F1-меры стало соответствовать ожидаемому.

В качестве следующего шага для подготовки цепочки обучения и валидации были воспроизведены оценки качества IGL-CA [14] (работа, посвящённая OpenIE6, обзор которой дан в Главе 1).

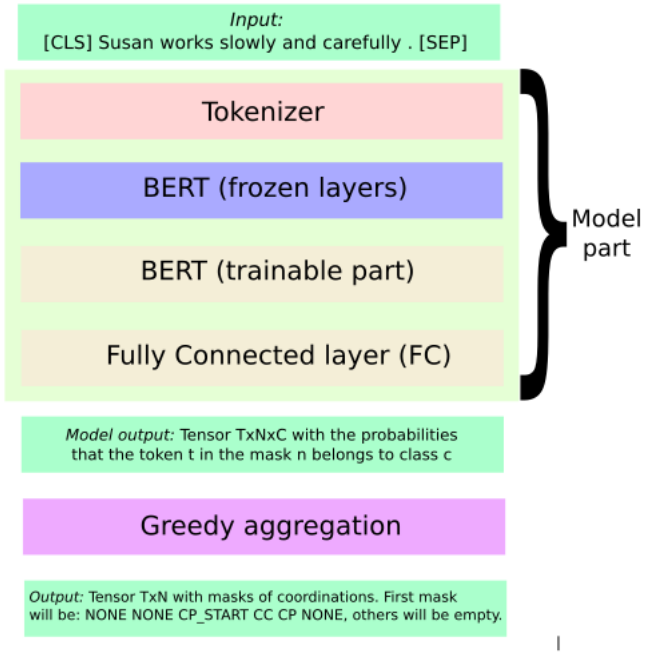

Рис. 10. Основные этапы работы DetIE-CA.

Работа с PTB-CA подразумевает специфическую предобработку данных, так как часть токенов в исходном корпусе была заменена на служебные ключевые слова. После приведения набора данных к виду, который можно напрямую использовать с BERT, то есть и с DetIE, и после обучения заново, результаты достигли значений в последней строке Таблицы 1.

Таблица 1. Оценка качества изначальной модели и после изменений в процедуре предобработки данных.

	Точность	**Полнота**	**F1-мера**
Изначальная DetIE-CA	0.843	0.722	**0.778**
После исправления ошибки	0.840	0.797	**0.818**
Замена тегов -LRB-, -RRB-, -LCB-, -RCB- на скобки	0.851	0.830	**0.840**

В дальнейшем полученный результат использовался как «отправная точка».

4.2. **Анализ ошибок.** Затем был проведён анализ ошибок модели на валидационной части датасета PTB-CA. Было выявлено два основных сценария ошибок:

- проблема «масштаба»,
- некорректность извлечённой маски.

В первом из них (проблема «масштаба») модель не находит структуры на «высоком уровне»: соединение простых предложений в составе сложного; также структуры, включающие большое количество второстепенных членов предложения (см. Рис. 11). Также сюда

можно отнести случаи, когда модель не «обнаруживает» структуры «низкого уровня», соединяющие однородные члены предложения (см. Рис. 12).

Ground truths:

1. Also under the agreement , debenture**[CP_START]** holders**[CP]** will**[CP]** get**[CP]** one**[CP]** million**[CP]** shares**[CP]** of**[CP]** common**[CP]** stock**[CP]** in**[CP]** exchange**[CP]** for**[CP]** $**[CP]** 7.7**[CP]** million**[CP]** in**[CP]** debentures**[CP]** and**[CC]** holders**[CP]** of**[CP]** BancOklahoma**[CP]** 's**[CP]** series**[CP]** A**[CP]** preferred**[CP]** stock**[CP]** will**[CP]** receive**[CP]** 1.25**[CP]** shares**[CP]** of**[CP]** common**[CP]** stock**[CP]** for**[CP]** every**[CP]** share**[CP]** of**[CP]** preferred**[CP]** they**[CP]** own**[CP]** , the company said .

Predictions:

detector 2: Also under the agreement , debenture holders will get one million shares of common stock in exchange for $ 7.7 million in debentures**[CP_START]** and**[CC]** holders**[CP]** of**[CP]** BancOklahoma**[CP]** 's**[CP]** series**[CP]** A preferred stock will receive 1.25 shares of common stock for every share of preferred they own , the company said .

Рис. 11. Нахождение моделью структуры «низкого уровня» вместо «высокого».

Ground truths:

1. The group includes Putnam**[CP_START]** Cos.**[CP]** and**[CP]** various**[CP]** affiliates**[CP]** based**[CP]** in**[CP]** Boston**[CP]** ;**[SEP]** Wells**[CP]** Fargo**[CP]** Bank**[CP]** ,**[CP]** San**[CP]** Francisco**[CP]** ;**[SEP]** the**[CP]** California**[CP]** Public**[CP]** Employees**[CP]** Retirement**[CP]** System**[CP]** ,**[CP]** Sacramento**[CP]** ,**[CP]** Calif.**[CP]** ,**[OTHERS]** and**[CC]** T.**[CP]** Rowe**[CP]** Price**[CP]** Associates**[CP]** Inc.**[CP]** ,**[CP]** Baltimore**[CP]** .
2. The group includes Putnam**[CP_START]** Cos.**[CP]** and**[CC]** various**[CP]** affiliates**[CP]** based**[CP]** in**[CP]** Boston**[CP]** ; Wells Fargo Bank , San Francisco ; the California Public Employees Retirement System , Sacramento , Calif. , and T. Rowe Price Associates Inc. , Baltimore .

Predictions:

detector 2: The group includes Putnam**[CP_START]** Cos.**[CP]** and**[CP]** various**[CP]** affiliates**[CP]** based**[CP]** in**[CP]** Boston**[CP]** ;**[SEP]** Wells**[CP]** Fargo**[CP]** Bank**[CP]** ,**[CP]** San**[CP]** Francisco**[CP]** ;**[SEP]** the**[CP]** California**[CP]** Public**[CP]** Employees**[CP]** Retirement**[CP]** System**[CP]** ,**[CP]** Sacramento**[CP]** ,**[CP]** Calif.**[CP]** ,**[OTHERS]** and**[CC]** T.**[CP]** Rowe**[CP]** Price**[CP]** Associates**[CP]** Inc.**[CP]** ,**[CP]** Baltimore**[CP]** .

Рис. 12. Извлечение моделью только структуры «высокого уровня» при наличии структуры «низкого уровня».

Во втором сценарии модель извлекает **некорректную маску** сочинительной связи. Например, иногда в структуру добавляются лишние артикли (см. Рис. 13) или не включаются элементы, отделённые запятыми (см. Рис. 14). Другой вариант — извлечение неправильной, с точки зрения постановки задачи, маски (см. Рис. 15 и 16): например, для каждой структуры метка `CP_START` должна быть ровно одна и должна соответствовать начальному токену структуры.

Большая часть из описанных далее подходов к улучшению исходной модели DetIE-CA направлены на решение выявленных в ходе анализа ошибок.

Для решения проблемы «масштаба» были рассмотрены следующие подходы: «чанкинг» (см. раздел 4.3), а также различные архитектурные изменения с добавлением в модель свёрточных слоёв (см. разделы 4.4.4, 4.4.5, 4.4.6). Для борьбы со некорректным извлечением

Ground truths:

1. I choose to believe it 's the latter , although it probably springs from the fact that just about everyone out here , including the A**[CP_START]** 's**[CP]** and**[CC]** Giants**[CP]** , is originally from somewhere else .

Predictions:

detector 2: I choose to believe it 's the latter , although it probably springs from the fact that just about everyone out here , including the**[CP_START]** A**[CP]** 's**[CP]** and**[CC]** Giants**[CP]** , is originally from somewhere else .

Рис. 13. Включение лишнего артикля «the» в извлечённую маску.

Ground truths:

1. On**[CP_START]** the**[CP]** screens**[CP]** ,**[CP]** only**[CP]** two**[CP]** forlorn**[CP]** blue**[CP]** figures**[CP]** remained**[CP]** ,**[OTHERS]** but**[CC]** the**[CP]** index**[CP]** had**[CP]** recovered**[CP]** a**[CP]** few**[CP]** points**[CP]** and**[CP]** was**[CP]** off**[CP]** about**[CP]** 140**[CP]** .
2. On the screens , only two forlorn blue figures remained , but the index had**[CP_START]** recovered**[CP]** a**[CP]** few**[CP]** points**[CP]** and**[CC]** was**[CP]** off**[CP]** about**[CP]** 140**[CP]** .

Predictions:

detector 2: On the screens , only**[CP_START]** two**[CP]** forlorn**[CP]** blue**[CP]** figures**[CP]** remained**[CP]** ,**[OTHERS]** but**[CC]** the**[CP]** index**[CP]** had**[CP]** recovered**[CP]** a**[CP]** few**[CP]** points**[CP]** and**[CP]** was**[CP]** off**[CP]** about**[CP]** 140**[CP]** .

detector 4: On the screens , only two forlorn blue figures remained , but the index had**[CP_START]** recovered**[CP]** a**[CP]** few**[CP]** points**[CP]** and**[CC]** was**[CP]** off**[CP]** about**[CP]** 140**[CP]** .

Рис. 14. Ошибочное невключение в извлечённую маску обособленного запятой обстоятельства «On the screens».

Ground truths:

1. Under the terms of the contract , New York-based Gitano has the option to acquire the remaining 50 % of Regatta , a maker of men**[CP_START]** 's**[CP]** and**[CC]** women**[CP]** 's**[CP]** clothes sold primarily in department stores , under certain conditions .

Predictions:

detector 2: Under the terms of the contract , New York-based Gitano has the option to acquire the remaining 50 % of Regatta , a maker of men**[CP_START]** 's**[CP_START]** and**[CC]** women**[CP]** 's**[CP]** clothes sold primarily in department stores , under certain conditions .

Рис. 15. Нахождение некорректной маски с двумя метками `CP_START`.

Ground truths:

1. FMC**[CP_START]** Corp.**[CP]** and**[CC]** Baxter**[CP]** International**[CP]** say unions also wo n't like plant**[CP_START]** relocations**[CP]** and**[CC]** needed**[CP]** restructuring**[CP]** , which means layoffs .

Predictions:

detector 2: FMC**[CP_START]** Corp.**[CP]** and**[CC]** Baxter**[CP]** International**[CP]** say unions also wo n't like plant**[CP]** relocations**[CP]** and**[CC]** needed**[CP]** restructuring**[CP]** , which means layoffs .

Рис. 16. Извлечение некорректной маски, в которой одна из сочинительных связей начинается с метки `CP`, а не `CP_START`.

масок было предложено и апробировано несколько вариантов регуляризации (см. разделы 4.4.1, 4.4.3) и изменение в процедуре пост-обработки (см. раздел 4.5.3). Ещё одна группа подходов направлена на увеличение полноты и точности (см. разделы 4.5.1, 4.4.2).

После изложения всех вариантов модификации DetIE-CA будут обсуждены гиперпараметры модели, представлена сводная таблица с результатами и подведены итоги.

4.3. **Предобработка данных — чанкинг.** Chunking[2] — это задача, заключающаяся в разбиении входного текста на синтаксически связанные группы. Пример такого разбиения приведён на Рис. 17.

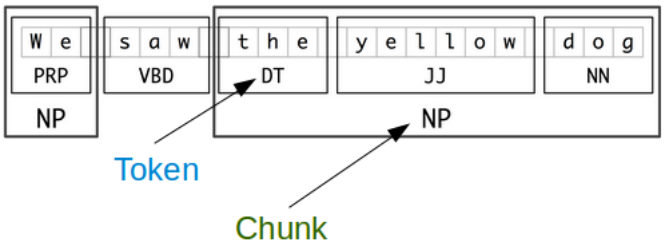

Рис. 17. Пример разделения предложения на chunks.

Первый подход к решению проблемы «масштаба», который был реализован, — обучить 2 модели DetIE-CA: одну на обычной разметке PTB-CA, вторую — на изменённой разметке с выделенными сочинительными связями не на целом предложении, а на отдельных chunks; а затем объединить предсказания этих двух моделей, предварительно удалив дубликаты.

Решение задачи chunking, которое было реализовано, основано на задании списка правил по меткам частей речи входных токенов. Метки части речи (Part-of-speech tagging, POS-теги) токенов из текстов датасета PTB-CA были получены при помощи методов библиотеки `spacy` [1], без использования синтаксического разбора[3].

В английском языке одно простое предложение содержит ровно одну глагольную группу. Соединение простых предложений может происходить с помощью[4]:

- относительных местоимений (that, which, who, whose),
- подчинительных союзов (while, because, although, as, when, until, unless, through, by, since, whenever, if, where, before и т.д.),
- сочинительных союзов (в таком случае обязательно присутствует запятая),
- глагольных структур.

Поэтому можно разбивать сложное предложение на простые части, находя отдельные глагольные группы и ставить «перегородки» в местах, где есть знаки препинания, относительные

[2]Возможные переводы — «синтаксическая сегментация» и «неглубокий парсинг».

[3]Полный синтаксический разбор как отдельный шаг в обучении и предсказании не использовался нами сознательно, так как это привело бы к значительному замедлению работы метода.

[4]Например, см. методические материалы: `https://web.archive.org/web/20220925112942/https://education.nsw.gov.au/teaching-and-learning/student-assessment/smart-teaching-strategies/literacy/writing/stage-3/sentence-structure/writing-complex-sentences`.

местоимения, подчинительные союзы. Исходя из этой идеи, был составлен список правил, задающих “chunk”:

(1) $\texttt{IS\_START}, \texttt{NOT\_VERB}^*, \texttt{VERB\_GROUP}, \texttt{NOT\_VERB}^*, \texttt{SCONJ}$
(2) $\texttt{IS\_START}, \texttt{NOT\_VERB}^*, \texttt{VERB\_GROUP}, \texttt{NOT\_VERB}^*, (\texttt{that} \vee \texttt{which} \vee \texttt{who} \vee \texttt{whose} \vee \texttt{whom})$
(3) $\texttt{IS\_START}, \texttt{NOT\_VERB}^*, \texttt{VERB\_GROUP}, \texttt{NOT\_VERB}^*, \texttt{IS\_PUNCT}$
(4) $\texttt{SCONJ}, \texttt{NOT\_VERB}^*, \texttt{VERB\_GROUP}, \texttt{NOT\_VERB}^*, \texttt{IS\_PUNCT}$
(5) $\texttt{SCONJ}, \texttt{NOT\_VERB}^*, \texttt{VERB\_GROUP}, \texttt{NOT\_VERB}^*, \texttt{SCONJ}$
(6) $\texttt{SCONJ}, \texttt{NOT\_VERB}^*, \texttt{VERB\_GROUP}, \texttt{NOT\_VERB}^*, (\texttt{that} \vee \texttt{which} \vee \texttt{who} \vee \texttt{whose} \vee \texttt{whom})$
(7) $(\texttt{that} \vee \texttt{which} \vee \texttt{who} \vee \texttt{whose} \vee \texttt{whom}), \texttt{NOT\_VERB}^*, \texttt{VERB\_GROUP}, \texttt{NOT\_VERB}^*, \texttt{SCONJ}$
(8) $\texttt{IS\_PUNCT}, \texttt{CCONJ}, \texttt{NOT\_VERB}^*, \texttt{VERB\_GROUP}, \texttt{NOT\_VERB}^*, \texttt{IS\_PUNCT}$

Введены следующие обозначения:

- * обозначает повторение ≥ 0 раз
- IS_START обозначает метку начального токена предложения
- NOT_VERB — токены, не являющиеся глаголами
- VERB_GROUP — глагольная группа, которая также задаётся правилом:

$$(\texttt{VERB} \vee \texttt{PART} \vee \texttt{ADP} \vee \texttt{AUX})^*, \texttt{VERB}, (\texttt{VERB} \vee \texttt{PART} \vee \texttt{ADP} \vee \texttt{AUX})^*,$$

  где VERB — глагол, PART — частица, ADP — предлог, AUX — вспомогательные части речи (например, “is”, “has (done)”, “will (do)”, “should (do)”)
- SCONJ — подчинительные союзы
- IS_PUNCT — метка знака препинания, за исключением дефиса и процента
- CCONJ — сочинительные союзы

После выделения chunks с помощью перечисленных выше правил идёт этап пост-обработки, включающий в себя дополнительные шаги:

(1) убрать «немаксимальные» chunks, то есть удалить те, которые являются подстрокой одного из извлечённых отрезков;
(2) отдельно добавить «отрезки», которые не покрылись извлечёнными chunks, то есть дополнительным chunk будет оставшаяся часть входного предложения.

Пример разделения входного предложения из валидационной части PTB-CA на chunks с помощью правил и пост-обработки, описанных выше, приведён на Рис. 18.

4.4. **Архитектурные изменения и регуляризация.** Вторая группа экспериментов была связана с изменением устройства модели, в том числе с добавлением различных регуляризаторов в функцию потерь.

4.4.1. *«Лингвистическое ограничение».* Эта добавка в функцию потерь «поощряет» модель за отнесение каждого соединительного союза (“and”, “nor”, “or” и “but”) в тексте к хотя бы

```
INPUT TEXT:
 Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new s
avings-and-loan bailout agency can raise capital , creating another potential obstacle to the government 's sale of
sick thrifts .

POSTPROCESSED CHUNKS:
1. Influential members of the House Ways and Means Committee introduced legislation that
2. that would restrict how
3. how the new savings-and-loan bailout agency can raise capital ,
4. creating another potential obstacle to the government 's sale of sick thrifts .
```

Рис. 18. Разбиение на chunks при помощи описанных правил и дальшейшей пост-обработки.

одной из извлечённых структур входного предложения в качестве метки `CC`. Идейно этот регуляризатор похож на регуляризатор POSC из модели OpenIE6, описанной в разделе 1.2.3.

Формула добавки:

$$\theta = \sum_{t=1}^{T} x_t^{imp} \cdot Pr_t$$

$$Pr_t = 1 - \max_{n=1}^{N} p_{\texttt{CC},t,n},$$

где $p_{\texttt{CC},t,n}$ — предсказанная моделью вероятность по классу `CC` у токена t в маске n; T — число токенов в маске, N — число извлекаемых масок; x_t^{imp} — индикатор интересующих токенов, он равен 1, если токен t равен “and”, “nor”, “or” или “but”, а также $t \neq 0$ (для того, чтобы рассматривать связи только внутри данного на входе текста), в ином случае равен 0.

4.4.2. *«Орто-регуляризация».* Данный тип регуляризации отвечает за разнообразие внутренних векторных представлений модели. Аналогичный регуляризатор используется в модели Attention-based Aspect Extraction (ABAE) [9], которая решает задачу выделения аспектов без учителя (Unsupervised Aspect Extraction), для обеспечения разнообразия эмбеддингов аспектов.

В ходе работы было рассмотрено два варианта «орто-регуляризации»:

- для векторных представлений токенов, извлечённых с помощью BERT,
- для ненормализованных предсказаний на выходах отдельных детекторов.

Первый вариант «орто-регуляризатора» отвечает за разнообразие эмбеддингов на выходе последнего слоя модели BERT. Это спорное «пожелание», однако, мы предположили, что при увеличении разнообразия эмбеддингов модели «легче» разделить разные классы меток. По нашему замыслу, таким образом можно влиять на точность предсказаний.

Пусть M — матрица полученных векторных представлений, отнормированная вдоль строк так, чтобы длины векторов (по евклидовой норме) стали равны 1. Размерность матрицы M равна $T \times dim_{emb}$, где T — это количество токенов во входной последовательности, dim_{emb} — размерность эмбеддингов. Тогда описанный регуляризатор вычисляется следующим образом:

$$L_{ortho_1} = ||M \cdot M^T - I^{(1)}||,$$

где $I^{(1)}$ — единичная матрица размера $T \times T$, $||.||$ — норма Фробениуса. Разнообразие эмбеддингов обеспечивается за счёт «поощрения» ортогональности между строками матрицы M.

Второй рассмотренный «орто-регуляризатор» накладывает «требование разнообразия» на усреднения предсказаний, полученных на выходе полносвязного слоя (до применения преобразования `softmax`) для разных извлечённых масок сочинительных структур. Пусть D — матрица размера $N \times C$, где N — число извлекаемых масок, C — количество классов. Эта матрица в каждой строке содержит усреднённое по токенам входного предложения предсказание модели. Матрица D также предварительно отнормирована вдоль строк. Данный вариант «орто-регуляризатора» вычисляется по формуле:

$$L_{ortho_2} = ||D \cdot D^T - I^{(2)}||,$$

где $I^{(2)}$ — единичная матрица размера $N \times N$. При использовании этого регуляризатора мы исходили из предположения, что, требуя, чтобы усреднения предсказаний были ортогональны друг другу, мы можем разнообразить «роли» «детекторов», выделяющих сочинительные структуры и влиять таким образом на полноту в целом.

4.4.3. *Ограничение на порядок извлекаемых меток.* Этот регуляризатор отвечает за корректировку порядка извлечённых моделью меток, а именно он «штрафует» модель за высокую вероятность встретить следующие пары подряд идущих меток:

- перед меткой `CP_START` стоит метка не `NONE`,
- `NONE`, затем метка, отличная от `CP_START` и `NONE`.

Для каждой извлечённой маски вычисляется «штраф», зависящий только от вероятностей, полученных моделью для данной маски:

$$L_{structure} = \sum_{t=1}^{T-1} \left((1 - p_{t-1,\texttt{NONE}})\ p_{t,\texttt{CP\_START}} + p_{t-1,\texttt{NONE}}\ (1 - p_{t,\texttt{CP\_START}} - p_{t,\texttt{NONE}}) \right),$$

где $p_{t,c}$ — предсказанная моделью вероятность того, что в текущей извлечённой маске токен t относится к классу c.

Итоговая формула добавки в функцию потерь получается последовательным усреднением по всем маскам, по всем парам подряд идущих токенов, по всем примерам в батче.

4.4.4. *Последовательные свёрточные слои.* В архитектуре исходной модели DetIE-CA за BERT следует только полносвязный слой. Использование на этом этапе не только полносвязного слоя, но и свёрточных, могло бы, по нашему замыслу, позволить модели ориентироваться на большее количество токенов вокруг текущего при формировании итоговых предсказаний.

Первый вариант изменения архитектуры модели с добавлением свёрточных слоёв, который был рассмотрен, заключался в замене полносвязного слоя в конце на 3 свёрточных слоя с

размерами ядра свёртки 7, 5 и 3 соответственно, идущих друг за другом, и полносвязный слой.

4.4.5. *Свёрточные слои «в ширину».* Второй вариант добавления свёрточных слоёв в архитектуру: соединить добавляемые свёрточные слои не последовательно, а параллельно. Похожим образом свёрточные слои используются в архитектуре, которую иногда называют KimCNN [12] и которая решает задачу классификации предложений.

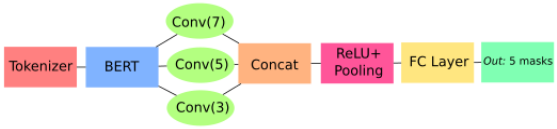

Рис. 19. Архитектура модели со свёрточными слоями «в ширину».

Однако в KimCNN после параллельного соединения свёрточных слоёв следует их агрегация с помощью max pooling вдоль временной оси, то есть от первого до последнего токена входного предложения. В случае задачи CA агрегировать вдоль временной оси нельзя, так как для каждого отдельного токена должна предсказываться своя метка. Поэтому в ходе этого эксперимента был придуман другой подход: либо последовательная конкатенация результатов свёрток с разным размером ядра (вновь были рассмотрены размеры ядра 7, 5 и 3), либо циклическая, то есть сначала идут результаты свёртки для первого токена для каждого из размеров ядер свёртки, затем для второго и т.д.; после конкатенации следует слой pooling (либо максимум, либо усреднение) и полносвязный слой (см. Рис. 19).

4.4.6. *Модель ConvBERT в качестве кодировщика.* Ещё один подход к решению проблемы «масштаба» — заменить кодировщик в архитектуре на ConvBERT [11]. Данная модель включает в себя новый тип «голов внимания», в котором используются свёртки. Модель ConvBERT более эффективна в обучении локальному и глобальному контексту. Авторы статьи показывают, что ConvBERT показывает лучшие результаты, чем обычный BERT, в поиске парафразов, вопросно-ответных задачах и в проверке грамматической правильности предложений.

4.5. **Изменение процедуры пост-обработки.** Третья группа экспериментов была связана с поиском наилучшего варианта агрегации и дальнейшей процедуры пост-обработки предсказаний модели.

4.5.1. *Агрегация с помощью Beam Search.* Как было сказано в главе 3.3, в изначальной версии DetIE-CA используется «жадная» агрегация, то есть в качестве итоговой метки класса для каждого токена внутри одной маски рассматривается класс с наибольшей предсказанной моделью вероятностью.

С целью повышения полноты был рассмотрен другой вариант агрегации с помощью *подобия* алгоритма лучевого поиска (Beam Search[5]). Данный алгоритм был впервые описан в статье [22].

У реализованного варианта агрегации есть несколько гиперпараметров:

- k — максимальное количество извлекаемых «путей» для одного тензора вероятностей,
- $thres$ — ограничение сверху на отношение первых двух максимумов (см. далее) среди вероятностей.

Алгоритм агрегации следующий:

(1) До индекса idx по токенам входного предложения выбираются «жадные» метки,
(2) idx выбирается таким, что отношение первого максимума вероятностей ко второму минимально,
(3) если данное отношение $\leq thres$, то начинается ветвление на k различных «путей». На каждом шаге, начиная с индекса idx, поддерживается массив из k наибольших по произведению оценок вероятностей в маске «путей». В итоге для одного тензора вероятностей строится k различных масок сочинительных структур.
Если же отношение $> thres$, то качестве предсказания выдаётся только одна маска, полученная с помощью «жадной» агрегации.

Сравнение с некоторым граничным значением требуется для того, чтобы не начинать ветвление, если первый и второй максимум по оценкам вероятностей недостаточно близки. Заметим, что рассмотренные произведения нельзя интерпретировать как вероятности последовательностей, поэтому формальная корректность метода — под вопросом (см. также далее); тем не менее, использование такой эвристики позволило повысить качество работы DetIE-CA.

Лучшие значения параметров $k = 2$ и $thres = 4.0$ были подобраны по оценкам качества на валидационной части датасета.

4.5.2. *Модель для уточнения условных вероятностей.* Обычно в области обработки естественного языка алгоритм Beam Search применяют для авторегрессионных моделей генерации. Пусть y_i — предсказанная такой моделью метка токена i, X — входная последовательность токенов. В таком случае при перемножении условных вероятностей $p_i = p(y_i|X, y_1, .., y_{i-1})$, которые будут на выходе для каждого входного токена, получается:

$$\prod_{i=1}^{n} p_i = \frac{p(X, y_1, y_2, .., y_n)}{p(X)}$$

Для одной и той же входной последовательности X можно считать вероятность $p(X)$ в знаменателе константой, поэтому в подобных моделях Beam Search рассматривает несколько первых максимумов по совместной вероятности токенов и меток во входной и выходной последовательностях.

[5]В литературе встречаются переводы «метод луча», «лучевой поиск» и другие.

В случае модели DetIE-CA вероятности на выходе получаются не в авторегрессивной манере, и каждая из них зависит не только от входной последовательности токенов и предыдущих предсказанных меток, но и от последующих меток тоже. Поэтому, с математической точки зрения, использовать в этом случае агрегацию Beam Search не совсем корректно.

В качестве более точной эвристики для Beam Search был придуман следующий подход: обучить модель, аналогичную DetIE-CA, предсказывающую для каждой пары подряд идущих токенов парные метки классов (всего классов будет $6 \cdot 6 = 36$). Тогда для каждого индекса по токенам будет известна оценка совместной вероятности двух подряд идущих меток. Используя также обычный вариант модели, предсказывающий метки для каждого токена, можно поделить совместную вероятность двух подряд идущих токенов на вероятность одного токена и получить оценку вероятности текущего токена при условии одного предыдущего.

4.5.3. *Корректировка извлечённых масок на правилах.* Этот подход направлен на корректировку масок сочинительных структур, полученных после этапа агрегации.

Было рассмотрено два варианта правил для данного этапа пост-обработки:

(1) подряд идущие метки `CP_START` заменяются на `CP_START`, `CP`, `CP`, ..., `CP`;
(2) прерывания в одной сочинительной структуре, то есть все метки `NONE`, которые находятся между ближайшей слева меткой `CP_START` и меткой, отличной от `CP_START` и `NONE`, справа, заменяются на `CP`.

4.6. **Схема обучения и подбора гиперпараметров.** Обучение модели происходило на тренировочной части датасета PTB-CA, а валидация — на валидационной (то есть было использовано то же разбиение, что и в работе [14]). Все эксперименты проводились на графической карте NVIDIA GeForce GTX 1080 Ti.

Во всех экспериментах, кроме "ConvBERT", описанного в разделе 4.4.6, использовался кодировщик `bert-base-multilingual-cased` из семейства `bert-base`, так как с `bert-large` не удавалось эффективно проводить эксперименты из-за большого количества требуемых вычислительных ресурсов.

Гиперпараметры подбирались с помощью сравнения оценок качества на валидационной части датасета для базовой модели, обученной на тренировочных данных с разным набором гиперпараметров, с фиксированным числом эпох, равным 370 (причём сохраняются веса модели на лучшей по F1-мере на валидации эпохе, по аналогии с обучением DetIE). Количество эпох подбиралось, исходя из графиков изменения F1-меры на тренировочной и валидационной частях: на 370 эпохе график выходит на «плато».

Перебор осуществлялся в следующих диапазонах: размер «батча» (подвыборки, на которой вычисляется и усредняется функция потерь для обновления параметров методом обратного распространения ошибки) варьировался от 32 до 128, количество предсказываемых моделью масок — от 3 до 6 (в экспериментах с ещё большим числом «детекторов» с ростом их числа

Таблица 2. Сравнение оценок качества работы моделей на валидационной части датасета при разных значениях гиперпараметров для фиксированного числа эпох, равного 370.

Гиперпараметры							Качество		
Размер батча	**IoU** с исп. фона или без	Кол-во масок	Кол-во обучаемых слоёв BERT	Оптим.	lr	wd	**P**	**R**	**F1**
128	Без	6	4	Adam	10^{-4}	10^{-6}	0.828	0.816	0.822
128	Без	3	4	Adam	10^{-4}	10^{-6}	0.832	0.804	0.818
128	**Без**	**5**	**4**	**Adam**	$\mathbf{10^{-4}}$	$\mathbf{10^{-6}}$	**0.832**	**0.812**	**0.822**
32	Без	5	4	Adam	10^{-4}	10^{-6}	0.826	0.803	0.814
128	Без	6	4	AdamW	10^{-4}	10^{-6}	0.815	0.796	0.805
128	Без	5	2	Adam	10^{-4}	10^{-6}	0.811	0.786	0.798
128	С	5	4	Adam	10^{-4}	10^{-6}	0.822	0.805	0.814
128	Без	5	4	Adam	10^{-5}	10^{-6}	0.813	0.793	0.803
128	Без	5	4	Adam	10^{-4}	10^{-5}	0.814	0.790	0.801

оценки качества скоро снижались), количество «размороженных» слоёв в кодировщике — от 2 до 4, с учётом токенов с меткой фонового класса при подсчёте **IoU** и без, оптимизаторы Adam [13] и AdamW [18] с различными значениями темпа обучения (learning rate), определяющего размер шага на каждой итерации, и коэффициента при L_2-регуляризаторе (сокращение весов, weight decay). L_2-регуляризация представляет собой добавку к функции потерь с коэффициентом weight decay, которая «штрафует» модель за слишком высокие значения L_2-нормы весов. В качестве планировщика темпа обучения (learning rate scheduler) используется ExponentialLR, который каждую эпоху домножает текущее значение темпа обучения на некоторую константу, и, таким образом, темп обучения экспоненциально растёт. Такой же планировщик темпа обучения использовался в работе [28].

Наибольшего значение F1-меры на валидации в предварительных экспериментах с базовой моделью удалось добиться при следующих гиперпараметрах: размер «батча» = 128, без учёта токенов с меткой фонового класса при подсчёте **IoU**, количество предсказываемых масок = 5, количество «размороженных» слоёв в кодировщике = 4, оптимизатор Adam с learning_rate=10^{-4} и weight_decay=10^{-6}.

В Таблицу 2 включены несколько пар оценок качества для подходов, отличающихся лишь одним гиперпараметром, для демонстрации степени его влияния. Оценка с лучшим найденным набором значений гиперпараметров также включена.

На основании предварительных экспериментов лучший набор гиперпараметров использовался в дальнейшем как базовый. При изменениях архитектуры, о которых пойдёт речь дальше, набор лучших гиперпараметров может, вне сомнения, быть другим. Однако дополнительные эксперименты со всеми из потенциальных улучшений моделей лишь подтверждали, что лишь подтверждали, что выбранный набор гиперпараметров предпочтительный.

4.7. **Результаты экспериментов.** Подходы, описанные в разделах 4.3, 4.4 и 4.5, продемонстрировали результаты, представленные в Таблице 3. Лучшее качество, которого удалось добиться, наблюдается при применении подхода Beam Search (см. 4.5.1): на тестовой части датасета PTB-CA изначальная модель DetIE-CA с дальнейшей агрегацией с помощью Beam Search достигает точности, равной 0.856, полноты — 0.833 и **F1-меры**— 0.844.

Таблица 3: Результаты экспериментов с разными подходами, оценка качества на тестовой части датасета PTB-CA (размер батча во всех экспериментах равен 128; в качестве оптимизатора использовался Adam с learning_rate=10^{-4} и weight_decay=10^{-6}). **P**, **R**, **F1** — точность, полнота и F1-мера соответственно. Полужирным шрифтом выделены строки с экспериментами, где F1-мера превосходит 0.84.

Эксперимент	Кодировщик	**P**	**R**	**F1**
Изначальная DetIE-CA	bert-base (multiling)	0.840	0.797	**0.818**
Изначальная DetIE-CA с большим числом эпох	bert-base (multiling)	0.845	0.802	**0.823**
Изначальная DetIE-CA с заменой тегов	**bert-base (multiling)**	**0.851**	**0.830**	**0.840**
Добавление регуляризатора, задающего лингвистическое ограничение, с весом 0.1	bert-base (multiling)	0.780	0.701	**0.738**
Добавление орто-регуляризации векторных представлений на выходе BERT, с весом 0.1	bert-base (multiling)	0.846	0.794	**0.819**
Добавление орто-регуляризации выходов полносвязного слоя, с весом 0.1	bert-base (multiling)	0.843	0.814	**0.828**
Добавление регуляризатора, задающего порядок меток, с весом 0.1	bert-base (multiling)	0.843	0.826	**0.834**

Добавление регуляризатора, задающего порядок меток, с весом 0.3	bert-base (multiling)	0.839	0.820	**0.829**
Добавление регуляризатора, задающего порядок меток, с весом 1.0	bert-base (multiling)	0.848	0.824	**0.836**
Chunking, объединение двух моделей	bert-base (multiling)	0.843	0.803	**0.823**
Добавление последовательных свёрточных слоёв	bert-base (multiling)	0.820	0.765	**0.792**
Добавление свёрточных слоёв в ширину (без pooling + последовательная конкатенация)	bert-base (multiling)	0.818	0.764	**0.823**
Добавление свёрточных слоёв в ширину (avg pooling + последовательная конкатенация)	bert-base (multiling)	0.836	0.769	**0.801**
Добавление свёрточных слоёв в ширину (avg pooling + циклическая конкатенация)	bert-base (multiling)	0.802	0.732	**0.765**
Добавление свёрточных слоёв в ширину (max pooling + последовательная конкатенация)	bert-base (multiling)	0.809	0.752	**0.779**
Добавление свёрточных слоёв в ширину (max pooling + циклическая конкатенация)	bert-base (multiling)	0.815	0.758	**0.786**
ConvBERT	YituTech/ conv-bert-base	0.836	0.795	**0.815**
Beam Search	**bert-base (multiling)**	**0.856**	**0.833**	**0.844**
Объединение обычной модели и модели «на парах»	bert-base (multiling)	0.873	0.774	**0.820**
Пост-обработка: CP_STARTS =>CP_START, CP, ...	**bert-base (multiling)**	**0.851**	**0.830**	**0.840**
Пост-обработка: заполнение прерываний	bert-base (multiling)	0.846	0.826	**0.836**
Пост-обработка: заполнение прерываний, CP_STARTS =>CP_START, CP, ...	bert-base (multiling)	0.848	0.828	**0.838**
Beam Search + Пост-обработка: CP_STARTS =>CP_START, CP, ...	**bert-base (multiling)**	**0.856**	**0.833**	**0.844**

Полученный результат сопоставим по качеству с актуальными современными подходами.

Идея добавления регуляризатора, задающего «лингвистическое ограничение» (см. 4.4.1), лишь понизила качество предсказаний. Скорее всего, это связано с тем, что данная добавка «побуждает» модель предсказывать в качестве метки `CC` только “and”, “nor”, “or” и “but”. Остальные предложенные методы регуляризации, предложенные в разделах 4.4.2, 4.4.3, также не дали прироста качества.

Подходы, направленные на решение проблемы «масштаба» и описанные в разделах 4.3, 4.4.4, 4.4.5 и 4.4.6, не дали улучшений качества работы. В случае с чанкингом (см. 4.3), скорее всего, это связано с неидеальностью выделения chunks с помощью предварительно заданных правил. А в подходах с добавлением свёрточных слоёв в изначальную архитектуру, возможно, получаются слишком «перегруженные» модели.

Как уже было сказано выше, подход с изменением процедуры агрегации наподобие алгоритма Beam Search, описанный в разделе 4.5.1, дал существенный прирост. Подход, направленный на уточнение условных вероятностей (см. раздел 4.5.2), лишь понизил качество работы. Добавление этапа пост-обработки на правилах, предложенное в разделе 4.5.3, никак не меняет значения оценок качества, однако, для возможных последующих приложений, вероятно, следует применять именно комбинацию «Beam Search + пост-обработка», так как благодаря последнему этапу она даёт дополнительные гарантии качества.

В Таблице 4 представлены оценки качества на тестовой части датасета PTB-CA для достигающей в настоящий момент наилучшего качества модели IGL-CA (см. параграф 1.2.3), второй модели по качеству Teranishi-19 [27] (см. раздел 1.1.1) и модели, описанной в настоящей квалификационной работе. Также модель DetIE-CA существенно превосходит актуальные подходы по производительности. Оценки скорости работы DetIE-CA и IGL-CA на «инференсе» (в режиме предсказания) также приведены в Табл. 4. Teranishi-19 заведомо медленнее остальных подходов, так как, во-первых, в её основной версии из статьи [27] используется рекуррентная архитектура, которая, в отличие от BERT, обрабатывает токены входного предложения последовательно. Безусловно, существуют рекуррентные подходы, которые на этапе предсказания могут работать быстрее, чем «тяжеловесная» архитектура из нескольких кодировщиков Transformer, но, во-вторых, даже если заменить в архитектуре Teranishi-19 «рекуррентную часть» на BERT, как это было сделано в статье [14], данная модель будет гарантированно работать медленнее, поскольку в ней используется алгоритм Кока-Янгера-Касами [24] для построения дерева разбора на основе особой грамматики, который имеет вычислительную сложность $\mathcal{O}(n^3 \cdot |G|)$, где n — количество токенов во входном предложении, а $|G|$ — размер грамматики в нормальной форме Хомского [3].

5. Заключение

В данной работе были исследованы различные подходы к улучшению нейросетевой архитектуры модели для решения задачи выделения сочинительных связей в предложениях на

Таблица 4. Сравнение качества и производительности с актуальными современными подходами. Уступая IGL-CA (`bert-base-cased`) 0.5% в F1-мере, предлагаемая модель позволяет решать целевую задачу, обрабатывая в 3.45 раз больше предложений в секунду в условиях аналогичной вычислительной инфраструктуры.

Модель	Точность	Полнота	**F1-мера**	Скорость «на инференсе» (предл./сек.)
DetIE-CA + Beam Search (`bert-base-multilingual-cased`)	0.856	0.833	0.844	**558**
IGL-CA (`bert-base-cased`)	0.863	0.836	**0.849**	162
Teranishi-19 (BiLSTM)	0.753	0.756	0.755	-
Teranishi-19 (`bert-base-cased`)	0.831	0.832	0.831	-

английском языке. Существенного прироста в качестве удалось достичь с помощью разработанного алгоритма агрегации наподобие Beam Search, значительное же число апробированных подходов показали себя бесперспективными на основе экспериментов на доступных нам вычислительных ресурсах.

Также было проведено сравнение качества и производительности полученной модели с актуальными современными подходами к задаче, которое показало, что полученный в работе результат сопоставим по качеству с лучшими современными решениями и значительно превосходит их по скорости работы на инференсе, что позволяет говорить о возможности более эффективного использования нейросетевых подходов к извлечению сочинительных связей.

В дальнейшем, при наличии доступа к инфраструктуре, позволяющей эффективно проводить большее число экспериментов с моделями на основе значительно большего числа параметров, имеет смысл апробировать предложенный подход, взяв за базовую языковую модель из семейства `bert-large`. Пример IGL-CA [14] позволяет предполагать, что это может существенно улучшить качество предсказаний.

Список литературы

[1] Spacy. `https://github.com/explosion/spaCy`, 2015.

[2] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. January 2015. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

[3] Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959.

[4] Jack Valmadre Christoph Heindl. py-lapsolver. `https://github.com/cheind/py-lapsolver`, 2018.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018.

[6] Jessica Ficler and Yoav Goldberg. Coordination annotation extension in the penn tree bank. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 834–842, 2016.

[7] Jessica Ficler and Yoav Goldberg. A neural network for coordination boundary prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 23–32, 2016.

[8] Shelley Gupta, Archana Singh, and Vivek Kumar. Emoji, text, and sentiment polarity detection using natural language processing. *Information*, 14(4), 2023.

[9] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 388–397, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Convbert: Improving BERT with span-based dynamic convolution. *CoRR*, abs/2008.02496, 2020.

[12] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, Soumen Chakrabarti, et al. Openie6: Iterative grid labeling and coordination analysis for open information extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3748–3761, 2020.

[15] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[16] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*. OpenReview.net, 2020.

[17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. cite arxiv:1907.11692.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[19] Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Using Large Corpora*, 273, 1994.

[20] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018. cite arxiv:1802.05365Comment: NAACL 2018. Originally posted to openreview 27 Oct 2017. v2 updated for NAACL camera ready.

[21] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[22] Raj Reddy. Speech understanding systems: A summary of results of the five-year research effort. Carnegie Mellon University, 1977.

[23] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015. cite arxiv:1506.02640.

[24] Itiroo Sakai. Syntax in universal translation. In *Proceedings 1961 International Conference on Machine Translation of Languages and Applied Language Analysis, Her Majesty's Stationery Office, London*, pages 593–608, 1962.

[25] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

[26] Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. Coordination boundary identification with similarity and replaceability. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 264–272, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing.

[27] Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. Decomposed local models for coordinate structure parsing. In *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)*, pages 3394–3403, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[28] Michael Vasilkovsky, Anton Alekseev, Valentin Malykh, Ilya Shenbin, Elena Tutubalina, Dmitriy Salikhov, Mikhail Stepnov, Andrey Chertok, and Sergey Nikolenko. Detie: Multilingual open information extraction inspired by object detection. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, 2022.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[30] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy, July 2019. Association for Computational Linguistics.

[31] Розенталь Д.Э. и Теленкова М.А. Словарь-справочник лингвистических терминов, издание третье. 1985.