Changellenge CUP-IT 2021
Секция Data Science

Задача:

- Решить задачу Text Entailment на датасете SNLI

В чем суть задачи:

- Нам даются основная строка, и строка на проверку. Мы знаем, что основная строка - истина. Нужно проверить, не противоречит ли ей строка на проверку.
- Датасет для обучения - так же две строки как параметры, как результат - пять независимых оценок, как именно строки взаимосвязаны.
- E - одна следует из другой (1 -> 2)
- C - строки противоречат (1 != 2)
- N - означает что нет ни contradiction(C), ни entailment(E)

Какие подходы можно использовать?

Подходы:

- CNN
- RNN(LSTM, biLSTM)
- Механизм внимания(BERT и др.)

Что можно использовать в качестве embedding layer:

- Word2vec/fastText
- ELMo

Проведем ресерч по каждому подходу из открытых источников

CNN: Описание, достоинства и недостатки

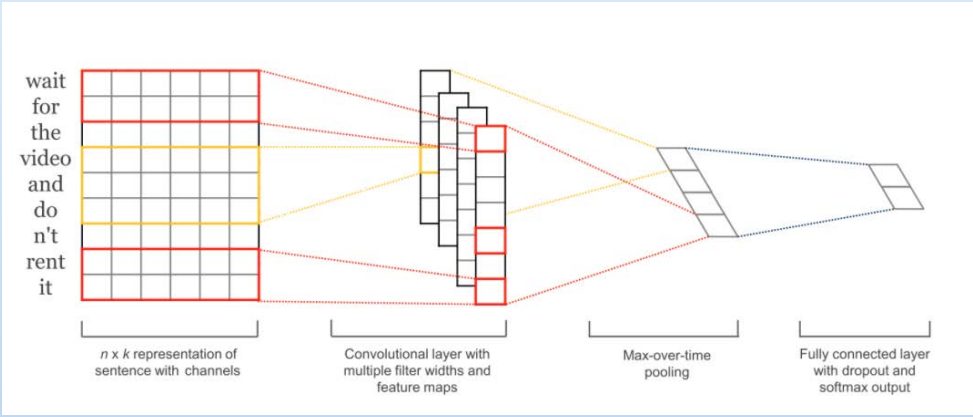
Описание: CNN для тескта работает аналогично CNN для изображения: на таблицу эмбедингов слов накладывается фильтр(ядро свертки), работая как «детектор» определенных сочетаний слов, что позволяет нам учитывать контекст слова

Достоинства:

- Хорошо подходят для выявления паттернов, вне зависимости от их позиции(инвариантность к переносу в пространстве)
- Быстрые, простые, хорошо учатся

Недостатки:

- Недостаточно гибкие и мощные, чтобы уметь находить широкие паттерны(например сравнивать первое и последнее слово, игнорируя при этом слова между ними)
- Ограничены в ширине контекста(Чтобы увеличить максимальную длину паттерна, нужно существенно увеличить число параметров нейросети)



RNN: Описание, достоинства и недостатки

Идея:

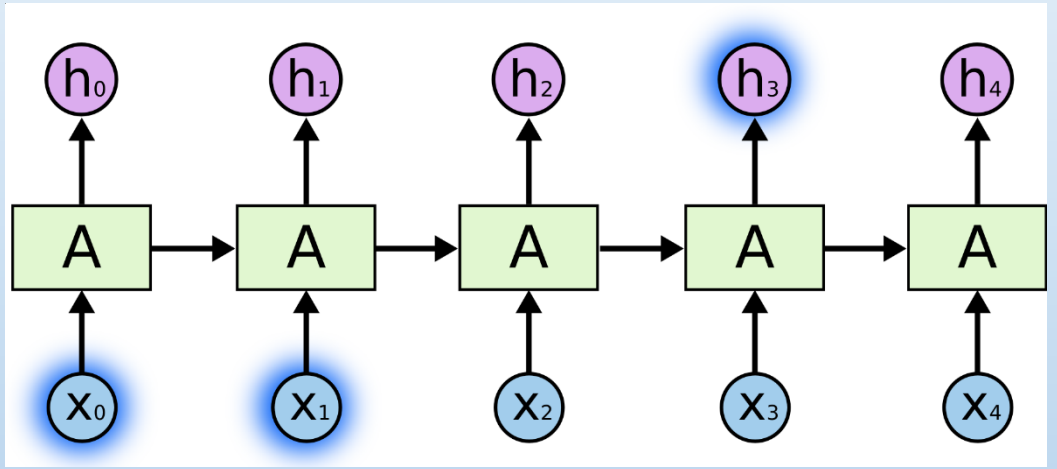
- Последовательная обработка текста – слово за словом
- На каждом шаге поддерживается и обновляется вектор скрытого состояния

Достоинства:

- Могут учитываться длинные зависимости (длина зависимости слабо связана с числом параметров)

Недостатки:

- Медленные, сложно учатся (из-за того что нельзя распараллелить вычисления на GPU)
- Проблема затухающего градиента (Решается использованием LSTM)
- Учитывается только контекст слева (Решается использованием двунаправленных LSTM или другими словами biLSTM)



Механизм внимания(attention mechanism, self-attention)

Описание: Механизм внимания осуществляет попарное сравнение элементов двух последовательностей (или одной последовательности с самой собой) и позволяет выбрать только наиболее значимые элементы, чтобы продолжить работу только с ними. Можно рассматривать как умный, адаптивный пулинг.

Идея:

- Попарное сравнение элементов последовательности
- Умная агрегация

Достоинства:

- Учет сколь угодно далеких зависимостей
- Быстрый, хорошо учится

Word2vec в качестве embedding layer

Идея:

- Получаем эмбеддинги слов, моделируя распределения вероятностей соседних слов
- Word2vec работает с локальным контекстом(окном небольшой ширины)
- Подразделяется на SkipGram(предсказание соседних слов по заданному слову) и CBOW(предсказание заданного слова по соседним)

Достоинства:

- Простота
- Сжатое представление, обобщение
- Работа с большими словарями

Недостатки:

- Не работает с неизвестными словами(решение – FastText = Word2vec на уровне целых слов и N-грамм, если они достаточно частотные)
- Не учитывает сложного контекста (Одно слово – один смысл)

Elmo в качестве embedding layer

Идея:

- Скрытые состояния со всех шагов и со всех слоёв biLSTM используются как признаки(векторные представления токенов) для решения прикладной задачи
- Причем нижние слои biLSTM будут отвечать за синтаксис, а верхние – за смысл слова

A biLM combines both a forward and backward LM. Our formulation jointly maximizes the log likelihood of the forward and backward directions:

$$\sum_{k=1}^{N} \left(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \right).$$

each token t_k, a L-layer biLM computes a set of $2L + 1$ representations

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}, \end{aligned}$$

where $\mathbf{h}_{k,0}^{LM}$ is the token layer and $\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$, for each biLSTM layer.

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

Преимущество ELMo в том, что она учитывает сложный контекст, в отличие от word2vec

Какой подход использовать?

- Исходя из вышеперечисленных достоинств и недостатков каждого из подходов, можно попробовать использовать следующие:

1) Elmo + biLSTM слои
2) Модель с механизмом внимания(например BERT)

Нами было принято решение реализовать первый способ.

Точность получилась около 75%, однако можно улучшить результат, потратив больше времени на обучение, а также усложнив модель(добавить больше слоёв и т.д.)

Наше решение

Python Notebook с нашим решением, а также файл results.csv есть в архиве

Также дублирую наш код ссылкой на гитхаб

Сохраненные веса модели

Ссылки на источники, которыми мы пользовались

- Курс «Нейронные сети и обработка текста» - https://stepik.org/course/54098/syllabus
- Лекция 22. ELMo & BERT - https://www.youtube.com/watch?v=Q4HVS6c92qU
- Word2vec в картинках - https://habr.com/ru/post/446530/
- NLU по-русски: ELMo vs BERT - https://habr.com/ru/company/mipt/blog/478358/
- BERT, ELMO и Ко в картинках (как в NLP пришло трансферное обучение) - https://habr.com/ru/post/487358/
- Применение сверточных нейронных сетей для задач NLP - https://habr.com/ru/company/ods/blog/353060/
- Рекуррентные нейронные сети (RNN) с Keras - https://habr.com/ru/post/487808/

Информация об участниках команды «Аналитики на час»

- Алексей Сергеев (Капитан команды) – lehasergeev201096@gmail.com
- Максим Петров - 89250609951@yandex.ru
- Ашот Маргарян - margaryan.ash31@gmail.com
- Никита Яшин - niknlo007@gmail.com

Спасибо за внимание!