Урок 7

Продвинутые методы анализа текстов

7.1. word2vec

Этот урок посвящён компьютерным методам анализа тектов и начинается с описания подхода word2vec. Этот метод позволяет представить каждое слово в виде числового вектора.

7.1.1. Похожие слова

Слова «идти» и «шагать» — это синонимы, они имеют похожий смысл. Однако для компьютера это просто строки, причём не очень похожие: у них разная длина, разные буквы. Просто по этим двум строкам компьютер не может сказать, что они имеют одинаковый смысл.

При анализе текстов хочется уметь оценивать, насколько похожи различные пары слов. Это можно сделать, используя текстовые данные. Оказывается, слова со схожим смыслом часто встречаются рядом с одними и теми же словами. Другими словами, похожие слова имеют одинаковые контексты. На этом принципе основан метод word2vec.

7.1.2. Векторные представления слов

Итак, требуется описать каждое слово w с помощью вектора $\vec{w}$ размерности d:

$$w \to \vec{w} \in \mathbb{R}^d.$$

При этом хочется иметь компактные представления слов, то есть величина d должна быть не очень большой. Также предполагается, что похожие слова должны иметь близкие векторы (например, по евклидовой или косинусной метрике). Наконец, к числовым векторам можно применять арифметические операции: складывать, вычитать, умножать на коэффициент. Хотелось бы, чтобы эти операции имели смысл. Как будет показано далее, полученные представления будут обладать всеми вышеперечисленными свойствами.

7.1.3. word2vec

При поиске вектора $\vec{w}$ работа будет происходить с вероятностями

$$p(w_i|w_j) = \frac{\exp(\langle \vec{w_i}, \vec{w_j} \rangle)}{\sum_w \exp(\langle \vec{w_i}, \vec{w} \rangle)}.$$

Эта функция показывает, насколько вероятно встретить слово $\vec{w_i}$ в контексте слова $\vec{w_j}$, то есть рядом.

Векторные представления слов будут настраиваться так, чтобы вероятности встретить слова, находящиеся в одном контексте, были высокими. В функционал для каждого слова входят вероятности встретить его вместе с k словами до и после него:

$$\sum_{i=1}^{n} \sum_{j=-k}^{k} \log p(w_{i+j}|w_i) \to \max.$$

Оптимизировать этот функционал можно с помощью стохастического градиентного спуска.

7.1.4. Свойства представлений

Если обучиться на большом корпусе текстов, то окажется, что векторы представлений имеют много интересных свойств. В частности, если слова похожи по смыслу, то соответствующие им векторы близки по косинусной метрике.

Что касается арифметических операций, то наблюдается следующее:

$$\vec{\text{king}} - \vec{\text{man}} + \vec{\text{woman}} \approx \vec{\text{queen}},$$

$$\vec{\text{Moscow}} - \vec{\text{Russia}} + \vec{\text{England}} \approx \vec{\text{London}}.$$

Более того, оказывается, такой подход можно использовать для перевода слов. Если обучить векторы одновременно на корпусе английских и испанских текстов, то

$$\vec{\text{one}} - \vec{\text{uno}} + \vec{\text{four}} \approx \vec{\text{quatro}}.$$

Этот метод проигрывает методам, специально разработанным для машинного перевода, тем не менее с его помощью удаётся получать интересные результаты.

Наконец, от представлений отдельных слов можно попытаться перейти к представлениям текстов. Оно получается усреднением векторов всех слов, входящих в документ. С полученным признаковым описанием целого текста также можно работать.

7.1.5. word2vec и обучение с учителем

Ранее уже обсуждалось, что проблема мешка слов состоит в том, что получается пространство признаков слишком большой размерности (десятки и сотни тысяч признаков). Если использовать подход word2vec, где размер вектора равен 100, то получится признаковое описание текста, размер которого также равен 100. При таком количестве признаков становится возможным использовать любые методы машинного обучения (например, случайные леса и градиентный бустинг), а не только линейные модели.

Таким образом, используя данный подход, можно значительно сжимать представления текстов, и обучать сложные методы машинного обучения.

7.2. Рекуррентные сети

Рекуррентные нейронные сети — это ещё один способ работать с текстами.

7.2.1. Мешок слов

Ранее были изучены два подхода к анализу текстов. Первый — мешок слов. В нём каждое слово — это отдельный признак. Порядок слов не учитывается, предполагается, что они независимы. Считается, что текст генерируется следующим образом. Существует некоторое распределение на словах, из него случайно выбирается слово и добавляется в текст. Могут быть и более сложные порождающие процессы, но основная идея не меняется: порядок слов никак не учитывается. Можно осуществить переход к n-граммам, skip-граммам, которые учитывают связи между словами, но общий порядок слов всё равно не принимается во внимание.

7.2.2. word2vec

Второй подход — это word2vec, обучение представлений слов. В нём для получения представления слова используется его контекст, то есть слова, рядом с которыми оно часто встречается. В этом случае появляется некоторый порядок. Однако признаковое описание текста получается усреднением признаковых описаний входящих в него слов, и порядок слов снова никак не учитывается.

7.2.3. Рекуррентная сеть

Возникает вопрос, существует ли подход, который использует текст именно в исходном виде, то есть учитывает порядок слов. Такие методы существуют, и один из них — это рекуррентные нейронные сети(рис. 7.1). Такая сеть устроена немного необычным образом. Она последовательно получается на вход слова. При этом в ней есть скрытый слой h, который обновляется после появления каждого нового слова или токена

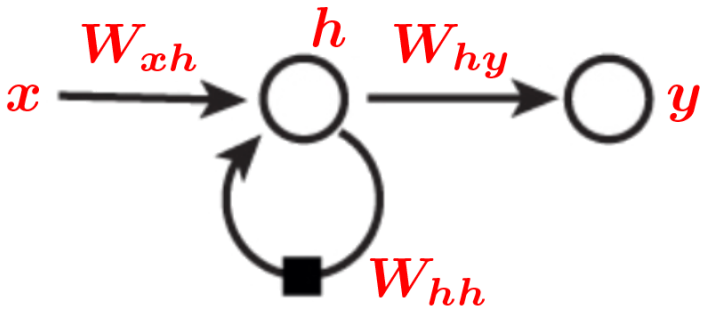

Рис. 7.1: Структура рекуррентной нейронной сети

(например, буквы). Слой обновляется, учитывая и своё предыдущее состояние, и полученное слово. После того как скрытое состояние обновлено, на его основе генерируется выход y для данного токена. Процесс повторяется до тех пор, пока через сеть не пройдут все слова в тексте.

Если говорить более строго, то данная сеть описывается двумя формулами. Первая — это обновление скрытого состояния. Оно обновляется путём некого усреднения предыдущего значения и нового входа:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1}),$$

где h_{t-1} — состояние слоя в предыдущий момент времени, x_t — слово, которое было подано на вход, f — некая нелинейная функция активации.

Вторая формула, которой описывается сеть, — это значение на выходе:

$$y_t = g(W_{hy}h_t),$$

где h_t — обновлённое состояние скрытого слоя, g — функция активации.

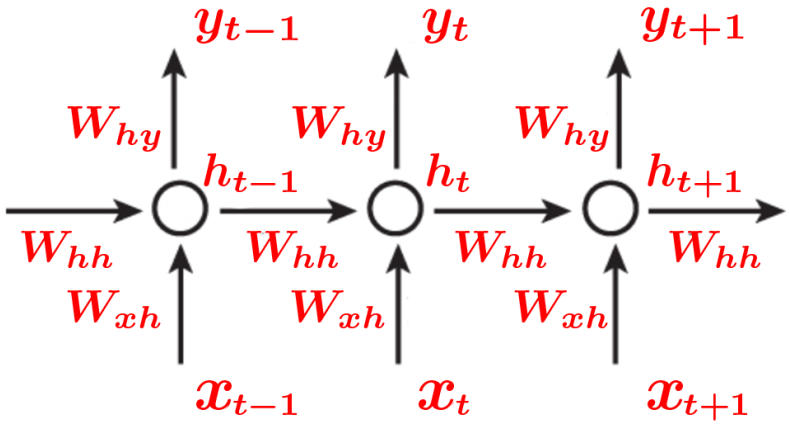

Рис. 7.2: Структура рекуррентной нейронной сети, развёрнутой во времени

Описанное выше представление является наглядным, но не очень удобным. Нейронную сеть можно изобразить в более привычном виде, если развернуть её во времени (рис. 7.2). Это обычная нейронная сеть прямого распространения, длина которой зависит от числа токенов, которые подаются на вход. При этом параметры, матрицы W_{hh}, W_{xh}, W_{hy}, являются общими и не меняются от шага к шагу. Изменяются только скрытые состояния h_t, входы x_t, выходы y_t. Этот вид сети нужен для того, чтобы её обучать.

7.2.4. Обучение

Обучать описанную выше сеть можно с помощью обратного распространения ошибки. Этот метод немного усложняется, потому что в сети между разными итерациями присутствуют общие параметры, и это нужно учитывать при дифференцировании. Также в обучении возникают особенности, связанные с тем, что градиенты распространяются очень далеко. Если нейронная сеть глубокая, в тексте много токенов, может возникнуть

затухание или, наоборот, взрыв градиентов. Эти проблемы здесь обсуждаться не будут, предполагается, что они решены в готовых пакетах для обучения нейронных сетей.

Существует много примеров использования описанной нейронной сети, далее будут рассмотрены два из них.

7.2.5. Пример: генерация текста

Порождение текстов — это, наверное, самый популярный способ использования рекуррентных нейронных сетей. На вход сети подаётся слово, а выход — это вектор вероятностей, размер которого равен размеру используемого словаря (количество слов или символов), то есть распределение на словах (или токенах). Если выбрать токен с наибольшей вероятностью, то таким образом сгенерируется новый токен. После этого можно его подать на вход, сгенерировать ещё один и т. д. Таким образом получается генерация текста.

Если обучить такую нейросеть на некотором корпусе, то можно будет генерировать похожие тексты.

Простой пример. Пусть нейронная сеть обучена на всех текстах из Википедии, причём обучение происходило на разметке этих текстов. Если сгенерировать новый текст для Википедии, то получится что-то похожее на текст, показанный на рисунке 7.3.

```
Naturalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immineners]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slop of Syrian influence that
was a famous German movement based on a more popular servicious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]]
```

Рис. 7.3: Текст, сгенериванный нейронной сетью, обученной на Википедии

Следует обратить внимание, что в этом тексте правильно сгенерирована разметка (ссылки с квадратными скобками, ссылки на веб-страницы и т.д.), то есть нейросеть хорошо улавливает структуру текста и элементы, которые в нём встречаются.

7.2.6. Пример: обучение с учителем

Ещё один пример использования рекуррентных нейронных сетей — это генерация признаков для обучения с учителем. Существует несколько подходов к тому, как извлекать из таких сетей признаки.

Первый подход заключается в том, чтобы пропустить весь текст через нейронную сеть, и в качестве признаков использовать получившийся вектор скрытого слоя. Поскольку скрытое состояние хранит в себе информацию обо всех прошедших через него словах в прошлом (если матрица переходов грамотно обучена), то полученное представление также будет содержать информацию обо всём тексте.

Можно поступить иначе: извлекать скрытый вектор после каждой итерации (после подачи каждого токена на вход сети), а после — каким-то образом их все агрегировать (например, усреднить).