ДОКЛАДЫ РОССИЙСКОЙ АКАДЕМИИ НАУК. МАТЕМАТИКА, ИНФОРМАТИКА, ПРОЦЕССЫ УПРАВЛЕНИЯ, 2023, том 514, № 2, с. 289–296

УДК 004.8

НЕЙРОСЕТЕВЫЕ МЕТОДЫ ВЫДЕЛЕНИЯ СОЧИНИТЕЛЬНЫХ СВЯЗЕЙ

© 2023 г. А. И. Пределина[1,*], С. Ю. Дуликов[2], А. М. Алексеев[1,3]

Представлено академиком РАН А.Л. Семеновым
Поступило 04.09.2023 г.
После доработки 15.09.2023 г.
Принято к публикации 18.10.2023 г.

Данная работа посвящена решению задачи выделения сочинительных связей в предложениях на английском языке нейросетевыми методами. Решение этой задачи позволяет устанавливать потенциально ценные связи и отношения между определенными частями предложения; в том числе поэтому выделение сочинительных связей – важный инструмент предобработки текстов. В настоящей работе апробирован ряд идей способов решения задачи в рамках подхода “одностадийных предсказаний” (one-stage detectors). Полученный в работе результат сопоставим по качеству и более чем в 3 раза превосходит по производительности наиболее современные методы выделения сочинительных связей.

Ключевые слова: обработка естественного языка, выделение сочинительных связей, машинное обучение, нейросетевые модели

DOI: 10.31857/S2686954323601975, **EDN:** HXUFSN

1. ВВЕДЕНИЕ

Задача выделения сочинительных связей. Извлечение информации из текстов является одним из основных направлений в области обработки естественного языка. Есть целый ряд задач, которые решаются исследователями в рамках этого направления. Одна из них – задача выделения сочинительных связей (Coordination Analysis, CA), состоящая в том, чтобы научиться находить внутри предложений синтаксические структуры, соединяющие грамматически равноправные части предложений [14]. 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).

Выделение сочинительных связей является важным методом предобработки текстов. Так, в работах [7, 13] показано, что этот этап подготовки значительно влияет на итог работы соответствующих методов.

[1]*Санкт-Петербургский государственный университет, Санкт-Петербург, Россия*
[2]*“Яндекс”, Москва, Россия*
[3]*Санкт-Петербургское отделение Математического института им. В.А. Стеклова РАН, Санкт-Петербург, Россия*
*****E-mail: a-predelina@mail.ru*

Решение задачи CA методами машинного обучения. Первые работы, посвященные решению задачи CA с помощью нейронных сетей, основаны 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 ее составных частей предложения остаются осмысленными. **Ficler и Goldberg** [3] в своей работе объединили синтаксический парсер и несколько рекуррентных нейронных сетей для подсчета характеристик, отражающих наличие или отсутствие описанных выше свойств. В работе [11] (“**Teranishi-17**”) устраняется один из недостатков предыдущего подхода – зависимость от внешнего парсера: авторы статьи решают задачу CA, используя только токены предложения и их частеречные метки. В 2019 г. теми же авторами была опубликована статья [12] (“**Teranishi-19**”), в которой идеи из “**Teranishi-17**” обобщались и улучшались.

В последнее время наиболее перспективные подходы к CA состоят в использовании моделей, решающих задачи, сходные с извлечением информации (Information extraction, IE). Одна из разновидностей задачи, Open Information Ex-

Рис. 1. Пример предложения с сочинительной связью.

Рис. 2. Пример предложения с сочинительной связью и соответствующей разметкой.

traction, 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” или “фон”. Аналогично можно сформулировать и задачу CA: для входного предложения модель должна научиться находить набор масок, где каждый токен помечен одним из 6 классов: `CP_START` – токен, с которого начинается структура; `CP` – элементы, соединяемые союзом, `CC` – союз; `SEP` – разделители разных частей структуры, например, запятые, `OTHERS` – все, что не относится к категориям выше, но присутствует в структуре; `NONE` – не относящиеся к структуре слова. Так, пример, приведенный выше, будет размечен как на рис. 2.

Достигающая в настоящий момент лучшего качества модель IGL-CA из работы, посвященной **OpenIE6** [7], решает задачу CA, используя подход, описанный выше. Авторы подготовили новую модель для решения задачи OpenIE, а затем изменили количество классов с 4 до 6 и применили точно такой же подход к задаче CA.

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. Для достижения цели было решено адаптировать модель DetIE [13], провести анализ ошибок, в том числе на основе которого выдвинуть предположения, что можно изменить в исходных архитектуре и процедуре подготовки данных для улучшения результатов.

DetIE – модель для решения задачи OpenIE, вдохновленная идеей одностадийных детекторов из компьютерного зрения. Преимуществами модели являются быстрый inference (предсказание), а также данный подход на нескольких наборах данных для оценки качества OpenIE продемонстрировал качество более высокое, чем OpenIE6.

2. МОДЕЛЬ

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

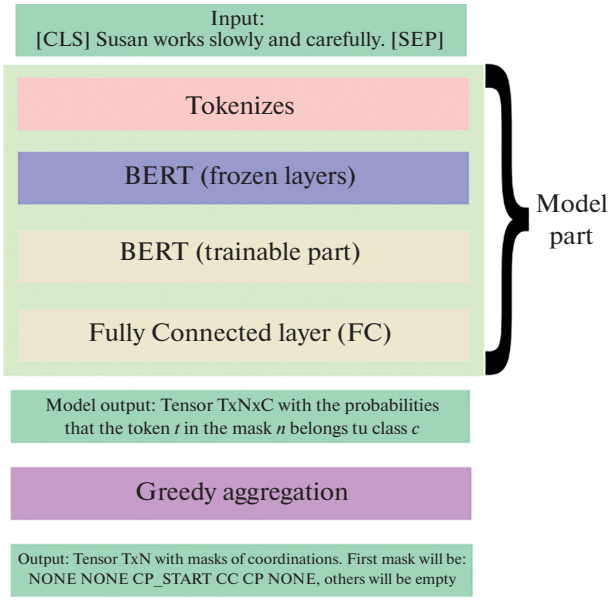

Рис. 3. Основные этапы работы DetIE-CA.

Далее идет несколько этапов, связанных с архитектурой модели. А именно – сначала предложение разделяется на токены, затем извлекаются эмбеддинги с помощью предобученной модели BERT (с несколькими последними “размороженными” слоями для возможности дообучить модель под задачу). Далее эмбеддинги подаются на вход полносвязному слою, на выходе которого 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.

Следующий этап – агрегация тензора вероятностей в итоговый набор масок. В DetIE итоговая метка класса для каждого токена t в каждой маске n вычисляется “жадно”, т.е. выбирается класс c, для которого вероятность p_{tnc} наибольшая. Таким образом, на выходе получается тензор, заполненный номерами классов, размера (T, N).

Этапы, связанные с архитектурой модели и агрегацией меток, отражены на рис. 3.

В ходе работы были рассмотрены различные изменения в начальной архитектуре, продиктованные выявленными основными сценариями ошибок, а именно: “чанкинг”, добавление в мо-

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

	Точность	Полнота	F1-мера
Изначальная DetIE-CA	0.840	0.797	**0.818**
Замена тегов -LRB-, -RRB-, -LCB-, -RCB- на скобки	0.851	0.830	**0.840**

дель сверточных слоев, несколько вариантов регуляризации, изменения в процедурах пост-обработки и агрегации (см. ниже).

3. ДАННЫЕ И ОЦЕНКА КАЧЕСТВА

3.1. Данные

Для обучения и тестирования используется набор данных Penn Treebank CA (PTB-CA; он же использовался в работе OpenIE6 [7]). 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.

3.2. Оценка качества

Был подготовлен и отлажен программный код для оценки решений, аналогичный использованному в OpenIE6 [7], чтобы иметь возможность сравнивать качество работы моделей. Качество работы модели оценивается на тестовой части датасета путем вычисления точности, полноты и F1-меры.

4. ЭКСПЕРИМЕНТЫ

4.1. Изначальное качество работы и процедура предобработки данных

После обучения изначальной модели DetIE-CA на датасете PTB-CA значения оценок качества на тестовой части достигли значений в первой строке табл. 1.

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, и после обучения заново, результаты достигли значений в последней строке табл. 1. Данный результат использовался как “отправная точка”.

4.2. Анализ ошибок

Затем был проведен анализ ошибок модели на валидационной части датасета PTB-CA. Было выявлено два основных сценария ошибок: (1) проблема “масштаба”, (2) некорректность извлеченной маски.

В первом из них модель не находит структуры на “высоком уровне”: соединение простых предложений в составе сложного; также структуры, включающие большое количество второстепенных членов предложения. Также сюда можно отнести случаи, когда модель не “обнаруживает” структуры “низкого уровня”, соединяющие однородные члены предложения.

Во втором сценарии модель извлекает **некорректную маску** сочинительной связи: например, добавление лишних артиклей или невключение элементов, отделенных запятыми. Другой вариант – извлечение неправильной, с точки зрения постановки задачи, маски: например, для каждой структуры метка `CP_START` должна быть ровно одна и должна соответствовать начальному токену структуры.

4.3. Предобработка данных – чанкинг

Chunking – это задача, заключающаяся в разбиении входного текста на синтаксически связанные группы.

Первый подход к решению проблемы “масштаба”, который был реализован, – обучить 2 модели DetIE-CA: одну на обычной разметке PTB-CA, вторую – на измененной 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 основано на задании списка правил по меткам частей речи входных токенов. Метки части речи были получены при помощи методов библиотеки `spacy` [1], без использования синтаксического разбора.

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

- относительных местоимений (that, which, who, whose),
- подчинительных союзов (while, because, although, as, when, until 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 место-

```
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. 4. Разбиение на chunks при помощи правил и дальнейшей пост-обработки.

имения, подчинительные союзы. Исходя из этой идеи, был составлен список правил, задающих “chunk”. После выделения chunks идет этап пост-обработки, включающий в себя дополнительные шаги:

(1) убрать “немаксимальные” chunks, т.е. удалить те, которые являются подстрокой одного из извлеченных отрезков;

(2) отдельно добавить “отрезки”, которые не покрылись извлеченными chunks, т.е. 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, приведен на рис. 4.

4.4. Архитектурные изменения и регуляризация

Вторая группа экспериментов была связана с изменением устройства модели.

4.4.1. “Орто-регуляризация”. Данный тип регуляризации отвечает за разнообразие внутренних векторных представлений модели. Аналогичный регуляризатор используется в модели Attention-based Aspect Extraction (ABAE) [4] для обеспечения разнообразия эмбеддингов аспектов.

В ходе работы было рассмотрено два варианта “орто-регуляризации”:

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

Первый вариант “орто-регуляризатора” отвечает за разнообразие эмбеддингов на выходе последнего слоя модели BERT. Пусть M – матрица полученных векторных представлений, отнормированная вдоль строк. Тогда описанный регуляризатор вычисляется следующим образом:

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

где $I^{(1)}$ – единичная матрица, $\|\cdot\|$ – норма Фробениуса. Разнообразие эмбеддингов обеспечивается за счет “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 извлеченных масок сочинительных структур. Пусть D – матрица размера $N \times C$, содержащая в каждой строке усредненное по токенам входного предложения предсказание модели и предварительно отнормированная вдоль строк. Данный вариант “орто-регуляризатора” вычисляется по формуле:

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

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.

4.4.2. Ограничение на порядок извлекаемых меток. Этот регуляризатор “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 извлеченной маски вычисляется “штраф”, зависящий только от вероятностей, полученных моделью для данной маски:

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

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

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

4.4.3. Добавление сверточных слоев. В архитектуре исходной модели DetIE-CA за BERT следует только полносвязный слой. Использование на этом этапе еще и сверточных слоев могло бы поз-

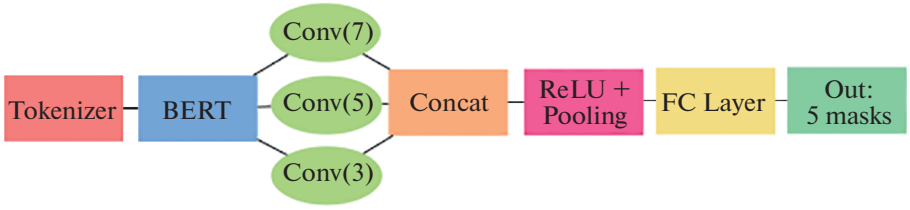

Рис. 5. Архитектура модели со сверточными слоями “в ширину”.

волить модели ориентироваться на большее количество токенов вокруг текущего при формировании итоговых предсказаний. Были рассмотрены варианты добавления последовательных и параллельных (см. рис. 5) сверточных слоев перед полносвязным слоем.

4.4.4. Модель ConvBERT в качестве кодировщика. Еще один подход к решению проблемы “масштаба” – заменить кодировщик на ConvBERT [5]. Данная модель включает в себя новый тип “голов внимания”, в котором используются свертки. 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.

4.5.1. Агрегация с помощью Beam Search. Как было сказано в главе 2, в изначальной версии DetIE-CA используется “жадная” агрегация. С целью повышения полноты был рассмотрен другой вариант агрегации с помощью *подобия* алгоритма лучевого поиска (Beam Search). Данный алгоритм был впервые описан в статье [9].

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 выдается 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 (уверенно предсказывается одна из меток). 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:

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

поэтому в подобных моделях Beam Search рассматривает несколько первых максимумов по совместной вероятности токенов и меток во входной и выходной последовательностях.

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. Тогда для каждого индекса по токенам будет известна оценка совместной вероятности двух подряд идущих меток. Используя также обычный вариант модели, можно поделить совместную вероятность двух подряд идущих токенов на вероятность одного токена и по-

Таблица 2. Сравнение оценок качества работы моделей на валидационной части датасета при разных значениях гиперпараметров

Гиперпараметры							Качество		
Размер батча	**IoU** с исп. фона или без	Кол-во масок	Кол-во обучаемых слоев 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

лучить оценку вероятности текущего токена при условии одного предыдущего.

4.5.3. Корректировка извлеченных масок на правилах. Этот подход направлен на корректировку масок сочинительных структур, полученных после этапа агрегации. Было рассмотрено два варианта правил:

(1) подряд идущие метки `CP_START` заменяются на `CP_START`, `CP`, `CP`, ..., `CP`;

(2) прерывания в одной сочинительной структуре, т.е. все метки `NONE`, которые находятся между ближайшей слева меткой `CP_START` и меткой, отличной от `CP_START` и `NONE`, справа, заменяются на `CP`.

5. СХЕМА ОБУЧЕНИЯ И ПОДБОРА ГИПЕРПАРАМЕТРОВ

Все эксперименты проводились на графической карте NVIDIA GeForce GTX 1080 Ti. Во всех экспериментах, кроме “ConvBERT”, использовался кодировщик `bert-base-multilingual-cased`.

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. 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” варьировался от 32 до 128, количество предсказываемых моделью масок – от 3 до 6, количество “размороженных” слоев в кодировщике – от 2 до 4, с учетом токенов с меткой фонового класса при подсчете **IoU** и без, оптимизаторы Adam [6] и AdamW [8] с различными значениями темпа обучения и коэффициента при L_2-регуляризаторе. В качестве планировщика темпа обучения используется ExponentialLR (он же использовался в работе [13]).

Наибольшего значения F1-меры на валидации в предварительных экспериментах с базовой моделью удалось добиться при следующих гиперпараметрах: размер “батча” =128, без учета токенов с меткой фонового класса при подсчете **IoU**, количество предсказываемых масок =5, количество “размороженных” слоев в кодировщике =4, оптимизатор Adam с lr = 10^{-4} и wd = 10^{-6}.

В табл. 2 включены несколько пар оценок качества для подходов, отличающихся лишь одним гиперпараметром, для демонстрации степени его влияния. На основании предварительных экспериментов лучший набор гиперпараметров использовался в дальнейшем как базовый.

6. РЕЗУЛЬТАТЫ

Подходы, описанные выше, продемонстрировали результаты, представленные в табл. 3.

Идеи добавления регуляризаторов, а также подходы, направленные на решение проблемы “масштаба”, не дали улучшений качества работы. В случае с чанкингом, скорее всего, это связано с неидеальностью выделения chunks с помощью предварительно заданных правил. А в подходах с добавлением сверточных слоев в изначальную архитектуру, возможно, получаются слишком “перегруженные” модели.

Таблица 3. Результаты экспериментов с разными подходами, оценка качества на тестовой части датасета PTB-CA. Полужирным шрифтом выделены строки с экспериментами, где F1-мера превосходит 0.84

Эксперимент	**P**	**R**	**F1**
Изначальная DetIE-CA	0.840	0.797	0.818
Изначальная DetIE-CA с большим числом эпох	0.845	0.802	0.823
Изначальная DetIE-CA с заменой тегов	0.851	0.830	0.840
Орто-регуляризация выходов BERT, вес=0.1	0.846	0.794	0.819
Орто-регуляризация выходов полносв. слоя, вес=0.1	0.843	0.814	0.828
Регуляризация, задающая порядок меток, вес=0.1	0.843	0.826	0.834
Регуляризация, задающая порядок меток, вес=0.3	0.839	0.820	0.829
Регуляризация, задающая порядок меток, вес=1.0	0.848	0.824	0.836
Chunking, объединение двух моделей	0.843	0.803	0.823
Добавление последовательных свёрточных слоёв	0.820	0.765	0.792
Свёрточные в ширину (послед. конкат.)	0.818	0.764	0.823
Свёрточные в ширину (avg pooling + послед. конкат.)	0.836	0.769	0.801
Свёрточные в ширину (avg pooling + циклич. конкат.)	0.802	0.732	0.765
Свёрточные в ширину (max pooling + послед. конкат.)	0.809	0.752	0.779
Свёрточные в ширину (max pooling + циклич. конкат.)	0.815	0.758	0.786
ConvBERT	0.836	0.795	0.815
Beam Search	0.856	0.833	0.844
Объединение обычной модели и модели “на парах”	0.873	0.774	0.820
CP_STARTS =>CP_START, CP, ...	0.851	0.830	0.840
Заполнение прерываний	0.846	0.826	0.836
Заполнение прерываний, CP_START =>CP_START, CP, ...	0.848	0.828	0.838
Beam Search, CP_START =>CP_START, CP, ...	0.856	0.833	0.844

Таблица 4. Сравнение качества и производительности с актуальными современными подходами

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	–

Подход с изменением процедуры агрегации наподобие алгоритма Beam Search дал существенный прирост. Подход, направленный на уточнение условных вероятностей, лишь понизил качество работы. Добавление этапа пост-обработки на правилах 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 дает дополнительные гарантии качества.

В табл. 4 представлены оценки качества на тестовой части датасета PTB-CA для достигающей в настоящий момент наилучшего качества модели IGL-CA, второй модели по качеству Teranishi-19 [12] и модели, описанной в настоящей работе. Также модель DetIE-CA существенно превосходит актуальные подходы по производительности. Оценки скорости работы DetIE-CA и IGL-CA на “инференсе” приведены в табл. 4. Teranishi-19 заведомо медленнее остальных подходов, так как в ней используется алгоритм Кока-Янгера-Касами [10] для построения дерева разбора на основе особой грамматики, который имеет вычислительную сложность $\mathbb{O}(n^3 \cdot |G|)$, где n – количество токенов во входном предложении, а $|G|$ – размер грамматики в нормальной форме Хомского [2].

7. ЗАКЛЮЧЕНИЕ

В данной работе были исследованы различные подходы к улучшению нейросетевой архитектуры модели для решения задачи выделения сочинительных связей в предложениях на английском языке. Существенного прироста в качестве удалось достичь с помощью разработанного алгоритма агрегации наподобие Beam Search.

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.

СПИСОК ЛИТЕРАТУРЫ

1. Spacy. https://github.com/explosion/spaCy, 2015.
2. *Noam Chomsky*. On certain formal properties of grammars. Information and control. 1959. V. 2 (2). P. 137–167.
3. *Jessica Ficler, Yoav Goldberg*. A neural network for coordination boundary prediction. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 1016. P. 23–32.
4. *Ruidan He, Wee Sun Lee, Hwee Tou Ng, 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). Association for Computational Linguistics. Vancouver, Canada, 2017. P. 388–397.
5. *Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, Shuicheng Yan*. Convbert: Improving BERT with span-based dynamic convolution. CoRR, abs/2008.02496, 2020.
6. *Diederik P Kingma, Jimmy Ba*. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
7. *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). 2020. P. 3748–3761.
8. *Loshchilov I., Hutter F*. Decoupled weight decay regularization. In International Conference on Learning Representations, 2019.
9. *Raj Reddy*. Speech understanding systems: A summary of results of the five-year research effort. Carnegie Mellon University, 1977.
10. *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, 1962. P. 593–608.
11. *Hiroki Teranishi, Hiroyuki Shindo, 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), Taipei, Taiwan. Asian Federation of Natural Language Processing. 2017. P. 264–272.
12. *Hiroki Teranishi, Hiroyuki Shindo, 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), Minneapolis, Minnesota, Association for Computational Linguistics. 2019. P. 3394–3403.
13. *Vasilkovsky M., Alekseev A., Malykh V., Shenbin I., Tutubalina E., Salikhov D., Stepnov M., Chertok A., Nikolenko S*. Detie: Multilingual open information extraction inspired by object detection. In Proceedings of the 36th AAAI Conference on Artificial Intelligence, 2022.
14. *Розенталь Д.Э., Теленкова М.А*. Словарь-справочник лингвистических терминов, издание третье, 1985.

NEURAL NETWORKS FOR COORDINATION ANALYSIS

A. I. Predelina[a], S. Yu. Dulikov[b], and A. M. Alekseev[a,c]

[a] *St. Petersburg State University, St. Petersburg, Russia*

[b] *"Yandex" Company, Moscow, Russia*

[c] *St. Petersburg Department of Steklov Mathematical Institute of Russian Academy of Sciences, St. Petersburg, Russia*

Presented by Academician of the RAS A.L. Semenov

The paper is dedicated to the development of a novel method for Coordination Analysis (CA) in English using the neural (deep learning) methods. An efficient solution for the task allows for the identification of potentially valuable links and relationships between speci″c parts of a sentence, making the extraction of coordinate structures an important text preprocessing tool. In this study, a number of ideas for approaching the task within the framework of "one-stage detectors" were tested. The achieved results are comparable in quality to the current most advanced CA methods while allowing to process more than 3x more sentences within a unit of time.

Keywords: natural language processing (NLP), coordination analysis (CA), machine learning (ML), neural networks