ADSC
Applied Data Science Center

Тренды в анализе данных. Статья 5

О цикле

Данный цикл статей основан на семинарских занятиях открытого курса Data Mining in Action по направлению “тренды в анализе данных”. На семинарах (и в статьях по ним) мы будем говорить о последних достижениях в области data science.
Общую лекцию для всего потока можно посмотреть здесь, а презентацию к ней скачать здесь.

Сегодня в статье:

1. Машинный перевод
2. Классическая модель seq2seq
3. Seq2seq + attention
4. Attention
5. Без обучающей выборки
6. Transfer learning

Машинный перевод

Машинный перевод - достаточно старая и важная задача, в которой мы хотим получать на вход текст из одного языка и переводить на другой язык. Алгоритмы машинного перевода используют многие пользователи и компании, в недавнем времени появилась тенденция переводить необходимую компаниям документацию автоматически, а потом сажать людей, менее квалифицированных, чем переводчики, править ошибки модели. Область быстро растет и развивается. Наша задача - взять на вход последовательность слов одного языка и сгенерить последовательность слов из другого языка.

Оценка качества

Допустим, мы построили некий черный ящик, который умеет брать предложение на одном языке и переводить на другой. Также у нас есть набор тестовых эталонных переводов. Мы хотим понять, правильно ли мы все сделали. Тут есть две группы способов:
1) Люди. Самый очевидный, но и самый ресурсоемкий способ - посадить людей смотреть на результат машинного перевода. Часто в таких исследованиях люди проставляют оценки от 0 до 9 информативности - отображает ли перевод смысл оригинала, и разборчивости - насколько это

адекватно читать. Такой прямой метод - это долго и дорого, поэтому хочется, как и везде, обойтись без людей.

2) Автоматический расчет. Для того, чтобы быстро и бесплатно все считать, был придуман набор специальных метрик. Самая популярная из них BLEU - усреднение встречаемости n-gram между истинным и нашим переводом. Например, у нас есть предложение истинного перевода (референса) ‘I want eat’ и сгенерированного перевода (кандидата) ‘I eat’. Сначала посчитаем количество совпавших униграмм в кандидате по отношению к референсу. Оба слова из кандидата ‘I’ и ‘eat’ встречаются в референсе. Среднее по ним - 1. Перейдем к биграмме ‘I eat’. Она не встречается ни разу - среднее 0. Далее сложим логарифмы полученных значений (вместо логарифма от 0 берем 0) со весами, которые исторически определились из соображений корреляции метрики с оценкой людей. Например, было установлено, что 4-граммы особенно сильно коррелируют с оценкой людей, поэтому среднему от 4-грамм дается больший вес. BLEU хорошо коррелирует с оценками людей, но есть более продвинутые метрики, которые учитывают ее недостатки. Например, NIST вознаграждает модель за корректный перевод редких слов, как и UBLEU, которые вознаграждают еще и за за корректный перевод синонимов. Есть и другие идеи, которые легли в основу метрик, например, TER (Translation Edit Rate): здесь мы считаем минимальное количество исправлений, которые нужно совершить для получения корректного перевода, нормированный на размер предложения. Подробней о метриках с формулами, преимуществами и недостатками можно почитать здесь.

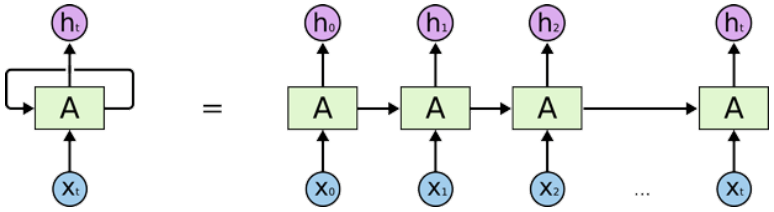
Классическая модель seq2seq

Seq2seq, как легко понять из названия, - это архитектура, позволяющая переводить одну последовательность в другую. Один из стандартных способов обработки последовательностей - это RNN.

RNN

Рекуррентные сети умеют брать на вход последовательность слов произвольной длины и делать из них один вектор, учитывая очередность слов и запоминая важную информацию по ходу последовательности. Делается это так: слово преобразуется в вектор (в простом случае - с помощью one-hot encoding, в сложном - это вектор, полученный с помощью использования семантической информации из большого набора текстов, например, word2vec, fasttext). Затем представление каждого вектора (x на рисунке) передается в ячейку рекуррентной сети вместе с вектором состояния от прошлого слова - памяти от каждой предыдущей ячейки, в которой хранится важная информация от всей предыдущей последовательности. Для первого слова нет

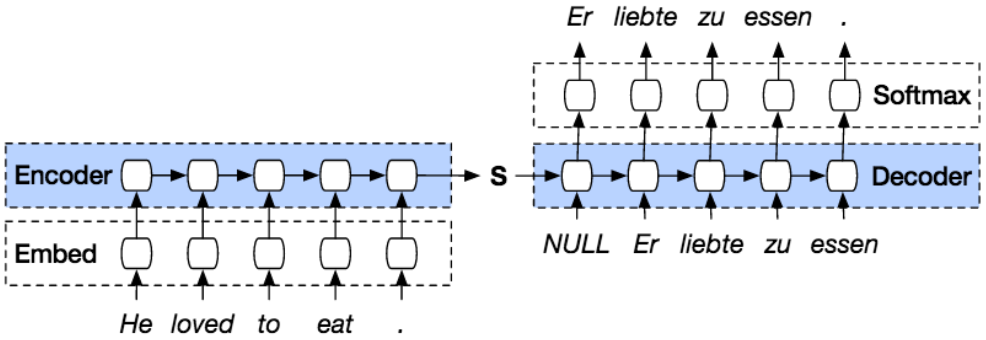
предыдущего, поэтому этот вектор инициализируется случайно. На выходе из ячейки выдается некое предсказание, например, вероятность текущего слова при условии всех предыдущих (h на рисунке):



Сеть вычисляет, что нужно запомнить, а что забыть, преобразует эту концепцию в вектор скрытого состояния и передает его дальше.
Часто используют двунаправленные рекуррентные сети, где мы добавляем скрытые состояния, которые передаются в обратном направлении.

Seq2seq

Вернемся к архитектуре seq2seq. Seq2seq состоит из энкодера и декодера.
Энкодер отвечает за то, чтобы представлять входящую информацию от текста, которое мы переводим, декодер - за то, чтобы генерировать перевод. Во время обучения энкодер берет на вход последовательность слов из одного языка и генерирует вектор, который представляет всю эту последовательность. Во время обучения и применения энкодер ведет себя одинаково.
Архитектура seq2seq:



Для предсказания каждого слова перевода, декодер берет на вход вектор из энкодера, предыдущее слово перевода и выдает вероятность выдать текущее слово. Во время обучения мы оптимизируем правдоподобность - вероятность перевода при условии оригинального предложения. Во время применения и обучения модели декодер работает по-разному. Разница между декодером на стадии обучения и применения заключается в следующем: на стадии обучения декодер предсказывает вероятность текущего слова при условии предыдущих с учетом слов, которые действительно были до этого. На стадии применения

декодер основывается на своих же предсказаниях. Часто это бывает проблемой: если мы предсказали неправильное слово, то потом мы тащем эту ошибку на все слова после этого. Частично исправить эту проблему позволяет beam-search; о нем речь пойдет в следующем разделе.

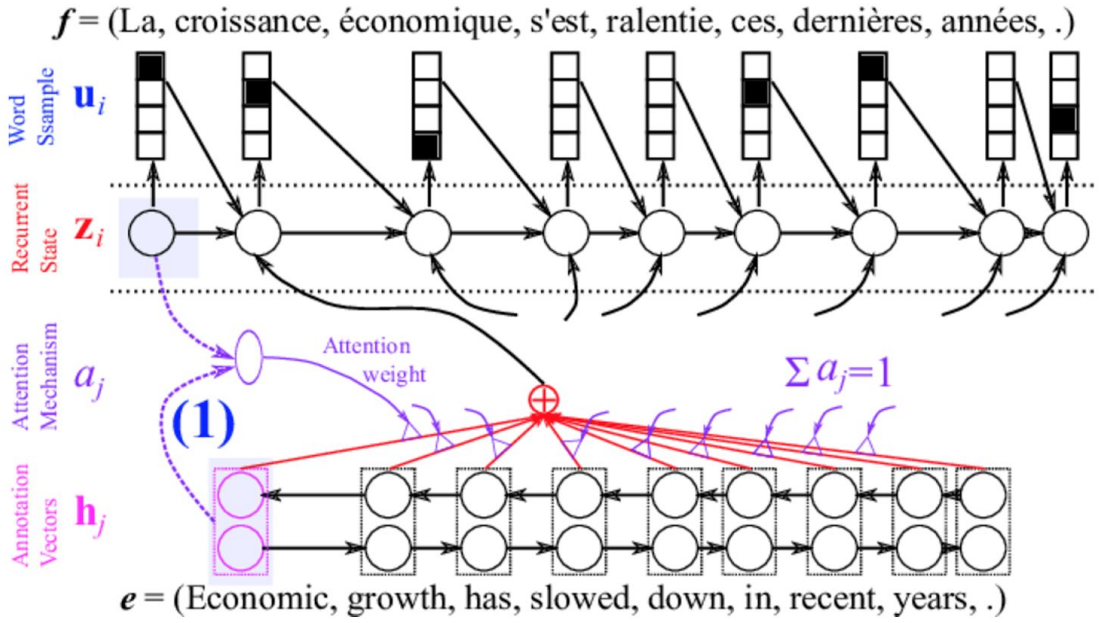
Decoder и beam-search

Итак, на этапе применения модели, когда мы предсказываем слово на выходе из декодера, можно просто выбрать слово, которому сеть дала наибольшую вероятность. Однако это не вполне хорошо. Приведем пример. Мы переводим предложение “*Хочу что-то понять в этой статье*” с русского на английский. Допустим, слово ‘хочу’ сеть перевела, как ‘want’ с вероятностью 0.44, и как ‘should’ с вероятностью 0.45. Тогда мы выберем на первом шаге первым же словом неправильный перевод, что потянет за собой череду ошибок, хотя истина была близко. А если мы оставим про запас несколько самых вероятных вариантов, посчитаем вероятности следующих слов, опять оставим несколько вероятных вариантов, а потом выберем самую вероятную в совокупности последовательность слов, то качество перевода заметно улучшится. Эта идея носит название beam-search и активно применяется в машинном переводе.

Seq2seq + attention

Классическая архитектура seq2seq хороша, но имеет существенный минус - если предложение, которое мы хотим переводить, достаточно длинное, то одного вектора на выходе из энкодере нам может быть мало, чтобы хорошо восстановить всю информацию, которая содержится в оригинале текста. Также часто в разных языках (например, в английском и русском) используется разный порядок слов в предложении. Возьмем, к примеру, предложение на русском “*На улице холодно*” и его перевод на английский ‘*It’s cold outside*’. Если при предсказании слов мы используем стандартную модель seq2seq, то информация об “улице”, идущей в начале русского предложения, к английскому слову outside, который находится в конце перевода потеряется.
Поэтому разумно научить декодер не просто обращаться к финальному скрытому состоянию энкодера, а к взвешенной сумме скрытых состояний энкодера в каждом моменте времени: это дает нам заметно больше гибкости. Веса, с которыми будем складывать эти состояния мы учим отдельного для каждого слова в декодере. Нужно это для того, чтобы понимать, на какую часть предложения в энкодере стоит обращать внимание декодеру. Таким образом, attention позволяет нам обращать внимание на нужную часть предложения и предоставляет большую гибкость и большее количество полезной информации в решении задач машинного перевода. Пример того, как выглядит архитектура

attention в применении к энкодеру двунаправленной рекуррентной сети, можно увидеть на рисунке:



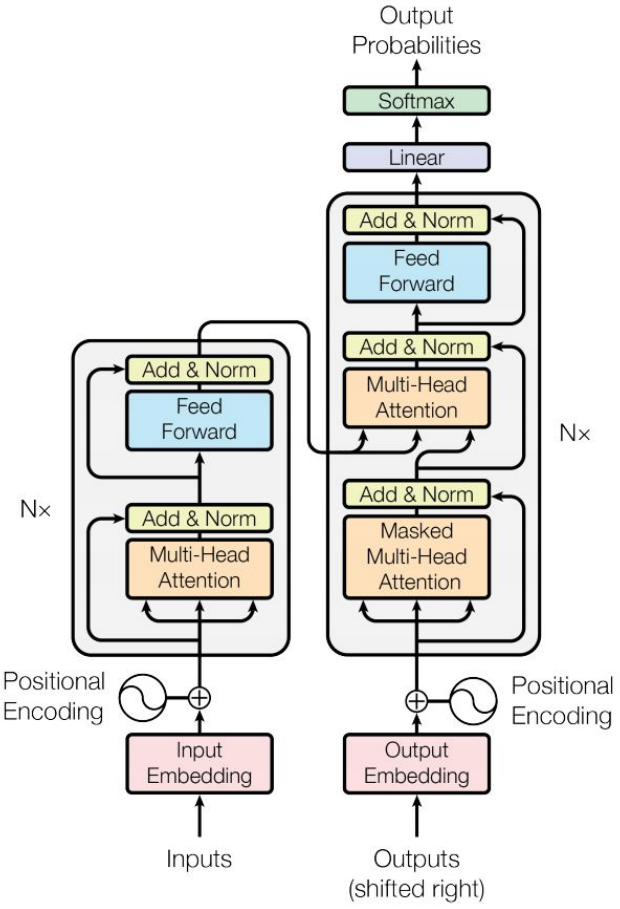
Attention

Заметим, что все обсуждаемые до этого архитектуры использовали RNN. У них есть один большой недостаток - их крайне сложно параллелить, так как нужно знать предыдущие скрытые состояния и вероятности слов, чтобы генерировать следующие слова. Хотелось бы придумать какую-то архитектуру, которая учитывала бы **взаимосвязь слов** (как RNN), но могла бы **эффективно параллелиться**.

На помощь приходит идея архитектуры Transformer, которая бьет SOTA (State of the art) по метрике BLEU на данной задаче. Идея:

Энкодер. На вход энкодеру по-прежнему подаются embeddings для слов языка оригинала, для каждого слова в embedding прибавляется еще и номер слова в последовательности (positional encoding). Мы хотим получить из них всех вектора для каждого слова, которые хорошо передают связь слов и особенности текста. Делаем это с помощью Multihead Attention. Для каждого входного слова считаем близость со всеми другими словами - это веса, с которыми мы прибавляем остальные слова к исходному, получая таким образом новый embedding для входного слова. Но похожесть бывает разная - лексическая, семантическая (внутри семантической есть также очень много смыслов), поэтому такие веса мы учим несколько раз, полученные суммарные вектора для всех весов пропускаем через сетку и получаем финальный embedding для каждого входного слова. Он отражает номер слова в последовательности и его взаимоотношения со всеми другими словами.

Embeddings для всех слов мы подаем на вход еще одной сети с residual connections и получаем финальный embedding для слов.
Декодер. Мы берем на вход embeddings для слов энкодера, применяем к ним архитектуру multihead attention, также берем предыдущие слова из декодера, также обработанные multihead attention и предсказываем следующее слово.
Подробнее про архитектуру Transformer можно прочитать здесь.
Так выглядит архитектура энкодера и декодера:



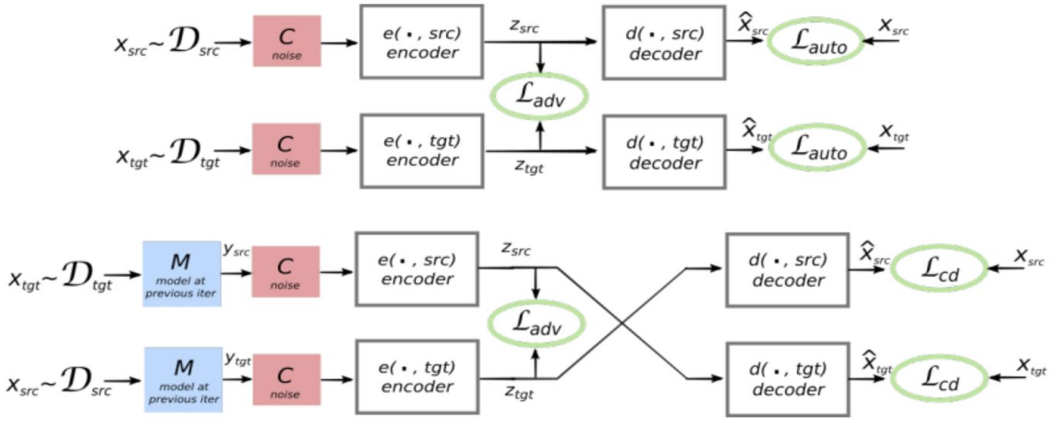
Без обучающей выборки

Теперь посмотрим на задачу машинного перевода совсем с другой стороны. Известно, что хорошая обучающая выборка для перевода есть далеко не для всех пар языков. Кроме этого, данные по переводу новых, специфичных текстов на профессиональные темы появляются гораздо более медленно, чем сами тексты. Поэтому в идеальном мире мы бы хотели уметь переводить набор текстов с одного языка на другой совсем без обучающей выборки.
Это и осуществили авторы статьи Unsupervised Machine Translation Using Monolingual Corpora Only , не слишком отстав по BLEU от методов обучения с учителем:

	Multi30k-Task1				WMT			
	en-fr	fr-en	de-en	en-de	en-fr	fr-en	de-en	en-de
Supervised	56.83	50.77	38.38	35.16	27.97	26.13	25.61	21.33
word-by-word	8.54	16.77	15.72	5.39	6.28	10.09	10.77	7.06
word reordering	-	-	-	-	6.68	11.69	10.84	6.70
oracle word reordering	11.62	24.88	18.27	6.79	10.12	20.64	19.42	11.57
Our model: 1st iteration	27.48	28.07	23.69	19.32	12.10	11.79	11.10	8.86
Our model: 2nd iteration	31.72	30.49	24.73	21.16	14.42	13.49	13.25	9.75
Our model: 3rd iteration	32.76	32.07	26.26	22.74	15.05	14.31	13.33	9.64

Table 1: **BLEU score on the WMT and Multi30k-Task1 datasets** using greedy decoding.

Основная идея алгоритма заключается в следующем. Допустим, мы хотим научиться переводить тексты с русского языка на английский. На входе у нас есть набор текстов на русском и английском без соответствия друг другу. Сделаем **два автоэнкодера**. Первый из которых умеет брать русский текст на вход, преобразовывать в сжатое латентное пространство, а из него уже восстанавливать тот же русский текст. Второй автоэнкодер будет делать то же самое для английского языка. Предположим, что есть некое о**бщее латентное смысловое пространство** для русского и английского, и будем пытаться приводить латентные пространства из русского и английского автоэнкодеров в это общее пространство. Для этого поставим на вход в эти пространства дискриминатор, который будет **штрафовать энкодер**, если у него получится определять, сжатое представление каких данных - русских или английских - пришло ему на вход. Когда мы добились того, что дискриминатор не может определить, русские или английские данные мы ему кормим, **поменяем местами декодеры** в этих автоэнкодерах и получим два переводчика сразу: с русского на английский и с английского на русский!



Transfer learning на задаче Machine Translation

В задачах машинного обучения на изображениях и текстах распространено использование предобученных моделей. Например, в компьютерном зрении часто используют тяжелые нейросети, обученные на огромных датасетах, которые извлекают признаки для картинок. Transfer learning переносит разработки с одной задачи на другую. В открытом доступе есть большое количество переведенных данных, на которых можно натренировать модель seq2seq и взять представления слов в энкодере как embeddings, которые хорошо представляют контекстную информацию. Это и сделали авторы статьи Learned in Translation: Contextualized Word Vectors, побив SOTA по ряду NLP задач (информация актуальна до выхода алгоритма ELMO в феврале 2018 года).

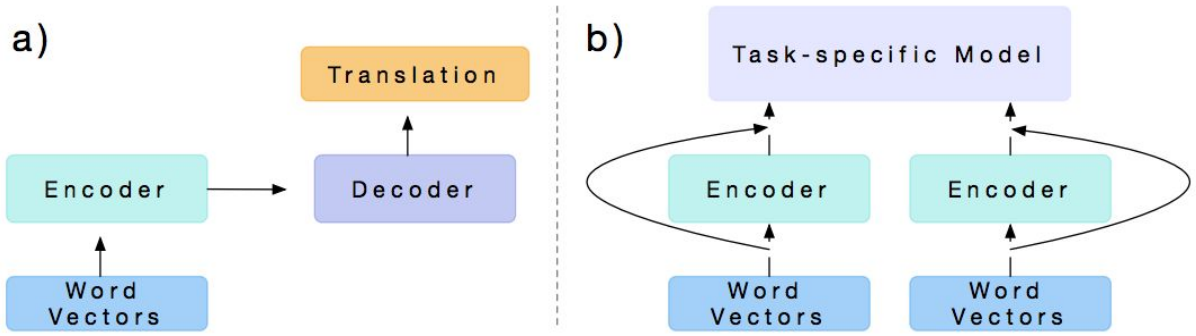


Figure 1: We a) train a two-layer, bidirectional LSTM as the encoder of an attentional sequence-to-sequence model for machine translation and b) use it to provide context for other NLP models.

Благодарим Татьяну Савельеву за помощь в написании статьи. Оставить отзыв по прочитанной статье вы можете здесь. По всем вопросам пишите на почту курса: dmia@applieddatascience.ru