Инженерный вестник Дона, №7 (2021)
ivdon.ru/ru/magazine/archive/n7y2021/7066

Методы распознавание именованных сущностей в русском языке

М.А. Маслова, А.С. Дмитриев, Д.О. Холкин

Волгоградский государственный технический университет

Аннотация: В данной статье кратко рассматриваются рекуррентные нейронные сети и двунаправленные рекуррентные нейронные сети. Также описано условное случайное поле. В том числе рассматриваются векторные представление в виде модели ELMo, двунаправленная языковая модель и ее тонкая настройка, а также приведена схема архитектуры двунаправленной рекуррентной нейронной сети с CRF.
Ключевые слова: распознавание именованных сущностей, нейронная сеть, векторное представление слов, языковая модель, условное случайное поле, архитектура нейронной сети, рекуррентная нейронная сеть, обработка естественного языка, двунаправленная рекуррентная нейронная сеть.

Введение

В области обработки естественного языка решается много задач и используются различные подходы. В настоящее время очень популярны методы с использованием глубоких нейронных сетей, поскольку они позволяют достичь наиболее лучших результатов. Важной частью в решении задач обработки естественного языка является выбор векторного представления слов. Цель данной работы - разработать эффективный метод для решения задачи распознавания именованных сущностей в русском языке.

Рекуррентная нейронная сеть

Для решения задачи распознавания именованных сущностей применяются такие нейронные сети, как рекуррентные. Рекуррентная нейронная сеть - это, по сути, полностью подключенная нейронная сеть, которая содержит преобразование некоторых своих слоев в петлю. Этот цикл обычно представляет собой итерацию сложения или объединения двух входных данных, умножения матриц и нелинейной функции. Рекуррентные нейронные сети отлично подходят для решения задач, в которых последовательность важна в большей степени, чем отдельные элементы. Данного вида сети очень неэффективны в ситуациях, когда необходимо учитывать предыдущую информацию из последовательности для расчёта

текущего выхода. Это происходит из-за влияния скрытого состояния или входа с шага t на последующие состояния рекуррентной сети. Сеть экспоненциально затухает. Данная проблема была решена при помощи специальной архитектуры рекуррентной нейронной сети, названной Long Short-Term Memory (LSTM) [1]. LSTM представляет собой ячейку памяти, которая содержит следующие компоненты: candidate cell state (1), input gate (2), forget gate (3), output gate (4), cell state (5), block output (6).

$$c'_t = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_{c'}), \quad (1)$$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \quad (2)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \quad (3)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot c'_t, \quad (5)$$

$$c'_t = o_t \odot tanh(c_t), \quad (6)$$

где x_t – входной вектор во время t, h_t – вектор скрытого состояния во время t, W_x (с разными вторыми индексами) – матрицы весов, применяющиеся ко входу, W_h – матрицы весов в рекуррентных соединениях, b – векторы свободных членов.

Двунаправленная рекуррентная нейронная сеть

Для того, чтобы правильно распознавать именованную сущность в предложении, необходимо учитывать контекст слова. Поскольку и следующие, и предыдущие слова имеют значение, будем использовать двунаправленную рекуррентную нейронную сеть. Вычисление контекста в

данной модели происходит в три этапа: рассчитываем состояние s_t (7) слева направо и состояние s'_t (8) справа налево, а затем соединяем их в один результат y_t (9) на уровне выхода.

$$s_t = \sigma(b + Ws_{t-1} + Ux_t), \quad (7)$$

$$s'_t = \sigma(b' + W's'_{t+1} + U'x'_t), \quad (8)$$

$$y_t = h(c + Vs_t + V's'_t), \quad (9)$$

где U – матрица весов для входов, W – матрица рекуррентных весов, V – матрица весов для выходов.

Условные случайные поля

Русский язык весьма богат как грамматически, так и морфологически. Исходя из этого, для более точного распознавания именованных сущностей, добавим к двунаправленной рекуррентной нейронной сети CRF-модель с выходными весами этой нейронной сети.

Условные случайные поля (Conditional Random Fields, CRF) — дискриминативная ненаправленная вероятностная графическая модель, являющаяся разновидностью марковских случайных полей (Markov random field). Данная модель основывается на методе скрытых марковских моделей (HMM) и методе моделей максимальной энтропии (MaxEnt) [2]. CRF рассматривает условное распределение $(y \mid x)$ последовательности токенов $y \in Y$, где вектор $x \in X$ состоит из наблюдаемых элементов. Из наблюдаемых и выходных элементов конструируется набор бинарных функций-признаков (feature functions, potential functions, factors), которые могут задаваться произвольно и включать в себя любое количество элементов.

Условным случайным полем называется распределение вида:

$$p(y|x) = \frac{1}{Z(X)} \prod_i e^{\sum_i \lambda_i f_i(y_t, y_{t-1}, x_t)}, \quad (10)$$

где f – функция-признак, λ – множитель Лагранжа, Z – коэффициент нормализации.

Векторное представление слов

Для решения задачи обработки естественного языка, в которую входит и задача распознавания именованных сущностей, обязательным компонентом нейронной сети является векторное представление слов. Мы будем использовать векторы, полученные из двунаправленного LSTM, который обучается с помощью связанной языковой модели на большом корпусе текста. Такая модель носит название ELMo (Embeddings from Language Models). Данное векторное представление будет использовать двунаправленную языковую модель (рис.1).

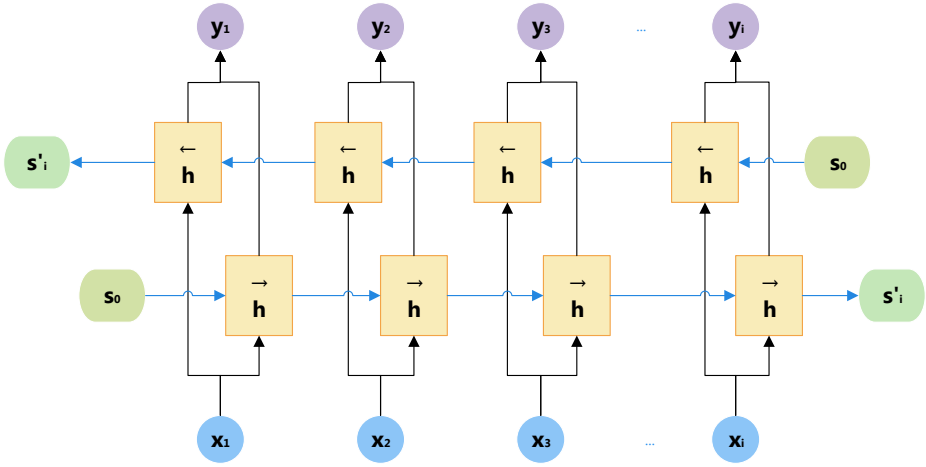

Рисунок 1 – Архитектура двунаправленной языковой модели.

Двунаправленная языковая модель

Рассмотрим двунаправленную языковую модель. Дана последовательность из N токенов (t_1, t_2, ..., t_N), модель прямого языка вычисляет вероятность этой последовательности путем моделирования вероятности токена t_i с учетом истории (t_1, ..., t_{i-1}):

$$p(t_1, t_2, \dots, t_N) = \prod_{i=1}^{N} p(t_i | t_1, t_2, \dots, t_{i-1}) \quad (11)$$

Современные нейронные языковые модели [3] рассчитывают контекстно-независимое представление токена x_i^{LM} (с помощью вложений токенов или свёрточной нейронной сети), затем его пропускают через L слоев прямой рекуррентной нейронной сети LSTM. В каждой позиции k каждый слой LSTM выводит контекстно-зависимое представление $\vec{h}_{i,j}^{LM}$, где $j = 1, \ldots, L$. Выход верхнего слоя LSTM $\vec{h}_{i,j}^{LM}$ используется для предсказания следующего токена t_{k+1} со слоем Softmax.

Обратная языковая модель похожа на прямую языковую модель, за исключением того, что она работает над последовательностью в обратном порядке, предсказывая предыдущий токен с учетом будущего контекста:

$$p(t_1, t_2, \ldots, t_N) = \prod_{i=1}^{N} p(t_i | t_{i+1}, t_{i+2}, \ldots, t_N) \quad (12)$$

Он может быть реализован аналогично прямой языковой модели, причем каждый обратный LSTM-слой j в L-слойной глубокой модели производит представления $\vec{h}_{i,j}^{LM}$ заданного токена t_i по следующим токенам ($t_{i+1}, \ldots, t_N$).

Двунаправленная языковая модель объединяет в себе как прямую, так и обратную языковую модель. Данная формулировка совместно максимизирует логарифмическую вероятность прямого и обратного направлений [4]:

$$\sum_{i=1}^{N} (\log p(t_i | t_1, \ldots, t_{i-1}; \Theta_x, \vec{\Theta}_{LSTM}, \Theta_s) + \log p(t_i | t_{i+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s)) \quad (13)$$

Мы объединяем параметры как для представления токенов (Θ_x), так и для слоя Softmax (Θ_s) в прямом и обратном направлении, сохраняя при этом отдельные параметры для LSTM в каждом направлении. В целом данная

формулировка подобна подходу [5], за исключением того, что мы распределяем некоторый вес между направлениями вместо использования абсолютно независимых параметров.

Перед тем, как данная двунаправленная языковая модель будет использоваться в ELMo, необходимо произвести ее тонкую настройку. Для этой настройки мы будем использовать подход, представленный в этой работе [6].

Дискриминативная тонкая настройка языковой модели

Поскольку различные слои языковой модели взаимодействуют с различными типами информации, они должны быть настроены в разной степени [7]. Исходя из этого, будем использовать дискриминативную тонкую настройку.

Вместо того, чтобы использовать одну и ту же скорость обучения для всех слоев модели, дискриминативная тонкая настройка предоставляет нам возможность настраивать каждый слой с отличными друг от друга скоростями обучения. Для контекста регулярное обновление параметров модели θ стохастическим градиентным спуском (SGD) на временном шаге t выглядит следующим образом [8]:

$$\theta_t = \theta_{t-1} - \eta \cdot \nabla_\theta J(\theta), \quad (14)$$

где η – скорость обучения, $\nabla_\theta J(\theta)$ – градиент по отношению к целевой функции модели.

Для дискриминативной тонкой настройки мы разбиваем параметры θ на $\{ \theta^1, \ldots, \theta^L \}$, где θ^L содержит параметры модели на L-слое, L – количество слоев. Аналогично получаем $\{ \eta^1, \ldots, \eta^L \}$, где η^L – скорость обучения L-слоя. Обновление стохастическим градиентным спуском с дискриминационной тонкой настройкой тогда выглядит следующим образом:

$$\theta_t^L = \theta_{t-1}^L - \eta^L \cdot \nabla_{\theta^L} J(\theta), \quad (15)$$

Практика показывает, что при выборе скорости обучения на последнем слое η^L , скорость обучения на предыдущих слоях рассчитывается следующим образом:

$$\eta^{L-1} = \frac{\eta^L}{2.6} \quad (16)$$

Для адаптации ее параметров к особенностям конкретной задачи мы хотели бы, чтобы модель быстро сходилась к подходящей области пространства параметров в начале обучения, а затем уточняла свои параметры. Использование одной и той же скорости обучения (LR) или отожженной скорости обучения на протяжении всего обучения - не лучший способ добиться такого поведения. Вместо этого мы используем наклонные треугольные скорости обучения (STLR), которые сначала линейно увеличивают скорость обучения, а затем линейно уменьшают ее в соответствии со следующим графиком обновления, который можно увидеть на рис. 2

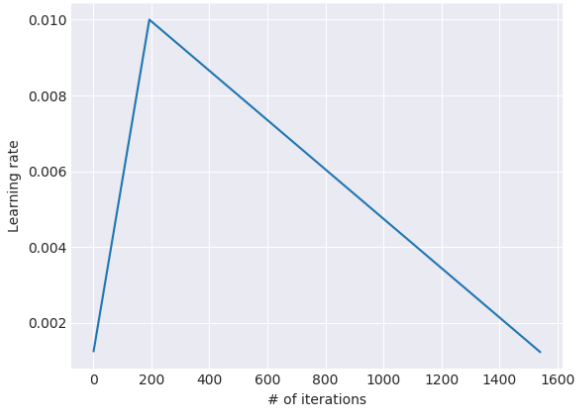

Рисунок 2 – График наклонных треугольных скоростей обучения.

Вместо тонкой настройки всех слоев сразу, что чревато катастрофическим забыванием, мы будем постепенно размораживать модель, начиная с последнего слоя, поскольку он содержит наименьшее общее знание: мы сначала размораживаем последний слой и точно настраиваем все

размороженные слои на одну эпоху. Затем мы размораживаем следующий нижний замороженный слой и повторяем, пока не настроим все слои до сходимости на последней итерации. Это похоже на " chain-thaw " [9], за исключением того, что мы добавляем слой за слоем к набору "оттаявших" слоев, а не только тренируем один слой за раз.

ELMo

Полученная тонко настроенная двунаправленная языковая модель будет использоваться в модели ELMo. ELMo - это специфичная комбинация задач промежуточного звена слоя в двунаправленной языковой модели. Для каждого токена t_i, а L-слой двунаправленной языковой модели вычисляется набором представлений $2L + 1$ (17).

$$R_i = \{x_i^{LM}, \vec{h}_{i,j}^{LM}, \overleftarrow{h}_{i,j}^{LM} \mid j = 1, \dots, L\} = \{h_{i,j}^{LM} \mid j = 0, \dots, L\}, \quad (17)$$

где $h_{i,0}^{LM}$ – слой токенов, $h_{i,j}^{LM} = [\vec{h}_{i,j}^{LM}; \overleftarrow{h}_{i,j}^{LM}]$ для каждого слоя двунаправленной рекуррентной нейронной сети.

Для включения в нижестоящую модель ELMo сворачивает все слои в R в один вектор $ELMo_i = E(R_i, \Theta_e)$. В простейшем случае ELMo просто выбирает верхний слой $E(R_i) = h_{i,L}^{LM}$. В более общем плане вычисляется удельный вес задачи для всех слоев двунаправленной языковой модели:

$$ELMo_i^{task} = E(R_i; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{i,j}^{LM}, \quad (18)$$

где s^{task} – softmax-нормализованные веса, γ^{task} – скалярный параметр, позволяющий модели задачи масштабировать весь вектор ELMo. Учитывая, что активации каждого слоя двунаправленной языковой модели имеют различное распределение, в некоторых случаях также помогает применение нормализации слоя к каждому слою языковой модели перед взвешиванием [10].

Большинство контролируемых моделей обработки естественного языка имеют общую архитектуру на самых нижних уровнях, что позволяет нам добавлять ELMo последовательным, унифицированным образом. Учитывая последовательность токенов (t_1, ..., t_N), нужно стандартно формировать контекстно-независимое представление токена x_i для каждой позиции токена с использованием предварительно обученных вложений слов и, возможно, символьных представлений. Затем модель формирует контекстно-зависимое представление h_i, используя двунаправленные рекуррентные нейронные сети.

Чтобы добавить ELMo в контролируемую модель, сначала заморозим веса двунаправленной языковой модели, а затем объединим вектор ELMo $ELMo_i^{task}$ с x_i и предадим расширенное представление ELMo в двунаправленную рекуррентную нейронную сеть. Архитектура двунаправленной рекуррентной нейронной сети с CRF - слоем и с предобученным векторным представлением в виде ELMo предоставлена на рис. 3. В итоге полученная модель обработки запроса пользователя должна определить, к какой категории относится то или иное слово в сообщении, полученном от пользователя.

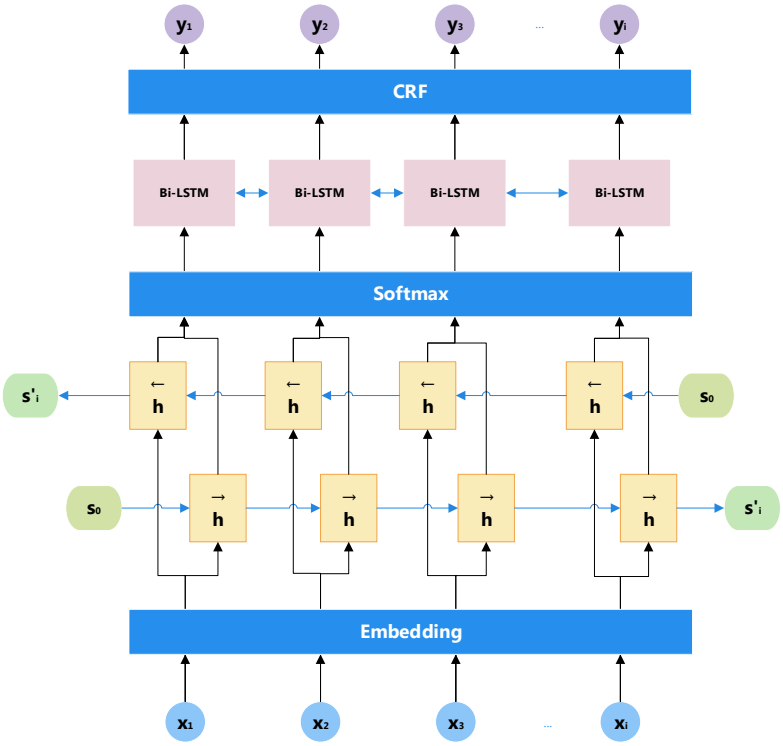

Рисунок 3. – Архитектура двунаправленной рекуррентной нейронной сети с CRF слоем.

Результаты

Оценим эффективность полученной модели с помощью метрики «Точность и полнота». В тесте участвовало 20 слов, у которых необходимо было определить принадлежность к сущности, из них 10 слов относилось к сущности «Категория мебели», 5 – к сущности «Цвет» и 5 – к сущности «Размер». 7 из 10 слов сущности «Категория мебели» были распознаны верно. 3 из 5 слов сущности «Цвет» были распознаны верно. И 4 из 5 слов сущности «Размер» были верно распознаны.

$$Precision = \frac{TP}{TP+FP}, \tag{19}$$

$$Recall = \frac{TP}{TP+FN}, \tag{20}$$

где TP – истинно-положительное решение, FP – ложно-положительное решение, FN – ложно-отрицательное решение.

Точность рассчитывается по формуле (19). И получается, что точность нашей модели равна 0.67, то есть 67 %. Полнота рассчитывается по формуле (20). И получается, что полнота нашей модели равна 0.7, то есть 70 %.

Вывод

Таким образом, для лучшего качества распознавания именованных сущностей в русском языке необходимо использовать не просто рекуррентную нейронную сеть, а двунаправленную рекуррентную нейронную сеть с добавлением слоя условных случайных полей. А также использовать предобученное векторное представление ELMo, так как при помощи данной модели используется глубокое контекстно-зависимое представление слова. Кроме того, используется тонкая настройка двунаправленной языковой модели для более точных предсказаний. Полученная модель будет эффективно решать задачи распознавания именованных сущностей в русском языке.

Литературы

1. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. - СПб. Питер, 2018. - 480 с.
2. А. Ю. Антонова, А. Н. Соловьев Использование метода условных случайных полей для обработки текстов на русском языке // Компьютерная лингвистика и интеллектуальные технологии. - 2013. - №14. - С. 321-325.
3. Jozefowicz R. et al. Exploring the limits of language modeling // arXiv preprint. URL: arxiv.org/abs/1602.02410 (дата обращения: 10.01.2021).
4. Peters M. E. et al. Deep contextualized word representations // arXiv preprint. URL: arxiv.org/abs/1802.05365 (дата обращения: 10.01.2021).
5. Peters M. E. et al. Semi-supervised sequence tagging with bidirectional language models // arXiv preprint. URL: arxiv.org/abs/1705.00108 (дата обращения: 11.01.2021).

6. Howard J., Ruder S. Fine-tuned Language Models for Text Classification // arXiv preprint. URL: arxiv.org/abs/1801.06146v1 (дата обращения: 10.01.2021).

7. Yosinski J. et al. How transferable are features in deep neural networks? // arXiv preprint. URL: arxiv.org/abs/1411.1792 (дата обращения: 11.01.2021).

8. Ruder S. An overview of gradient descent optimization algorithms // arXiv preprint URL: arxiv.org/abs/1609.04747 (дата обращения: 11.01.2021).

9. Felbo B. et al. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm // arXiv preprint. URL: arxiv.org/abs/1708.00524 (дата обращения: 9.01.2021).

10. Ba J. L., Kiros J. R., Hinton G. E. Layer normalization // arXiv preprint. URL: arxiv.org/abs/1607.06450 (дата обращения: 9.01.2021).

Referents

1. Nikolenko S., Kadurin A., Arkhangel′skaya E., Glubokoye obucheniye [Deep Learning]. SPb. Piter, 2018. 480 p.

2. Antonova A. YU. , Solov'ev A. N., Komp'yuternaya lingvistika i intellektual'nye tekhnologii [Computational linguistics and intelligent technologies]. 2013. pp. 321-325.

3. Jozefowicz R. et al. arXiv preprint. URL: arxiv.org/abs/1602.02410 (Data access: 10.01.2021).

4. Peters M. E. et al. arXiv preprint. URL: arxiv.org/abs/1802.05365 (Data access: 10.01.2021).

5. Peters M. E. et al. arXiv preprint. URL: arxiv.org/abs/1705.00108 (Data access: 11.01.2021).

6. Howard J., Ruder S., arXiv preprint. URL: arxiv.org/abs/1801.06146v1 (Data access: 10.01.2021).

7. Yosinski J. et al. arXiv preprint. URL: arxiv.org/abs/1411.1792 (Data access: 11.01.2021).

8. Ruder S. arXiv preprint. URL: arxiv.org/abs/1609.04747 (Data access: 11.01.2021).

9. Felbo B. et al. arXiv preprint. URL: arxiv.org/abs/1708.00524 (Data access: 9.01.2021).

10. Ba J. L., Kiros J. R., Hinton G. E. arXiv preprint. URL: arxiv.org/abs/1607.06450 (Data access: 9.01.2021).