Типы задач компьютерной лингвистики.

Алексей Андреевич Сорокин

МГУ им. М. В. Ломоносова
весенний семестр 2022–2023 учебного года
Межфакультетский курс “Введение в компьютерную лингвистику” 22 марта, занятие 5

Нейронные архитектуры и задачи

- Вход нейронной сети: последовательность векторов (иногда две последовательности).
- Возможные ответы:
  - Вектор (вероятностное распределение на множестве классов).
  - Последовательность векторов (распределений) той же длины.
  - Последовательность векторов (распределений) произвольной длины.

Нейронные архитектуры и задачи

- Вход нейронной сети: последовательность векторов (иногда две последовательности).
- Возможные ответы:
  - Вектор (вероятностное распределение на множестве классов).
  - Последовательность векторов (распределений) той же длины.
  - Последовательность векторов (распределений) произвольной длины.
- Промежуточные операции:
  - Преобразования векторов (полносвязные слои).
  - Локальные преобразования последовательности (свёрточные слои).
  - Глобальные преобразования последовательности (рекуррентные сети).

Задачи разметки последовательности

- Разметка последовательности — класс задач, где для каждого элемента текста нужно предсказать метку из некоторого конечного множества.
- Нейросеть предсказывает для каждого элемента распределение вероятностное распределение по классам.
- Пример – морфологический анализ:

Его	DET
решение	NOUN, gender=Neut, case=Nom, number=Plur
задачи	NOUN, gender=Fem, case=Gen, number=Sing
было	AUX, tense=Past, aspect=Imp, number=Sing, gender=Neut
неправильным	ADJ, number=Sing, gender=Neut, case=Ins

Задачи разметки последовательности

- К задачам разметки последовательности сводятся задача выделения групп.
- Пример – распознавание именованных сущностей:

Андрей	B-PER
Николаевич	I-PER
Колмогоров	I-PER
работал	O
в	O
Москве	B-LOC
в	O
Московском	B-ORG
Государственном	I-ORG
Университете	I-ORG
.	O

Задачи разметки последовательности

- Любая задача выделения групп сводится к разметке последовательности с помощью BIO-разметки.
- Аналогичные задачи:
  - Извлечение терминов.
  - Извлечение аспекта.

Задачи разметки последовательности

- Любая задача выделения групп сводится к разметке последовательности с помощью BIO-разметки.
- Аналогичные задачи:
  - Извлечение терминов.
  - Извлечение аспекта.
- Также существует BMES-разметка.
- Её используют для разбиения на морфемы:

у	ч	и	т	е	л	ь	н	и	ц	а
B-ROOT	E-ROOT	S-SUFF	B-SUFF	M-SUFF	M-SUFF	E-SUFF	B-SUFF	M-SUFF	E-SUFF	S-END

Задачи разметки последовательности: архитектура

- Стандартная архитектура для разметки последовательности:

$$
\begin{aligned}
X &= [x_1, \ldots, x_L] (\text{набор векторов}), \\
x_i &- \text{эмбеддинги слов (конкатенация эмбеддингов)}, \\
[h_1, \ldots, h_L] &= \mathrm{Encoder}([x_1, \ldots, x_L]), \\
p_i &= \mathrm{softmax}(W_{K \times d} h_i + b_K), \\
K &- \text{число классов.}
\end{aligned}
$$

- Энкодер – рекуррентная или свёрточная сеть.

Задачи разметки последовательности: архитектура

- Энкодер – рекуррентная или свёрточная сеть.
- Разновидность сети зависит от задачи:
  - “Локальные” задачи – свёрточные сети.
  - Задачи с нелокальными зависимостями – рекурретные сети.

Задачи разметки последовательности: архитектура

- Энкодер – рекуррентная или свёрточная сеть.
- Разновидность сети зависит от задачи:
  - “Локальные” задачи – свёрточные сети.
  - Задачи с нелокальными зависимостями – рекурретные сети.
- Локальные задачи:
  - Разбиение на морфемы.
- Нелокальные задачи:
  - Распознавание именованных сущностей.

Задачи разметки последовательности: архитектура

- Вход задачи – один эмбеддинг или конкатенация нескольких эмбеддингов:
  - Обучаемый слой эмбеддингов.
  - Предобученные эмбеддинги (для несловарных слов).

Задачи разметки последовательности: архитектура

- Вход задачи – один эмбеддинг или конкатенация нескольких эмбеддингов:
  - Обучаемый слой эмбеддингов.
  - Предобученные эмбеддинги (для несловарных слов).
  - Посимвольные эмбеддинги:
    - Вычисляются отдельной подсетью, идущей по символам слова (рекуррентной или свёрточной).
    - Нужны при морфологическом анализе.

Задачи разметки последовательности: архитектура

- Вход задачи – один эмбеддинг или конкатенация нескольких эмбеддингов:
  - Обучаемый слой эмбеддингов.
  - Предобученные эмбеддинги (для несловарных слов).
  - Посимвольные эмбеддинги:
    - Вычисляются отдельной подсетью, идущей по символам слова (рекуррентной или свёрточной).
    - Нужны при морфологическом анализе.
  - Вектор признаков:
    - Слово с большой буквы (только из больших) – нужно в распознавании именованных сущностей.

Порождение последовательности

- Порождение последовательности – задачи, где длина ответа заранее неизвестна.
- Базовая задача – языковое моделирование (генерация текста).

Порождение последовательности

- Порождение последовательности – задачи, где длина ответа заранее неизвестна.
- Базовая задача – языковое моделирование (генерация текста).
- Основные проблемы:
  - Ответ на k-ом шаге является частью входа на $(k+1)$-ом шаге.
  - Длина заранее неизвестна – нужно останавливаться после порождения специального токена $\langle \mathrm{END} \rangle$.
  - Жадные алгоритмы порождения могут генерировать неоптимальный ответ.

Языковые модели: введение

- Базовая задача языкового моделирования – предсказание следующего слова по предыдущим:

You know nothing, John Snow.

$$
\begin{aligned}
\langle \text{BEGIN} \rangle &\mapsto \text{You} \\
\langle \text{BEGIN} \rangle \text{ You} &\mapsto \text{know} \\
\langle \text{BEGIN} \rangle \text{ You know} &\mapsto \text{nothing} \\
\langle \text{BEGIN} \rangle \ldots \text{nothing} &\mapsto , \\
\ldots \quad \ldots &\mapsto \ldots \\
\langle \text{BEGIN} \rangle \ldots \text{Snow} &\mapsto . \\
\langle \text{BEGIN} \rangle \ldots . &\mapsto \langle \text{END} \rangle
\end{aligned}
$$

Языковые модели: введение

- Базовая задача языкового моделирования – предсказание следующего слова по предыдущим:

You know nothing, John Snow.

$$
\begin{aligned}
\langle \text{BEGIN} \rangle &\mapsto \text{You} \\
\langle \text{BEGIN} \rangle \text{ You} &\mapsto \text{know} \\
\langle \text{BEGIN} \rangle \text{ You know} &\mapsto \text{nothing} \\
\langle \text{BEGIN} \rangle \ldots \text{nothing} &\mapsto , \\
\ldots \quad \ldots &\mapsto \ldots \\
\langle \text{BEGIN} \rangle \ldots \text{Snow} &\mapsto . \\
\langle \text{BEGIN} \rangle \ldots . &\mapsto \langle \text{END} \rangle
\end{aligned}
$$

- Раньше решалась энграммными моделями.
- В них учитывается только $n - 1$ предыдущее слово.

Базовая энграммная модель

- Предположение энграммной модели: каждое слово зависит только от $n-1$ предыдущего.

$$p(w_N | w_1 \dots w_{N-1}) = p(w_N | w_{N-n+1} \dots w_{N-1})$$

Базовая энграммная модель

- Предположение энграммной модели: каждое слово зависит только от $n-1$ предыдущего.

$$p(w_N|w_1 \dots w_{N-1}) = p(w_N|w_{N-n+1} \dots w_{N-1})$$

- Чаще всего берут $n \leqslant 3$ ($n = 1$ — униграммы, $n = 2$ — биграммы, $n = 3$ — триграммы).
- Как считать энграммные вероятности?

Базовая энграммная модель

- Предположение энграммной модели: каждое слово зависит только от $n-1$ предыдущего.

$$p(w_N|w_1 \dots w_{N-1}) = p(w_N|w_{N-n+1} \dots w_{N-1})$$

- Чаще всего берут $n \leqslant 3$ ($n = 1$ — униграммы, $n = 2$ — биграммы, $n = 3$ — триграммы).
- Как считать энграммные вероятности?
- Наивный подход: $p(w_n|\mathbf{w}_{1,n-1}) = \dfrac{c(\mathbf{w}_{1,n})}{c(\mathbf{w}_{1,n-1})}$ — доля w_n среди продолжений истории $w_1 \dots w_{n-1}$.
- Здесь и далее $\mathbf{w}_{1,n} = w_1 \dots w_n$.

Базовая энграммная модель

- Предположение энграммной модели: каждое слово зависит только от $n-1$ предыдущего.

$$p(w_N|w_1 \dots w_{N-1}) = p(w_N|w_{N-n+1} \dots w_{N-1})$$

- Чаще всего берут $n \leqslant 3$ ($n = 1$ — униграммы, $n = 2$ — биграммы, $n = 3$ — триграммы).
- Как считать энграммные вероятности?
- Наивный подход: $p(w_n|\mathbf{w}_{1,n-1}) = \dfrac{c(\mathbf{w}_{1,n})}{c(\mathbf{w}_{1,n-1})}$ — доля w_n среди продолжений истории $w_1 \dots w_{n-1}$.
- Здесь и далее $\mathbf{w}_{1,n} = w_1 \dots w_n$.
- Недостаток: нулевые вероятности.

Пример

я читал		1864			
я читал	книгу	19	$\frac{19}{1864}$	$\approx$	0.010
я читал	газету	3	$\frac{3}{1864}$	$\approx$	0.002
я читал	лекцию	11	$\frac{11}{1864}$	$\approx$	0.006
я читал	доклад	0	$\frac{0}{1864}$	$=$	0?
я читал	инструкцию	0	$\frac{0}{1864}$	$=$	0?

Аддитивное сглаживание

- Можно применить аддитивное сглаживание:

$$p(t_n|t_1 \dots t_{n-1}) = \frac{c(t_1 \dots t_{n-1} t_n) + \alpha}{c(t_1 \dots t_{n-1} \bullet) + \alpha|D|},$$

  где D — словарь (множество возможных униграмм),
  $\alpha > 0$ — сглаживающее слагаемое
- При аддитивном сглаживании считается, что каждое слово дополнительно встречается α раз.

Аддитивное сглаживание

- Можно применить аддитивное сглаживание:

$$p(t_n|t_1 \dots t_{n-1}) = \frac{c(t_1 \dots t_{n-1} t_n) + \alpha}{c(t_1 \dots t_{n-1} \bullet) + \alpha|D|},$$

  где D — словарь (множество возможных униграмм),
  $\alpha > 0$ — сглаживающее слагаемое
- При аддитивном сглаживании считается, что каждое слово дополнительно встречается α раз.
- Теперь уже нет нулевых вероятностей. Но как выбирать значение α?
- Маленькая α — риск переподгонки под обучающую выборку.
- Большая α — не учитываем наблюдаемые вероятности.

Интерполяция и откат

- Недостатки аддитивного сглаживания:
  - непонятно, как подбирать α (зависит от размера корпуса, размера словаря, порядка энграмм и т. д.)

Интерполяция и откат

- Недостатки аддитивного сглаживания:
  - непонятно, как подбирать α (зависит от размера корпуса, размера словаря, порядка энграмм и т. д.)
  - метод негибкий, не учитывает историю $t_1 \dots t_{n-1}$ (если история встречалась часто, то сглаживание должно быть более слабым).

Интерполяция и откат

- Недостатки аддитивного сглаживания:
  - непонятно, как подбирать α (зависит от размера корпуса, размера словаря, порядка энграмм и т. д.)
  - метод негибкий, не учитывает историю $t_1 \dots t_{n-1}$ (если история встречалась часто, то сглаживание должно быть более слабым).
- Основная идея: будем использовать $p(t_n|t_2 \dots t_{n-1})$ для вычисления $p(t_n|t_1 \dots t_{n-1})$, если $c(t_n|t_1 \dots t_{n-1}) = 0$.
- Если слово не встречалось после текущей истории, перейдём к более короткой.

Интерполяция и откат

- Недостатки аддитивного сглаживания:
  - непонятно, как подбирать α (зависит от размера корпуса, размера словаря, порядка энграмм и т. д.)
  - метод негибкий, не учитывает историю $t_1 \dots t_{n-1}$ (если история встречалась часто, то сглаживание должно быть более слабым).
- Основная идея: будем использовать $p(t_n|t_2 \dots t_{n-1})$ для вычисления $p(t_n|t_1 \dots t_{n-1})$, если $c(t_n|t_1 \dots t_{n-1}) = 0$.
- Если слово не встречалось после текущей истории, перейдём к более короткой.
- Общая интерполяционная формула:

$$p_I(t_n|t_1 \dots t_{n-1}) = \lambda p_C(t_n|\mathbf{t}_{1,n-1}) + (1-\lambda)p_I(t_n|\mathbf{t}_{2,n-1})$$

Интерполяция и откат

- Недостатки аддитивного сглаживания:
  - непонятно, как подбирать α (зависит от размера корпуса, размера словаря, порядка энграмм и т. д.)
  - метод негибкий, не учитывает историю $t_1 \dots t_{n-1}$ (если история встречалась часто, то сглаживание должно быть более слабым).
- Основная идея: будем использовать $p(t_n|t_2 \dots t_{n-1})$ для вычисления $p(t_n|t_1 \dots t_{n-1})$, если $c(t_n|t_1 \dots t_{n-1}) = 0$.
- Если слово не встречалось после текущей истории, перейдём к более короткой.
- Общая интерполяционная формула:

$$p_I(t_n|t_1 \dots t_{n-1}) = \lambda p_C(t_n|\mathbf{t}_{1,n-1}) + (1-\lambda)p_I(t_n|\mathbf{t}_{2,n-1})$$

$p_C(t_n|t_1 \dots t_{n-1}) = \dfrac{c(t_1 \dots t_{n-1}t_n)}{c(t_1 \dots t_{n-1}\bullet)}$ — "корпусная" вероятность, λ — коэффициент, вообще говоря, зависящий от $t_1 \dots t_{n-1}$.

Пример

$w_1\,w_2$	w_3	$c(w_1\,w_2\,w_3)$	$p(w_3 \mid w_1\,w_2)$	w_2	w_3	$c(w_2\,w_3)$	$p(w_3 \mid w_2)$
я читал		1832		читал		18149	
я читал	газету	3	0.0016	читал	газету	149	0.0082
я читал	книгу	19	0.0103	читал	книгу	138	0.0076
я читал	лекцию	11	0.0060	читал	лекцию	81	0.0045
я читал	доклад	0	0	читал	доклад	22	0.0012

Пример

$w_1\,w_2$	w_3	$c(w_1\,w_2\,w_3)$	$p(w_3 \mid w_1\,w_2)$	w_2	w_3	$c(w_2\,w_3)$	$p(w_3 \mid w_2)$
я читал		1832		читал		18149	
я читал	газету	3	0.0016	читал	газету	149	0.0082
я читал	книгу	19	0.0103	читал	книгу	138	0.0076
я читал	лекцию	11	0.0060	читал	лекцию	81	0.0045
я читал	доклад	0	0	читал	доклад	22	0.0012

При $\lambda = 0.5$ получаем

$$p(\text{газету}|\text{я читал}) = 0.5 * 0.0016 + 0.5 * 0.0082 = 0.0049$$
$$p(\text{доклад}|\text{я читал}) = 0.5 * 0.0000 + 0.5 * 0.0012 = 0.0006$$

Интерполяция и откат

- Обозначим $\mathbf{t}_{i,j} = t_i \ldots t_j$.
- Общая интерполяционная формула:

$$p_I(t_n|\mathbf{t}_{1,n-1}) = \lambda p_C(t_n|\mathbf{t}_{1,n-1}) + (1-\lambda)p_I(t_n|\mathbf{t}_{2,n-1})$$

Интерполяция и откат

- Обозначим $\mathbf{t}_{i,j} = t_i \dots t_j$.
- Общая интерполяционная формула:

$$p_I(t_n|\mathbf{t}_{1,n-1}) = \lambda p_C(t_n|\mathbf{t}_{1,n-1}) + (1-\lambda)p_I(t_n|\mathbf{t}_{2,n-1})$$

- Формула отката (backoff):

$$p_I(t_n|t_1 \dots t_{n-1}) = \begin{cases} \alpha(\mathbf{t}_{1,n-1})p_C(t_n|\mathbf{t}_{1,n-1}), & c(\mathbf{t}_{1,n-1}t_n) > 0, \\ \beta(\mathbf{t}_{1,n-1})p_I(t_n|\mathbf{t}_{2,n-1}), & c(\mathbf{t}_{1,n-1}t_n) = 0 \end{cases}$$

Интерполяция и откат

- Обозначим $\mathbf{t}_{i,j} = t_i \dots t_j$.
- Общая интерполяционная формула:

$$p_I(t_n|\mathbf{t}_{1,n-1}) = \lambda p_C(t_n|\mathbf{t}_{1,n-1}) + (1-\lambda)p_I(t_n|\mathbf{t}_{2,n-1})$$

- Формула отката (backoff):

$$p_I(t_n|t_1 \dots t_{n-1}) = \begin{cases} \alpha(\mathbf{t}_{1,n-1})p_C(t_n|\mathbf{t}_{1,n-1}), & c(\mathbf{t}_{1,n-1}t_n) > 0, \\ \beta(\mathbf{t}_{1,n-1})p_I(t_n|\mathbf{t}_{2,n-1}), & c(\mathbf{t}_{1,n-1}t_n) = 0 \end{cases}$$

- Чем больше λ (α в формуле отката), тем больше мы доверяем истории $\mathbf{t}_{1,n-1}$.

Интерполяция и откат

- Обозначим $\mathbf{t}_{i,j} = t_i \dots t_j$.
- Общая интерполяционная формула:

$$p_I(t_n|\mathbf{t}_{1,n-1}) = \lambda p_C(t_n|\mathbf{t}_{1,n-1}) + (1-\lambda)p_I(t_n|\mathbf{t}_{2,n-1})$$

- Формула отката (backoff):

$$p_I(t_n|t_1 \dots t_{n-1}) = \begin{cases} \alpha(\mathbf{t}_{1,n-1})p_C(t_n|\mathbf{t}_{1,n-1}), & c(\mathbf{t}_{1,n-1}t_n) > 0, \\ \beta(\mathbf{t}_{1,n-1})p_I(t_n|\mathbf{t}_{2,n-1}), & c(\mathbf{t}_{1,n-1}t_n) = 0 \end{cases}$$

- Чем больше λ (α в формуле отката), тем больше мы доверяем истории $\mathbf{t}_{1,n-1}$.
- Много случайных продолжений у $\mathbf{t}_{1,n-1}$ — λ мало.
- Продолжений мало и они частотные — $\lambda \approx 1$

Интерполяция и откат

- Обозначим $\mathbf{t}_{i,j} = t_i \dots t_j$.
- Общая интерполяционная формула:

$$p_I(t_n|\mathbf{t}_{1,n-1}) = \lambda p_C(t_n|\mathbf{t}_{1,n-1}) + (1-\lambda)p_I(t_n|\mathbf{t}_{2,n-1})$$

- Формула отката (backoff):

$$p_I(t_n|t_1 \dots t_{n-1}) = \begin{cases} \alpha(\mathbf{t}_{1,n-1})p_C(t_n|\mathbf{t}_{1,n-1}), & c(\mathbf{t}_{1,n-1}t_n) > 0, \\ \beta(\mathbf{t}_{1,n-1})p_I(t_n|\mathbf{t}_{2,n-1}), & c(\mathbf{t}_{1,n-1}t_n) = 0 \end{cases}$$

- Чем больше λ (α в формуле отката), тем больше мы доверяем истории $\mathbf{t}_{1,n-1}$.
- Много случайных продолжений у $\mathbf{t}_{1,n-1}$ — λ мало.
- Продолжений мало и они частотные — $\lambda \approx 1$
- β подбирают, чтобы сумма вероятностей получилась 1.

Нейронные языковые модели: введение

- Нейронные модели перевода генерируют слова на основе уже переведённых слов и исходного предложения.

Нейронные языковые модели: введение

- Нейронные модели перевода генерируют слова на основе уже переведённых слов и исходного предложения.
- Без исходного предложения — языковая модель (задача предсказания следующего слова).
- данная задача требует знаний о всех уровнях языка:

Я ел вкусную ?

ADJ/NOUN	–	синтаксис,
case=Acc	–	синтаксис,
... у	–	морфология,
съедобный объект	–	семантика

Нейронные языковые модели: введение

- Нейронные модели перевода генерируют слова на основе уже переведённых слов и исходного предложения.
- Без исходного предложения — языковая модель (задача предсказания следующего слова).
- данная задача требует знаний о всех уровнях языка:

Я ел вкусную ?

ADJ/NOUN	–	синтаксис,
case=Acc	–	синтаксис,
... у	–	морфология,
съедобный объект	–	семантика

- Раньше решалась через энграммные модели (масса недостатков).

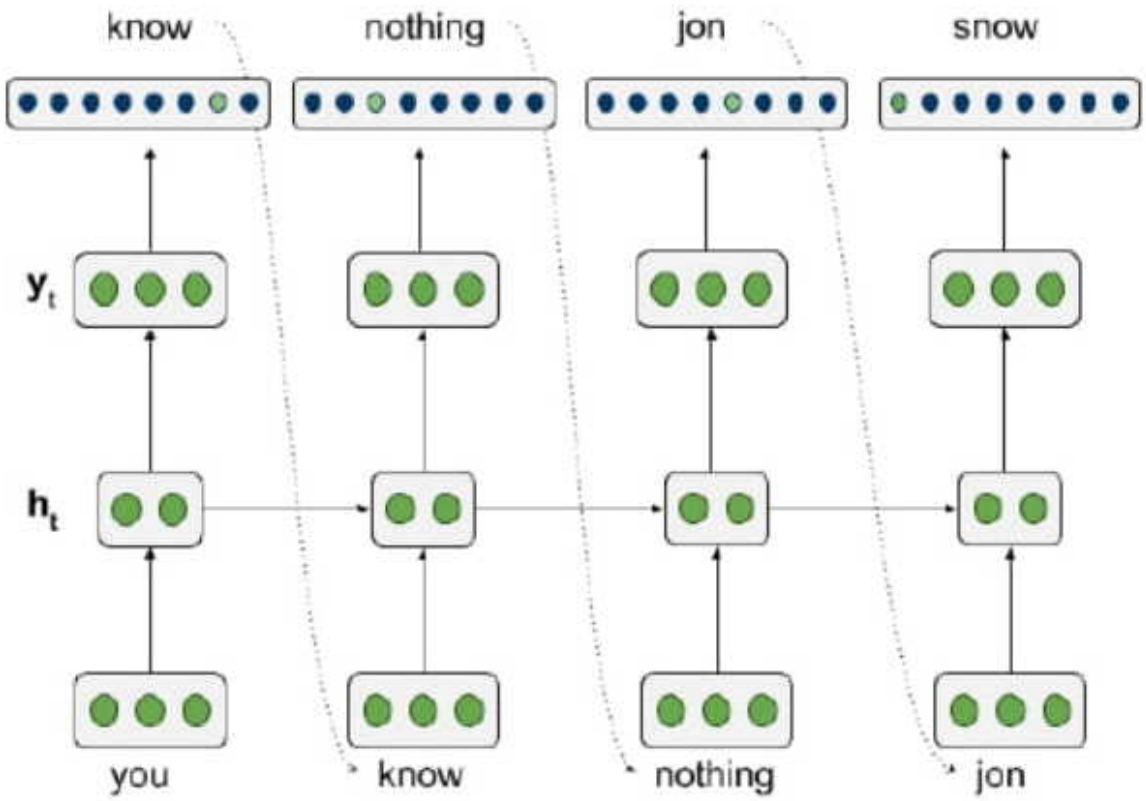
Недостатки энграммных моделей

- Не учитывают структуру языка.
- Не учитывают схожесть слов.

Недостатки энграммных моделей

- Не учитывают структуру языка.
- Не учитывают схожесть слов.
- Не учитывают дистантный контекст.

Нейронные языковые модели



Нейронные языковые модели

- Наиболее частая схема – рекуррентная сеть (часто несколько слоёв):

$$
\begin{aligned}
[h_i, c_i] &= LSTM(c_{i-1}, x_i) \\
z_i &= Wh_i + b,\ W \in \mathbb{R}^{D \times d} \\
p_i &= \mathrm{softmax}(z_i) \\
D &- \text{размер словаря (10000-100000)}, \\
d &- \text{размер состояния (100-1000)}
\end{aligned}
$$

Нейронные языковые модели

- Наиболее частая схема – рекуррентная сеть (часто несколько слоёв):

$$\begin{aligned} [h_i, c_i] &= LSTM(c_{i-1}, x_i) \\ z_i &= Wh_i + b,\ W \in \mathbb{R}^{D \times d} \\ p_i &= \mathrm{softmax}(z_i) \\ D &- \text{размер словаря (10000-100000)}, \\ d &- \text{размер состояния (100-1000)} \end{aligned}$$

- По сути, языковая модель предлагает представление следующего слова и сравнивает его со словарными.

$$z_{ij} = \langle W_{j\cdot}, h_i \rangle$$

Нейронные языковые модели

- Языковая модель предлагает представление следующего слова и сравнивает его со словарными.

$$z_{ij} = \langle W_{j\cdot}, h_i \rangle$$

- p_{ij} монотонно зависит от z_{ij}.
- Скалярное произведение — мера похожести h_i на каждую из строк матрицы.

Нейронные языковые модели

- Языковая модель предлагает представление следующего слова и сравнивает его со словарными.

$$z_{ij} = \langle W_{j\cdot}, h_i \rangle$$

- p_{ij} монотонно зависит от z_{ij}.
- Скалярное произведение — мера похожести h_i на каждую из строк матрицы.
- То есть h_i – это “идеальное” векторное представление следующего слова.
- Вероятность словарного слова в этой позиции пропорциональна близости к h_i.

Преимущества нейронных моделей

- Схожие слова – схожие входные вектора – схожие предсказания следующих слов.
- Дистантный контекст запоминается в состоянии LSTM.

Преимущества нейронных моделей

- Схожие слова – схожие входные вектора – схожие предсказания следующих слов.
- Дистантный контекст запоминается в состоянии LSTM.
- Лингвистическая информация неявно запоминается в представлениях слов.
- Также её можно явно конкатенировать с входными векторами.

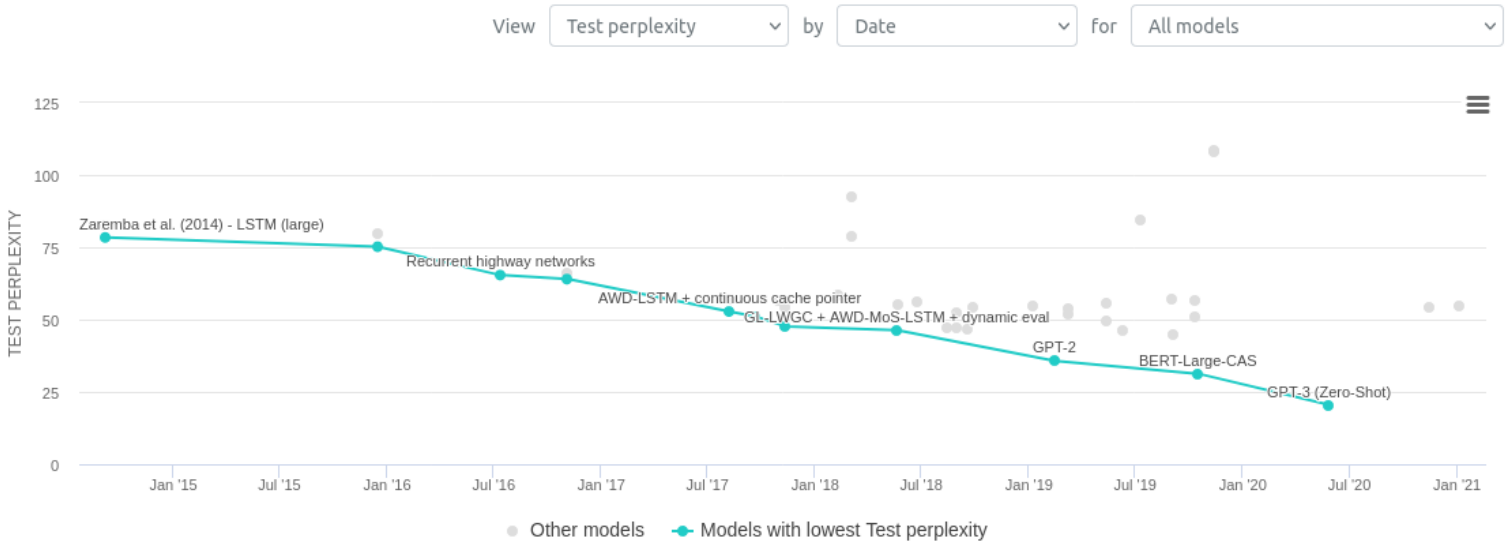
Преимущества нейронных моделей

- Перплексия нейронных моделей существенно ниже, то есть качество лучше (данные на Penn Treebank):

Model	Parameters	Validation	Test
Mikolov & Zweig (2012) - KN-5	2M[‡]	–	141.2
Mikolov & Zweig (2012) - KN5 + cache	2M[‡]	–	125.7
Mikolov & Zweig (2012) - RNN	6M[‡]	–	124.7
Mikolov & Zweig (2012) - RNN-LDA	7M[‡]	–	113.7
Mikolov & Zweig (2012) - RNN-LDA + KN-5 + cache	9M[‡]	–	92.0
Zaremba et al. (2014) - LSTM (medium)	20M	86.2	82.7
Zaremba et al. (2014) - LSTM (large)	66M	82.2	78.4
Gal & Ghahramani (2016) - Variational LSTM (medium)	20M	81.9 ± 0.2	79.7 ± 0.1
Gal & Ghahramani (2016) - Variational LSTM (medium, MC)	20M	–	78.6 ± 0.1
Gal & Ghahramani (2016) - Variational LSTM (large)	66M	77.9 ± 0.3	75.2 ± 0.2
Gal & Ghahramani (2016) - Variational LSTM (large, MC)	66M	–	73.4 ± 0.0
Kim et al. (2016) - CharCNN	19M	–	78.9
Merity et al. (2016) - Pointer Sentinel-LSTM	21M	72.4	70.9

Преимущества нейронных моделей

- Перплексия нейронных моделей существенно ниже, то есть качество лучше (данные на Penn Treebank):



Нейронный машинный перевод

- Нейронный машинный перевод — задача условного порождения текста.
- Обычная вероятностная модель порождает текст на основе предыдущих слов.

$$w_i \sim p(w|h_{i-1})$$

h_{i-1} – состояние языковой модели после $i-1$ слова

Нейронный машинный перевод

- Нейронный машинный перевод — задача условного порождения текста.
- Обычная вероятностная модель порождает текст на основе предыдущих слов.

$$\begin{aligned} w_i &\sim p(w|h_{i-1}) \\ h_{i-1} &- \text{состояние языковой модели после } i-1 \text{ слова} \end{aligned}$$

- Условная вероятностная модель:

$$\begin{aligned} w_i &\sim p(w|h_{i-1}, c) \\ c &- \text{глобальный контекст} \end{aligned}$$

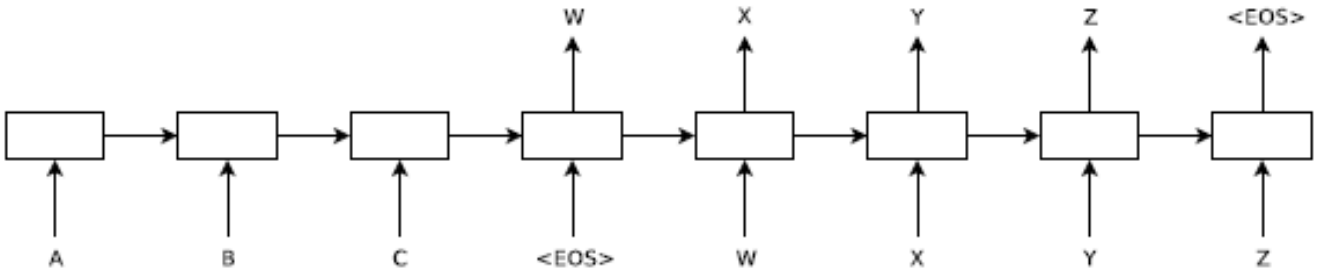
- Основная проблема: как вычислять c.

Нейронный машинный перевод

- Базовая модель: кодировщик-декодировщик (encoder-decoder):

$$\begin{aligned} c &= LSTM(x_1, \ldots, x_m), \\ x_1 \ldots x_m &- \text{исходное предложение} \end{aligned}$$

- Вектор c запоминает всю информацию об исходном предложении в одном векторе.



Нейронный машинный перевод

- Обычно на каждый шаг декодера явно подаётся предыдущее слово:

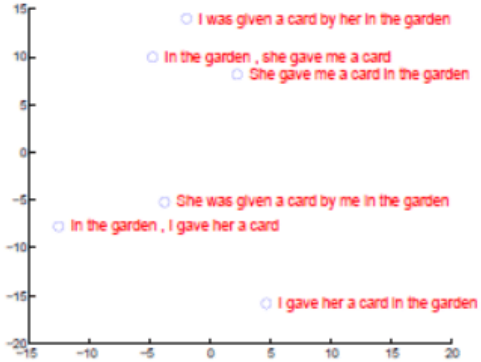
$$p(y_t | [\![y_1, \ldots, y_{t-1}]\!], c) = g(y_{t-1}, s_t, c)$$

- Как и энкодер, так и декодер включают в себя несколько слоёв.
- Входом следующего служит выход предыдущего.

Вектора предложений

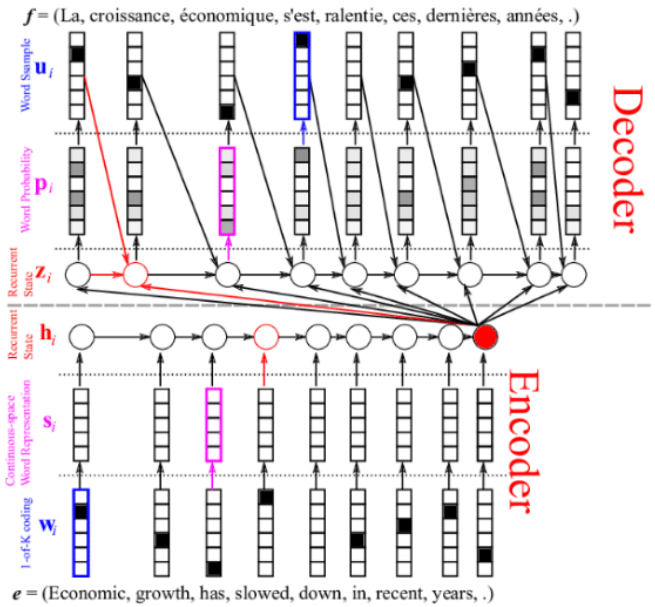
Секвенциальные модели

Вектора предложений



Нейронный машинный перевод

Нейронный машинный перевод



Обучение нейронных систем

- Многоклассовые нейронные классификаторы обучаются с помощью кросс-энтропии:

$$L(y, y') = -\sum y_i \log y'_i$$

- y_i, y'_i — правильное и предсказанное вероятностное распределения.

Обучение нейронных систем

- Многоклассовые нейронные классификаторы обучаются с помощью кросс-энтропии:

$$L(y, y') = -\sum y_i \log y'_i$$

- y_i, y'_i — правильное и предсказанное вероятностное распределения.
- Можно вычислять кросс-энтропию для каждого слова

$$\begin{aligned} L(\mathbf{t}, \mathbf{t}') &= -\sum_j L_0(t_j, t'_j) \\ L_0(t_j, t'_j) &= -\sum t_{ij} \log t'_{ij} \end{aligned}$$

- Здесь в каждой позиции предсказывается вероятностное распределение для текущего переводного слова.

Обучение нейронных систем

- Многоклассовые нейронные классификаторы обучаются с помощью кросс-энтропии:

$$L(y, y') = -\sum y_i \log y'_i$$

- y_i, y'_i — правильное и предсказанное вероятностное распределения.
- Можно вычислять кросс-энтропию для каждого слова

$$\begin{aligned} L(\mathbf{t}, \mathbf{t}') &= -\sum_j L_0(t_j, t'_j) \\ L_0(t_j, t'_j) &= -\sum t_{ij} \log t'_{ij} \end{aligned}$$

- Здесь в каждой позиции предсказывается вероятностное распределение для текущего переводного слова.
- Конец предложения: специальный символ END.
- Позиции после конца предложения: специальный символ PAD.

Обучение нейронных систем: некорректная позиция в предложении

- Система может предсказать несколько слов вместо одного:

This is	the shortest	way home
Это	кратчайший	путь домой
Это	самый короткий	путь домой

- После “самый короткий” правильным словом будет считаться “домой”.

Обучение нейронных систем: exposure bias

- Большинство систем машинного перевода использует предыдущее слово при порождении следующего.
- При этом используется *корректное* предыдущее слово.

Обучение нейронных систем: exposure bias

- Большинство систем машинного перевода использует предыдущее слово при порождении следующего.
- При этом используется *корректное* предыдущее слово.
- Объяснение: если брать действительно предсказанное слово, то следующее слово в исходном предложении не будет корректным:

This is	the shortest	way home
Это	самый	путь? домой

Обучение нейронных систем: exposure bias

- Большинство систем машинного перевода использует предыдущее слово при порождении следующего.
- При этом используется *корректное* предыдущее слово.
- Объяснение: если брать действительно предсказанное слово, то следующее слово в исходном предложении не будет корректным:

This is	the shortest	way home
Это	самый	путь? домой

- Это означает, что при обучении модель не видит неправильных вариантов.

Обучение нейронных систем: exposure bias

- Большинство систем машинного перевода использует предыдущее слово при порождении следующего.
- При этом используется *корректное* предыдущее слово.
- Объяснение: если брать действительно предсказанное слово, то следующее слово в исходном предложении не будет корректным:

This is	the shortest	way home
Это	самый	путь? домой

- Это означает, что при обучении модель не видит неправильных вариантов.
- Как следствие, если неправильный вариант появляется при предсказании, модель запутывается.

Обучение нейронных систем: функции штрафа и метрики

- Модели нейронного машинного перевода обучаются минимизировать кросс-энтропию.
- Эта функция штрафа не имеет отношения к реальному качеству перевода.

Обучение нейронных систем: функции штрафа и метрики

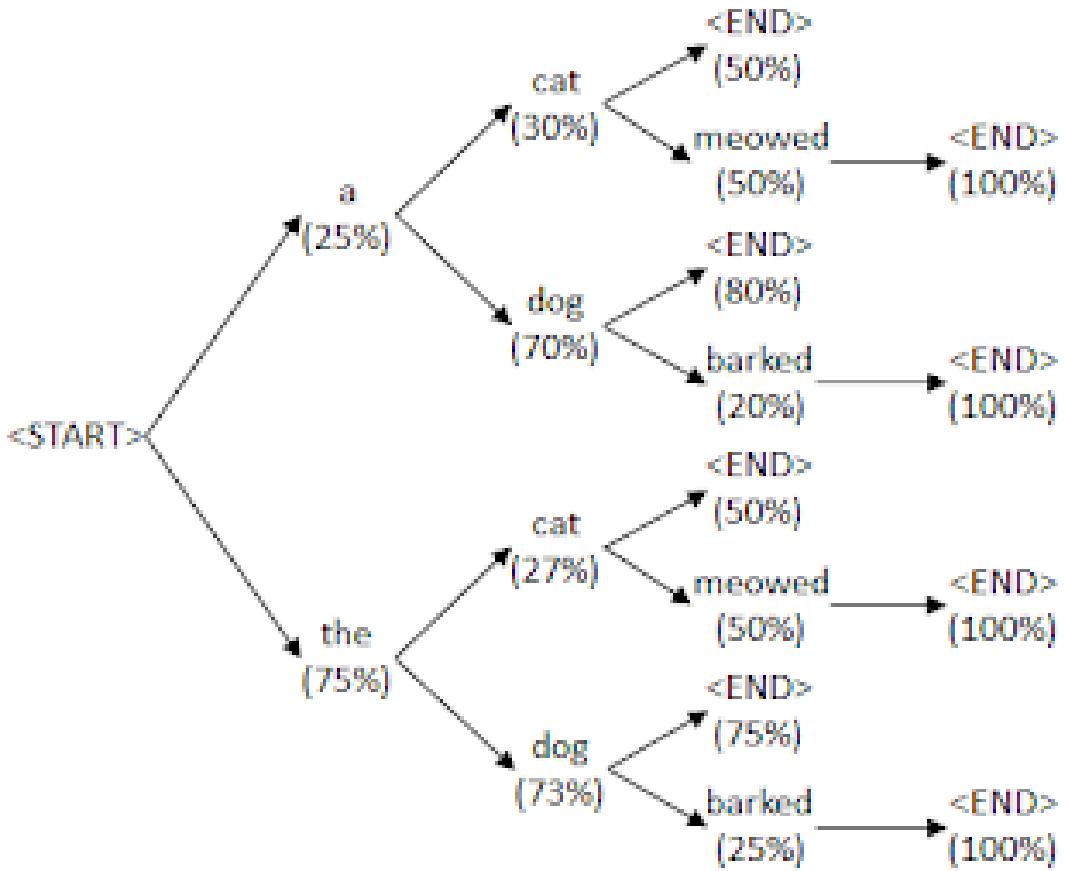
- Модели нейронного машинного перевода обучаются минимизировать кросс-энтропию.
- Эта функция штрафа не имеет отношения к реальному качеству перевода.
- Стандартная метрика для машинного перевода – BLEU.
- Основана на точности по словам и энграммам из правильного перевода (проверяет, что перевод не содержит лишних слов).
- Есть дополнительный штраф за слишком короткий ответ.

Обучение нейронных систем: функции штрафа и метрики

- Модели нейронного машинного перевода обучаются минимизировать кросс-энтропию.
- Эта функция штрафа не имеет отношения к реальному качеству перевода.
- Стандартная метрика для машинного перевода – BLEU.
- Основана на точности по словам и энграммам из правильного перевода (проверяет, что перевод не содержит лишних слов).
- Есть дополнительный штраф за слишком короткий ответ.
- BLEU неидеально коррелирует с оценкой качества человеком, но ничего лучше не придумано.
- BLEU недифференцируема, поэтому её нельзя оптимизировать градиентным спуском.

Машинный перевод: декодирование

- Стандартный способ декодирования в машинном переводе – поиск по лучу:



Машинный перевод: декодирование

- Стандартный способ декодирования в машинном переводе – поиск по лучу (beam search):
  - Поддерживается k наилучших гипотез (часто $k \in 5 \dots 10$).
  - На каждом шаге считаются все возможные продолжения существующих гипотез и пересчитываются их вероятности.
  - После этого снова выбираются k наилучших продолжений.
- Частный случай $k = 1$ – жадный поиск.

Машинный перевод: декодирование

- Стандартный способ декодирования в машинном переводе – поиск по лучу (beam search):
  - Поддерживается k наилучших гипотез (часто $k \in 5 \dots 10$).
  - На каждом шаге считаются все возможные продолжения существующих гипотез и пересчитываются их вероятности.
  - После этого снова выбираются k наилучших продолжений.
- Частный случай $k = 1$ – жадный поиск.
- Недостатки алгоритма:
  - Небольшое k – будет отсекаться много полезного.
  - Большое k – будут появляться вероятные ответы “общей тематики” (не связанные с конкретным входом).

Упрощение генерации

- Генерация последовательностей – сложная задача (и вычислительно, и алгоритмически).
- Иногда её можно упростить, сведя к классификации или разметке последовательности:

$$\text{корова} \mapsto \text{коровой}$$
$$\text{корова} \mapsto \text{а-ой}.$$

Упрощение генерации

- Генерация последовательностей – сложная задача (и вычислительно, и алгоритмически).
- Иногда её можно упростить, сведя к классификации или разметке последовательности:

$$\text{корова} \mapsto \text{коровой}$$
$$\text{корова} \mapsto \text{а-ой}.$$

- Более сложные шаблоны тоже можно упростить:

песок	$\mapsto$	песком	(1+о+2)-(1+2+ом)
volver	$\mapsto$	vuelvo	(1+o+2+er)-(1+ue+2+o)

Упрощение генерации

- Генерация последовательностей – сложная задача (и вычислительно, и алгоритмически).
- Иногда её можно упростить, сведя к классификации или разметке последовательности:

$$\text{корова} \mapsto \text{коровой}$$
$$\text{корова} \mapsto \text{а-ой}.$$

- Более сложные шаблоны тоже можно упростить:

песок	$\mapsto$	песком	(1+о+2)-(1+2+ом)
volver	$\mapsto$	vuelvo	(1+o+2+er)-(1+ue+2+o)

- Суммаризация (extractive summarization) в простейших случаях сводится к разметке последовательности (KEEP, DELETE, специальные операции).