Аннотация

В данной работе исследована возможность использования семантико-синтаксического анализатора Compreno в качестве источника высокоуровневых признаков для решения задачи распознавания именованных сущностей в рамках нейросетевого подхода. Исследование проводилось на англоязычном корпусе CoNLL 2003. Полученные результаты показывают, что высокоуровневые признаки дают ощутимый прирост оценки качества, без какой-либо инженерии над ними.

Ключевые слова: нейронные сети, распознавание именованных сущностей.

Содержание

Введение

Согласно "Инструкции по определению именованных сущностей"[1], именованной сущностью считается слово или словосочетание, предназначенное для конкретного, вполне определённого предмета или явления, выделяющее этот предмет или явление из ряда однотипных предметов или явлений. Примерами именованных сущностей являются имена людей, названия организаций и локаций. Задача распознавания именованных сущностей (Named Entity Recognition, NER) состоит в выделении и классификация именованных сущностей в тексте. В рамках конференции CoNLL 2003 проводилось соревнование для оценки качества методов распознавания именованных сущностей четырех типов на англоязычном корпусе [Tjong Kim Sang and De Meulder 2003]. Для решения задачи NER предлагалось много разных подходов [Nadeau and Sekine 2007]. В последнее время было показано, что методы на основе нейронных сетей показывают лучшие результаты для различных языков и корпусов, включая CoNLL 2003 [Yang et al. 2016].

Вместо большого количества вручную построенных признаков, нейросетевые методы используют универсальные векторные представления слов [Mikolov et al. 2013]. Согласно гипотезе о дистрибутивности, эти представления кодируют в себе смысл слов [Sahlgren 2008]. Это позволяет строить мультизадачные и языконезависимые архитектуры [Collobert et al. 2011, Yang et al. 2016].

Несмотря на то, что использование универсальных векторных представлений получило в последнее время огромную популярность в силу своей эффективности и огромной экономии человеческих усилий, большой интерес все еще представляет исследование возможностей использования высокоуровневых признаков в качестве входных данных для нейросетей. Так, например, в работах [Xu et al. 2014, Bian et al. 2014] описано использование морфологических, синтаксических и семантических признаков для построения более совершенных векторных представлений слов.

[1] http://opencorpora.org/wiki/Nermanual/1

Compreno - это технология автоматического анализа текстов на естественном языке, в основе которой лежит многоуровневое лингвистическое описание, создававшееся профессиональными лингвистами в течение длительного времени [Anisimovich et al. 2012]. Помимо ручного описания Compreno использует для анализа большое количество информации, извлекаемой различными статистическими методами из текстовых корпусов. В Compreno реализована процедура семантико-синтаксического анализа текста, в результате которой любому предложению на естественном языке (английском или русском) ставится в соответствие семантико-синтаксическое дерево, моделирующее смысл предложения и содержащее грамматическую и семантическую информацию о каждом слове предложения.

В данной работе исследована возможность использования семантико-синтаксического анализатора Compreno в качестве источника высокоуровневых признаков для задачи NER на корпусе CoNLL 2003 в рамках нейросетевого подхода.

Данная работа организована следующим образом:

— в 1 части отражена постановка задачи, проведен обзор методов решения и обоснован выбранный метод решения задачи,

— во 2 части приведен алгоритм решения поставленной задачи и её программная часть,

— в 3 разделе описана экспериментальная часть.

Полученные результаты показывают повышение F1-меры почти на 1% на корпусе CoNLL 2003 при использовании синтактико-семантических признаков Compreno (87.49% против 88.47%). При этом затраты на их внедрение были минимальными - инженерия над признаками не проводилась.

1 Теоретико-аналитическая часть

1.1 Постановка задачи

Исследовать возможность использования семантико-синтаксического анализатора Compreno в качестве источника высокоуровневых признаков для задачи NER на корпусе CoNLL 2003 в рамках нейросетевого подхода.

1.2 Обзор литературы

Победители соревнования по NER CoNLL 2003 [Florian et al. 2003], получившие 88.76% F1, представили систему использующую комбинацию различных алгоритмов машинного обучения. В качестве признаков был использован их собственный, вручную составленный газетир, POS-теги, CHUNK-теги, суффиксы, префиксы и выход других NER-классификаторов, тренированных на внешних данных.

[Collobert et al. 2011] представили комбинацию сверточной нейронной сети с условными случайными полями, получившую 89.59% F1 на корпусе CoNLL 2003. Их нейросетевая архитектура не зависит от задачи и используется как для NER, так и для частеречной разметки (part-of-speech tagging), поиска синтаксически связанных групп соседних слов (chunking), установления семантических ролей (semantic role labelling). Для задачи NER они использовали три типа признаков - векторное представление слова, капитализацию и небольшой газетир, включенный в соревнование CoNLL 2003.

[Chiu and Nichols 2015] представили комбинацию сверточных сетей, рекуррентных сетей и условных случайных полей показывающую 91.62% F1. Они использовали такие же признаки как у [Collobert et al. 2011], дополнительный, вручную сформированный газетир на основе DBpedia и обучались на train+dev[1] выборке CoNLL 2003. Кроме корпуса CoNLL 2003 они тестировали архитектуру на более крупном англоязычном корпусе OntoNotes 5.0. На нем они получили state-of-the-art результат 86.28%.

[1] Объединенная обучающая и валидационная выборки

[Yang et al. 2016] представили глубокую иерархическую рекуррентную нейросетевую архитектуру с условными случайными полями для разметки последовательностей. Они использовали такие же признаки как у [Collobert et al. 2011]. Кроме англоязычного корпуса CoNLL 2003, где они получили state-of-the-art 90.94% F1 при обучении только на обучающей выборке (train set), они тестировали работу нейросети на CoNLL 2002 Dutch NER и CoNLL 2003 Spanish NER. На этих корпусах они улучшили предыдущий state-of-the-art результат: 82.82% до 85.19% на CoNLL 2002 Dutch NER и 85.75% до 85.77% на CoNLL 2003 Spanish NER.

Современные работы используют векторное представление слов и условные случайные поля в своих моделях. Из сторонних признаков применяют только газетиры. В работах [Xu et al. 2014, Bian et al. 2014] описано применение дополнительных признаков для слов (морфологических, синтаксических, семантических) для создания более совершенных векторных представлений. Такие векторные представления помогают повысить оценку качества в прикладных задачах [Xu et al. 2014].

1.3 Обзор корпуса CoNLL 2003

В данной работе рассмотрен англоязычный корпус CoNLL 2003, т.к. он является одним из самых распространенных корпусов на котором год от года измеряют оценку качества методов распознавания именованных сущностей.

CoNLL 2003 [Tjong Kim Sang and De Meulder 2003] - англоязычный корпус для оценки качества методов распознавания именованных сущностей. Корпус содержит обучающую, тестовую и валидационную выборку. Размечено 4 типа сущностей - персоны (PER), организации (ORG), локации (LOC) и другие (MISC).

Таблица 1.1 — Количество статей, предложений, токенов и именованных сущностей

Выборка	Статьи	Предлож-я	Токены	LOC	MISC	ORG	PER
Обучающая	946	14987	203621	7140	3438	6321	6600
Валидац-я	216	3466	51362	1837	922	1341	1842
Тестовая	231	3684	46435	1668	702	1661	1617

Корпус размечен по схеме *Inside, Outside, Begin (IOB)*:

— слово помечают тегом O (Outside), если оно не является именованной сущностью.

— Тегом I-XXX (Inside), где XXX - тип именованной сущности, если слово является именованной сущности или ее частью.

— Тегом B-XXX (Begin), если слово является началом именованной сущности.

Пример:

, O
Surrey I-ORG
captain O
Chris B-PER
Lewis I-PER
, O

Оценка качества считается с помощью метрики $F_{\beta=1}$(F1-micro-average):

$$F_\beta = \frac{(\beta^2 + 1) * precision * recall}{\beta^2 * precision + recall},$$

где *precision* (точность) - процент корректных именованных сущностей найденных системой, *recall* (полнота) - процент всех именованных сущностей найденных системой. Именованная сущность считается найденной корректно, если вся сущность помечена правильно. В CoNLL 2003 включен скрипт conlleval, оценивающий качество классификации.

В работе [Collobert et al. 2011] предлагается использовать схему *Inside, Outside, Begin, End, Single (IOBES)*, т.к. она более явно указывает тег для слова:

— слово помечают тегом O (Outside), если оно не является именованной сущностью.

— Тегом I-XXX (Inside), где XXX - тип именованной сущности, если слово является частью именованной сущности.

— Тегом B-XXX (Begin), если слово является началом именованной сущности.

— Тегом E-XXX (End), если слово является концом именованной сущности.

— Тегом S-XXX (Single), если именованная сущность состоит из одного слова.

Пример:

, O
Surrey S-ORG
captain O
Chris B-PER
Lewis E-PER
, O

Для оценки качества IOBES конвертируют в IOB и подают на вход conlleval.

Также в CoNLL 2003 включен небольшой газетир.

1.4 Обзор нейросетевого подхода к решению задачи NER

Традиционным подходом к решению задач из области автоматической обработки текстов, включая NER, является использованием алгоритма обучения с учителем, например машины опорных векторов с линейным ядром. Вручную составленные признаки подаются на вход алгоритма обучения с учителем. Выбор признаков - это практически полно-

стью эмпирический процесс, построенный на лингвистической интуиции и решаемой задаче.

Нейросетевой подход предполагает использование минимального количества признаков. Обычно это векторные представления слов полученные из большого корпуса с использованием алгоритмов обучения без учителя.

В этом разделе будет рассмотрен нейросетевой подход из работы [Collobert et al. 2011]. Была выбрана именно эта работа так как:

— в ней представлена модель, получающая сравнимую со state-of-the-art F1-меру на CoNLL 2003,

— представленная модель имеет различные программные имплементации,

— векторные представления слов под названием «Senna Embeddings» используемые в этой работе находятся в открытом доступе в сети.

Последние два пункта указывают на возможность воспроизведения результатов из статьи относительно небольшими усилиями.

1.4.1 Векторные представления слов

Согласно Википедии[1], *векторное представление* — это общее название для различных подходов к моделированию языка и обучению представлений в обработке естественного языка, направленных на сопоставление словам (и, возможно, фразам) из некоторого словаря векторов из R^n, где n - значительно меньше количества слов в словаре (обычно от 50 до 1000).

[Collobert et al. 2011] использовали нейронные сети для построения векторных представлений слов. Они тренировали нейронную сеть на данных корпусе английской Википедии[2] и Reuters RCV1[3]. Были использованы 130000 самых частотных слов, остальные слова кодировались специальным токеном UNKNOWN. Полученные векторные пред-

[1] `https://en.wikipedia.org/wiki/Word_embedding`

[2] На данных ноября 2007 года из http://download.wikimedia.org

[3] Доступно http://trec.nist.gov/data/reuters/reuters.html

ставления называются Senna Embeddings и доступны в сети по адресу: http://ronan.collobert.com/senna/.

Senna Embeddings использовались как входные признаки на нейронные сети для решения задачи NER.

1.4.2 Описание нейросетевых моделей

В работе [Collobert et al. 2011] описываются две нейросетевые модели:

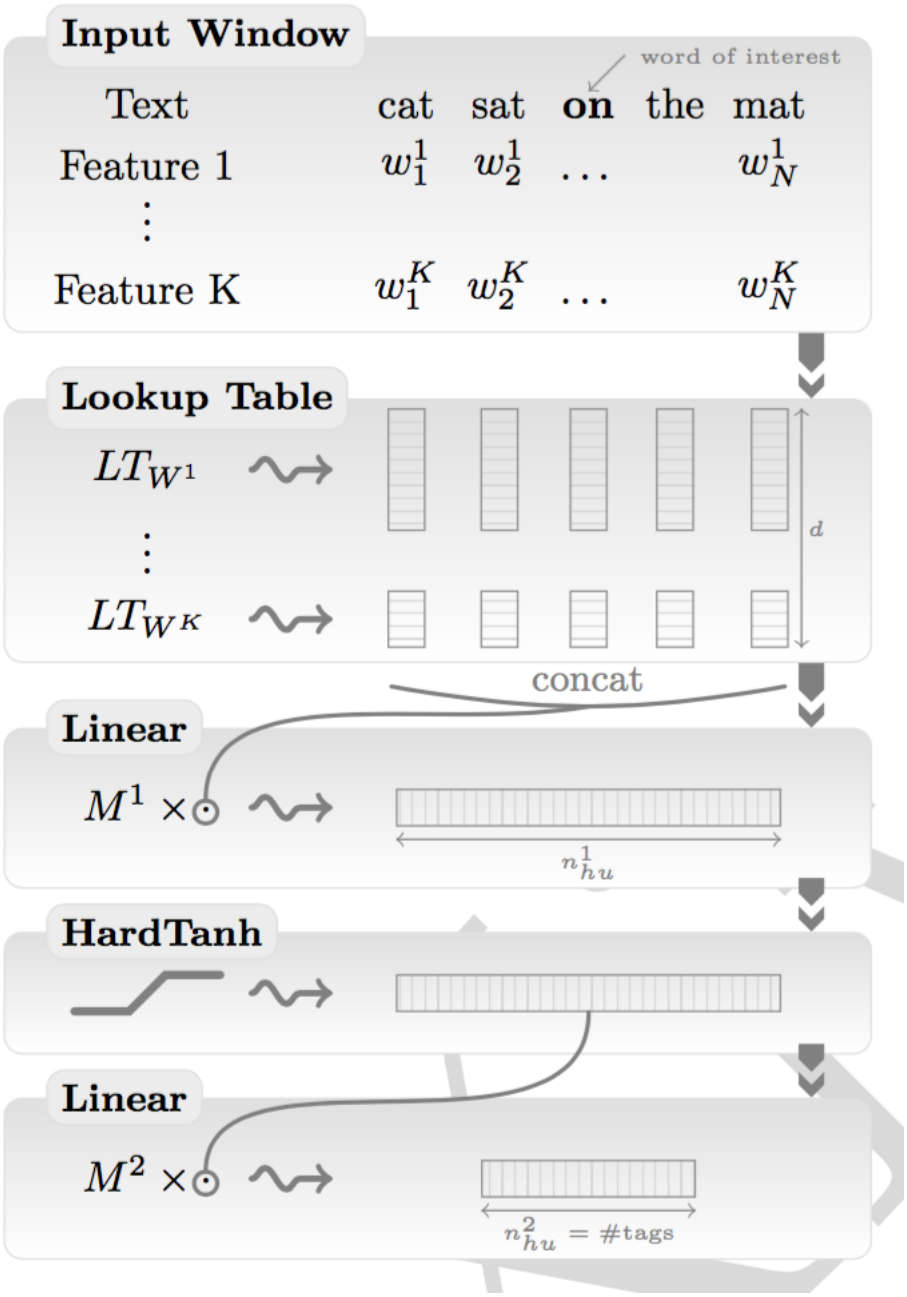
— Оконный (window), который предсказывает тег слова на основе контекста (окна) вокруг слова,

— сверточный (convolution), который предсказывает тег слова используя всё предложение.

1.4.2.1 Оконная модель

Оконная модель представлена на рис. 1.1.

Рисунок 1.1 — Оконная модель из [Collobert et al. 2011]



На рис. 1.1 предсказывается тег для слова on. При классификации используется контекст для этого слова. Всё окно состоящее из 5 слов пропускается через так называемый Lookup Table.

Lookup Table - это специальной слой в нейронной сети, который отображает каждое слово в вектор весов. Причем вектор весов обучается вместе с сетью. Более формально, каждому слову w из словаря D ставится в соответствие вектор размерности d, который задается Lookup

Table слоем $LT_W(w)$:

$$LT_W(w) = W_{\cdot,w},$$

где $W \in R^{d \times |D|}$ - матрица весов для обучения, $W_{\cdot,w}$ - w-ый столбец матрицы W. Размерность d - гиперпараметр.

Также этот слой может принимать на вход последовательность слов $w_1 \dots w_K$, где K - величина окна. В этом случае выходом будет матрица:

$$f_1 = LT_W(w_1 \dots w_K) = (W_{\cdot,w_1} \dots W_{\cdot,w_K})$$

После Lookup Table слоя, полученная матрица преобразуется в один вектор с помощью операции *конкатенации Concat*:

$$f_2 = Concat(LT_W(w_1 \dots w_K)) = \begin{bmatrix} W_{\cdot,w_1} \\ \vdots \\ W_{\cdot,w_K} \end{bmatrix},$$

Затем этот вектор подается на *полносвязный слой (Linear Layer)*, который выполняет аффинное преобразование:

$$f_3 = Linear(f_2) = W^2 f_2 + b^2, \tag{1.1}$$

где $W^2 \in R^{d_2 \times |f_2|}$, $b \in R^{d_2}$. Гиперпараметр d_2 - это количество нейронов в данном слое.

Каждый элемент полученного вектора $f_3 \in R^{d_2}$ пропускается через нелинейную функцию. На рис. 1.1 это *HardTanh*:

$$f_4 = HardTanh(f_3),$$

$$HardTanh(x) = \begin{cases} -1, \text{if } x < -1 \\ x, \text{if } -1 <= x <= 1 \\ 1, \text{if } x > 1 \end{cases} \tag{1.2}$$

Преимуществом этой функции перед гиперболическим тангенсом является более быстрое время вычисления.

Последний слоем является полносвязный слой. Количество нейронов в нем равно количеству предсказываемых классов.

$$f_5 = Linear(f_4) = W^5 f_4 + b^5.$$

Каждый элемент x_i полученного вектора f_5 пропускается через функцию *Softmax* для получения вероятностей:

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Другими словами это нормализация вектора с помощью экспоненциальной функции.

Для обучения нейронной сети минимизируется функционал ошибки с использованием пакетного градиентного спуска (mini-batch gradient descent). *Функционал ошибки* C:

$$C = -\sum_{(x,y_k)\in T} \log(\mathsf{P}(y_k|x,\theta)),$$

$$C \to \min_{\theta},$$

$$\log(\mathsf{P}(y_k|x,\theta)) = f_4(y_k) - \log\sum_i e^{f_4(y_i)},$$

где (x, y_k) - пара объект, класс из обучающей выборки T; θ - веса всей нейросети (все матрицы W), $f_4(y_i)$ - значение последнего слоя для класса y_i.

Функционал ошибки C минимизируется с помощью *пакетного градиентного спуска*:

$$\theta = \theta - \alpha \sum_{(x,y_k)\in T_s} \nabla_\theta(-\log \mathsf{P}(y_k|x,\theta)),$$

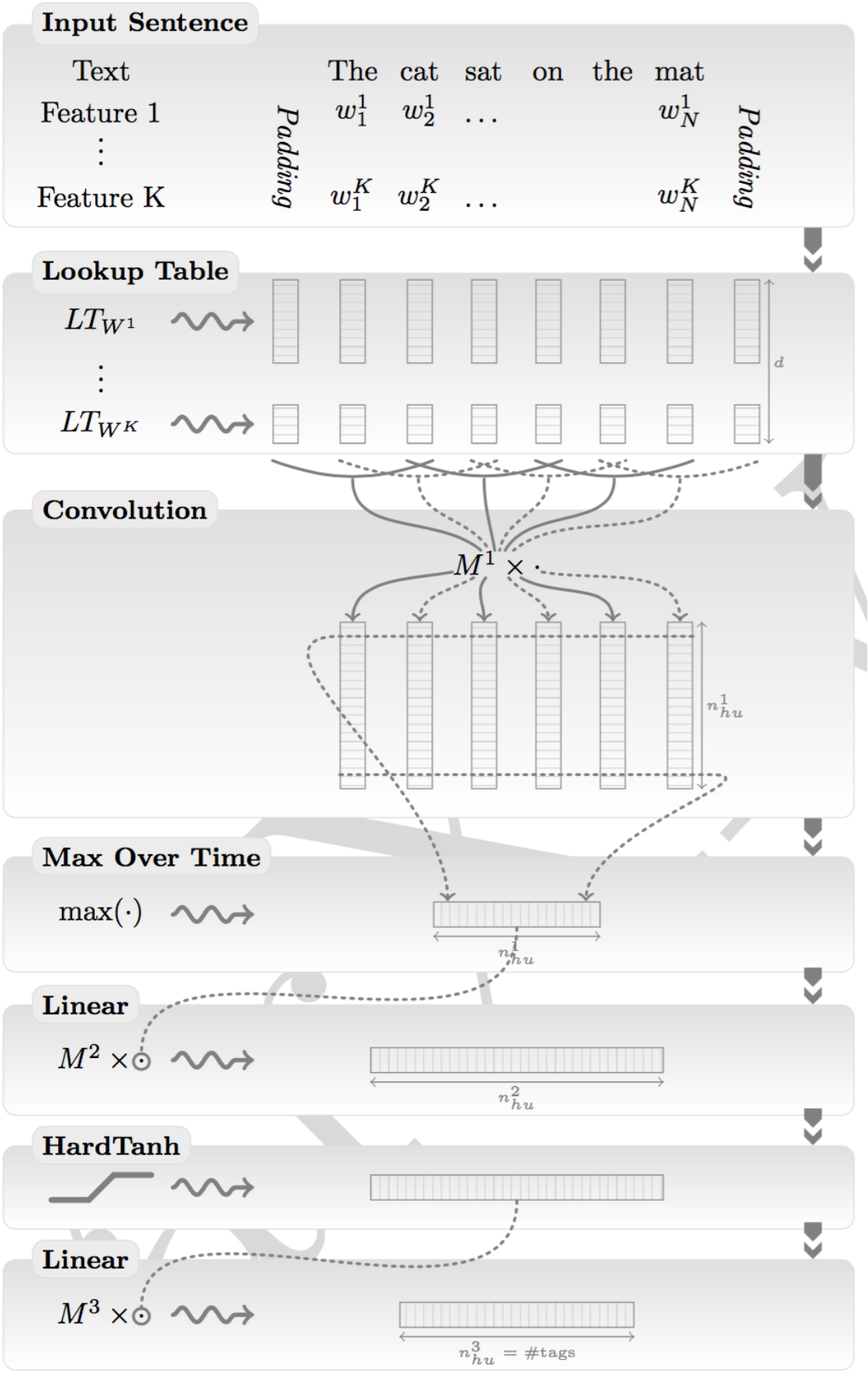
где α - шаг обучения, T_s - случайное подмножество объектов из обучающей выборки. Размер подмножества T_s и α являются гиперпараметрами.

Стохастического градиентный спуск (stochastic gradient descent) является модификацией пакетного градиентного спуска. Отличие заключается в том, что размер T_s равен 1.

1.4.2.2 Сверточная модель

Сверточная модель представлена на рис. 1.2.

Рисунок 1.2 — Сверточная модель из [Collobert et al. 2011]



В отличие от оконной модели, она принимает на вход всё предложение.

Все слова предложения пропускаются через Lookup Table, как описано в разделе 1.4.2.1, после чего попадают на сверточный слой.

Сверточный слой (temporal convolution) является обобщением полносвязного слоя (1.1) из оконного подхода:

а) сначала осуществляется проход окном на полученной на предыдущем шаге матрице f_1,

б) столбцы попавшие в окно конкатенируются,

в) полученный вектор пропускается через полносвязный слой, причем матрица весов W^3 является одной и той же в каждом проходе.

На выходе, после прохода окном по всей матрице f_1, получается матрица f_3. Более формально:

$$f_3 = (W^3 f_1[1, d_{win}] + b^3, \ldots, W^3 f_1[|f_1| - d_{win}, d_{win}] + b^3),$$

$$d_{f_1,win} = d_{win} * d,$$

где $f_1[i, d_{win}]$ - означает конкатенацию столбцов f_1 с i по $i + d_{win}$, $d_{f_1,win}$ - размерность столбцов после конкатенации, $W^3 \in R^{d_2 \times d_{f_1,win}}$, $b \in R^{d_2}$. Гиперпараметр d_2 - это количество нейронов в данном слое.

После сверточного слоя получается матрица $f_3 \in R^{d_2 \times |f_1| - d_{win}}$. Количество строк в ней фиксировано, но количество столбцов зависит от длины предложения. Чтобы получить вектор признаков фиксированный длины выполняется *операция получения максимума по строкам (Max over time)* над f_3. Смысл этой операции заключается в получении наиболее значимых признаков из каждого окна. Из каждой строки матрицы f_3 извлекается максимум и в результате этой операции получается вектор $f_5 \in R^{d_2}$.

Далее над полученным фиксированным вектором признаков проводятся уже знакомые по разделу 1.4.2.1 операции с полносвязными слоями (1.1) и нелинейной функцией HardTanh (1.2).

Для обучения используется минимизация функционала ошибки и пакетный градиентный спуск описанные в разделе 1.4.2.1.

В оригинальное статье [Collobert et al. 2011] в сверточном подходе минимизировался другой функционал ошибки, который включает в себя условные случайные поля. Это позволило им достичь более высокой оценки качества, но в то же время замедлило время проведения экспериментов. Данный функционал не поддерживается библиотеками для работы с нейронными сетями, поэтому его необходимо реализовывать с нуля. Из-за этих причин этот функционал ошибки не рассмотрен в этой работе.

1.4.2.3 Уточнения к моделям

В разделе 1.4.2.2 теги предсказываются для каждого слова. Т.к. на вход нейросети поступает предложение, то необходимо кодировать местоположение слова в предложении для которого предсказывается тег. Это осуществлялось с помощью Lookup Table. Позиция слова w_j в предложении кодировалась с помощью подсчета расстояния относительно слова w_i для которого предсказывается тег. Слово w_i кодировалось в Lookup Table как 0. Слово w_{i-k} как $-k$. Слово w_{i+k} как $+k$.

В качестве регуляризации использовался Dropout слой после каждого полносвязного слоя, кроме последнего. Dropout слой [Srivastava et al. 2014] с заданной вероятностью p зануляет выход нейрона.

Важным условием хорошего обучения нейронной сети является начальная инициализация весов. В данной работе использовано 2 подхода к инициализации весов:

а) $U(\frac{-1}{\sqrt{fi}}, \frac{1}{\sqrt{fi}})$
б) $U(-\sqrt{\frac{2}{fi+fo}}, \sqrt{\frac{2}{fi+fo}})$,

где U - равномерное распределение, fi - количество входов в слой, fo - количество выходов из слоя. Первый подход описан в [Collobert et al. 2011], второй в [Glorot and Bengio 2010].

1.5 Обзор синтактико-семантических признаков

Существует много инструментов для получения дополнительных признаков для слова. Для извлечения синтаксических признаков часто

используют MaltParser [Nivre et al. 2006]. Для получения семантических признаков применяют BabelNet [Navigli and Ponzetto 2010].

В данной работе для получения синтактико-семантических признаков используется Compreno. Признаки синтактико-семантического дерева Compreno кодировались в бинарные вектора и соотносились с токенами исходного текста[1], тем самым наделяя их синтактико-семантическими признаками. Размерность пространства синтактико-семантических признаков получилась равной 83950.

Плотные вектора большой размерности сильно замедляют процесс оптимизации и для хорошего обучения требуется много данных и вычислительных ресурсов. В таких случаях часто применяют методы для уменьшения размерности, например сингулярное разложение или автоэнкодеры. Минусом таких методов является потеря информации после сжатия.

Если же вектора большой размерности разреженные, то используют специальные методы для работы с такими данными [Davis et al. 2016].

В данной работе предлагается 2 способа внедрения синтактико-семантических признаков:

— сжать синтактико-семантические вектора с помощью сингулярного разложения (SVD) и добавить как еще один Lookup Table в сверточную нейронную сеть;

— добавить еще одну нейронную сеть для синтактико-семантических признаков и оптимизировать её вместе со сверточной нейронной сетью.

[1]Почти для всех токенов в соответствующем дереве нашлась соответствующая вершина. Токены для которых не была найдена вершина, кодировались специальным признаком 83951

2 Проектная часть

2.1 Описание моделей

Для экспериментов были реализованы следующие модели:

— оконная (WindowNet) - данная модель подробно описана в разделе 1.4.2.1,

— сверточная (ConvNet) - данная модель подробно описана в разделе 1.4.2.2,

— полносвязная нейронная сеть для работы с разреженными синтактико-семантическими признаками (ComprenoNet),

— комбинация сверточной и полносвязной сети (ConvNet + ComprenoNet)

2.1.1 Гиперпараметры моделей и признаки

Общие для всех моделей гиперпараметры:

— $\alpha = 0.01$ - шаг обучения,

— $p = 0.5$ - вероятность для Dropout слоя,

— количество нейронов в последнем полносвязном слое равно 17, т.к. используется схема IOBES. Четыре для каждого из четырех типов тегов и один для Outside.

Гиперпараметры для оконной модели (рис. 1.1) следующие:

— $d_2 = 300$ - количество нейронов в полносвязном слое,

— $K = 5$ - величина окна. Если в окне нет слов, то окно дополняется специальным токеном PADDING,

— $|T_s| = 32$ - размер подмножества для пакетного градиентного спуска.

Гиперпараметры для сверточной модели (рис. 1.2) следующие:

— $d_2 = 300$ - количество нейронов в сверточном слое,

— количество нейронов в полносвязном слое также равно 300,

— $d_{win} = 3$ - величина окна. Если в окне нет слов, то окно дополняется специальным токеном PADDING,

— $|T_s| = 33$ - размер подмножества для пакетного градиентного спуска.

Для экспериментов были использованы следующие признаки:

— Embeddings - векторные представления слов Senna Embeddings,

— Capitalization - дискретный признак капитализации слова с 5 возможными вариантами (слово начинается с большой буквы, слово содержит большую букву, все символы в слове состоят из больших букв, нет больших букв в слове, не слово),

— Position - кодирование позиции слова в предложении (описано в разделе 1.4.2.2),

— Gazetteer - проверяется присутствие слова в газетире CoNLL 2003,

— Compreno sparse features - синтактико-семантические признаки Compreno размерности 83950,

— Compreno SVD 1024 - синтактико-семантически признаки Compreno размерности 83950 были сжаты с использованием модификации сингулярного разложения для работы с разреженными признаками TruncatedSVD[1] до размерности 1024. После сжатия описываемая дисперсия была равна 72%. Т.е. потерялось 28% информации.

Все признаки, кроме Compreno sparse features, были включены в нейронные сети с использованием Lookup Table.

2.1.2 ComprenoNet

Архитектура и гиперпараметры полносвязной нейронной сети для работы с разреженными синтактико-семантическими признаками (ComprenoNet) представлены ниже:

```
input -> (1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> output
(1): nn.SparseLinear(83951, 256)
(2): nn.Dropout(0.5)
(3): nn.HardTanh
(4): nn.Linear(256 -> 256)
(5): nn.Dropout(0.5)
(6): nn.HardTanh
(7): nn.Linear(256 -> 17)
```

[1] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

SparseLinear представляет из себя обычный полносвязный слой (Linear), который оптимизирован для работы с разреженными данными. Другие слои и процесс обучения аналогичны тем, что были в разделе 1.4.2. Общий алгоритм работы сети следующий:

а) На вход подается разреженный вектор признаков слова (размерность 83951) для которого предсказывается тег.

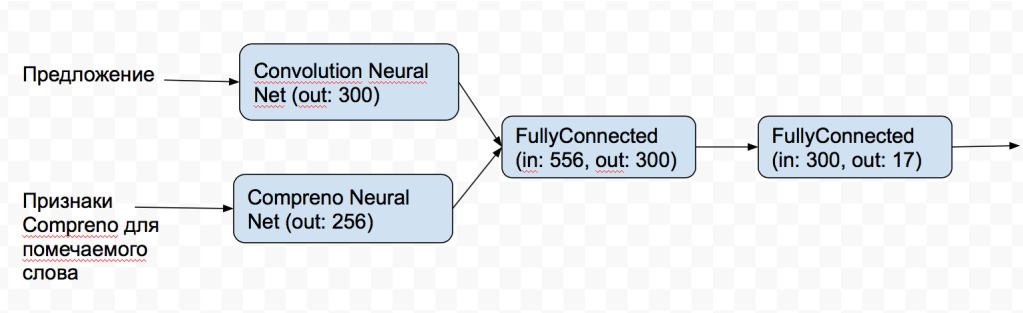
б) Далее этот вектор пропускается через 2 полносвязных слоя.

в) На выходе еще один полносвязный слой, который выдает вероятность определенного тега. Выходов также 17.

2.1.3 ConvNet + ComprenoNet

Архитектура и гиперпараметры комбинации сверточной и полносвязной сети (ConvNet + ComprenoNet) представлены на рис. 2.1. Эти

Рисунок 2.1 — Комбинация сверточной и полносвязной сети (ConvNet + ComprenoNet)



сети соединяются следующим образом:

а) Из обеих нейросетей удаляются выходные слои.

б) Предыдущие слои из обеих сетей соединяются в новый полносвязный слой.

в) Новый полносвязный слой соединяется с выходным слоем. Выходов как и тегов 17.

Веса у объединенной сети были инициализированы предобученными моделями (ConvNet, ComprenoNet).

2.2 Программная реализация

Для программной реализации описанных выше моделей существуют различные фреймворки для нейронных сетей. *Фреймворк*, согласно Википедии[1] - это программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Выбору фреймворка для работы с нейросетями посвящена работа [Bahrampour et al. 2015]. В ней они сравнивали такие программные платформы как:

— Caffe,
— Neon,
— Theano,
— Torch.

В таблице 2.1 из [Bahrampour et al. 2015] указаны особенности этих фреймворков.

Таблица 2.1 — Особенности нейросетевых фреймворков

	Caffe	Neon	TensorFlow	Theano	Torch
Язык	C++	Python	C++	Python	Lua
CPU	+	+	+	+	+
Поддержка многопоточ-и CPU	+	-	+	+	+
GPU	+	+	+	+	+
Поддержка неск-х GPU	+	+	+	-	+
Nvidia cuDNN	+	-	+	+	+
Простота модификации	-	-	+-	+	+
Поддержка сообществом	+++	+	++	+++	++

Практически все современные нейросетевые библиотеки поддерживают работу с графическими процессорами (GPU). Производители GPU делают специальные библиотеки для научных вычислений на гра-

[1] `https://en.wikipedia.org/wiki/Software_framework`

фических процессорах. Например, Nvidia cuDNN (NVIDIA CUDA Deep Neural Network library) - специальная библиотека, которая ускоряет вычислений для нейронных сетей. Использование графических процессоров позволяет ускорить обучение нейросетей в разы по сравнению с центральным процессорным устройством (CPU). Некоторые фреймворки поддерживают работу с несколькими GPU одновременно. Это позволяет еще более значительно ускорить работу нейронных сетей.

Есть программные платформы заточенные для решения одной определенной задачи. Например, Caffe - нейросетевой фреймворк для решения задач связанных с компьютерным зрением. Его использование для текстовых данных является затруднительным. В то же время есть универсальные фреймворки, такие как Torch, TensorFlow, которые позволяют решать задачи с использование нейронных сетей в различных областях, включая автоматическую обработку текстов. Они имеют гибкую архитектуру, поэтому их можно достаточно просто модифицировать.

Все фреймворки описанные в таблице 2.1 являются общедоступными, имеют открытый код и поддерживаются сообществом разработчиков и разными компаниями. Например, TensorFlow поддерживает компания Google, torch компания Facebook.

Также стоит отметить, что сверточная модель описанная в разделе 1.4.2.2 уже имеет программную реализацию в открытом доступе в сети по адресу: http://ronan.collobert.com/senna/. Минусом данной реализации является то, что она написана без использования нейросетевых фреймворков, на языке программирования C, не поддерживает GPU, не поддерживается сообществом, имеет тяжелый в понимании код, заточена под конкретную задачу и не содержит в себе различные современные нейросетевые решения, например Dropout.

Исходя из следующих требований:

— гибкость в реализации нейросетевых архитектур,
— быстрота проведения экспериментов,
— поддерживаемый код,

был выбран нейросетевей фреймворк torch[1].

Код для воспроизведения экспериментов выложен по адресу: github.com/sld/torch-conv-ner.

Скорость обучения на машине с GPU Amazon AWS g2.2xlarge[1]:

— 1 эпоха[2] при одиночной обработке (stochastic gradient descent): ∼450 сек.

— 1 эпоха при пакетной обработке (mini-batch gradient descent): ∼171 сек.

— Модель получающая 87.49% обучалась 91 эпоху (∼4.2 часа).

— 1 эпоха при пакетной обработке с использованием признаков Compreno: ∼615 сек.

Скорость классификации составляет 2500 токенов в секунду при пакетной обработке.

[1] `http://torch.ch`

[1] https://aws.amazon.com/ru/ec2/instance-types/

[2] Эпоха - это один полный проход по обучающей выборке

2.3 Эксперименты

2.3.1 Эксперименты без синтактико-семантических признаков

По таблице 2.2 видно, что результаты немного ниже чем у [Collobert et al. 2011]. Это связано с тем, что для Window подхода использован другой метод оптимизации, а для Convolution подхода не были применены условные случайные поля (ConvNet + CRF в таблице 2.2).

В качестве референсной, будет использована модель из последнего эксперимента показывающая 87.49% F1. Это сделано для чистоты эксперимента, т.к. далее обучение происходило только на обучающей выборке по правилам соревнования CoNLL 2003 и применялся mini-batch gradient descent для ускорения экспериментов.

Таблица 2.2 — Результаты экспериментов без использования синтактико-семантических признаков

Модель	Признаки	Выборка	Метод оптимизации	Полученная F1, %	F1 в статье Collobert et al. [2011]
Window	Embeddings, Capitalization	train	Mini-batch gradient descent	86.27	-
Window	Embeddings, Capitalization	train	Stochastic gradient descent	-	86.97
ConvNet + CRF	Embeddings, Capitalization, Position	train	Stochastic gradient descent	-	88.67
ConvNet + CRF	Embeddings, Capitalization, Position, Gazetteer	train	Stochastic gradient descent	-	89.59
ConvNet	Embeddings, Capitalization, Position	train	Stochastic gradient descent	86.77	-
ConvNet	Embeddings, Capitalization, Position, Gazetteer	train	Stochastic gradient descent	87.89	-
ConvNet	Embeddings, Capitalization, Position, Gazetteer	train + dev	Stochastic gradient descent	88.37	-
ConvNet	Embeddings, Capitalization, Position, Gazetteer	train	Mini-batch gradient descent	**87.49**	-

2.3.2 Эксперименты с синтактико-семантическими признаками сжатыми с использованием SVD

Таблица 2.3 — Результаты с синтактико-семантическими признаками сжатыми SVD

Модель	Признаки	Выборка	Метод оптимизации	Полученная F1, %
ConvNet	Position, Compreno SVD 1024	train	Mini-batch gradient descent	75.89
ConvNet	Capitalization, Position, Gazetteer, Compreno SVD 1024	train	Mini-batch gradient descent	81.83
ConvNet	Embeddings, Capitalization, Position, Gazetteer, Compreno SVD 1024	train	Mini-batch gradient descent	86.85
ConvNet	Embeddings, Capitalization, Position, Gazetteer	train	Mini-batch gradient descent	87.49

По таблице 2.3 видно, что такой способ ведет к небольшому ухудшению F1-меры.

2.3.3 Эксперименты с синтактико-семантическими признаками для совместно-оптимизированной нейросети

Веса у ConvNet + Compreno Net были инициализированы обученными моделями - моделью показывающую 87.49% для сверточной сети и моделью показывающую 72.85% (см. таблицу 2.4) для второй нейронной сети.

По таблице 2.4 видно, что признаки Compreno улучшают F1-меру почти на один процент.

Таблица 2.4 — Результаты с синтактико-семантическими признаками для объединенной нейросети

Модель	Признаки	Выборка	Метод оптимизации	Полученная F1, %
Compreno Net	Compreno sparse features	train	Mini-batch gradient descent	72.85
ConvNet	Embeddings, Capitalization, Position, Gazetteer	train	Mini-batch gradient descent	87.49
ConvNet + Compreno Net	Embeddings, Capitalization, Position, Gazetteer, Compreno sparse features	train	Mini-batch gradient descent	**88.47**
ConvNet + Compreno Net	Embeddings, Capitalization, Position, Gazetteer, Compreno sparse features	train + dev	Mini-batch gradient descent	88.81

Заключение

В данной работе исследована возможность использования семантико-синтаксического анализатора Compreno в качестве источника высокоуровневых признаков для задачи NER на корпусе CoNLL 2003 в рамках нейросетевого подхода. Удалось найти простой вариант подключения признаков Compreno к сверточной нейронной сети за счет которого F1-мера повысилась с 87.49% до 88.47%.

В будущем планируется внедрить условные случайные поля в существующую модель для повышения F1-меры и исследовать работу предложенного решения на других корпусах. Также интересным направлением для исследований является создание векторных представлений слов с учетом синтактико-семантических признаков.

Список использованных источников

Anisimovich, K. V., Druzhkin, K. J., Minlos, F. R., Petrova, M. A., Selegey, V. P., and Zuev, K. A. (2012). Syntactic and semantic parser based on ABBYY Compreno linguistic technologies. In *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference Dialog*, page 18.

Bahrampour, S., Ramakrishnan, N., Schott, L., and Shah, M. (2015). Comparative study of caffe, neon, theano, and torch for deep learning. *arXiv preprint arXiv:1511.06435*.

Bian, J., Gao, B., and Liu, T.-Y. (2014). Knowledge-powered deep learning for word embedding. In *Machine Learning and Knowledge Discovery in Databases*, pages 132–148. Springer.

Chiu, J. P. and Nichols, E. (2015). Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Davis, T. A., Rajamanickam, S., and Sid-Lakhdar, W. M. (2016). A survey of direct methods for sparse linear systems.

Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. (2003). Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Navigli, R. and Ponzetto, S. P. (2010). Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 216–225. Association for Computational Linguistics.

Nivre, J., Hall, J., and Nilsson, J. (2006). Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219.

Sahlgren, M. (2008). The distributional hypothesis. from context to meaning: Distributional models of the lexicon in linguistics and cognitive science (special issue of the italian journal of linguistics). *Rivista di Linguistica*, 20(1).

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Xu, C., Bai, Y., Bian, J., Gao, B., Wang, G., Liu, X., and Liu, T.-Y. (2014). Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1219–1228. ACM.

Yang, Z., Salakhutdinov, R., and Cohen, W. (2016). Multi-task cross-lingual sequence tagging from scratch. *CoRR*, abs/1603.06270.

Список иллюстраций

Список таблиц