



# Языковые модели

Машинное обучение в диалоговых системах

Лекция 7

26 марта 2019

# Суть



# План на сегодня

- Авторегрессионные модели
- Способы оценивания и перплексия
- LM на  $n$ -граммах
- LM на рекуррентных сетях
- Генерация текстов
- Shallow / deep fusion
- Transfer / multi-task learning
- Всякие трюки для ускорения обучения

# Марковские цепи

**Цепь Мэркова** —

последовательность случайных событий с конечным или счётным числом исходов, характеризующаяся тем свойством, что, говоря нестрого, при фиксированном настоящем будущее независимо от прошлого. Названа в честь А. А. Маркова (старшего).



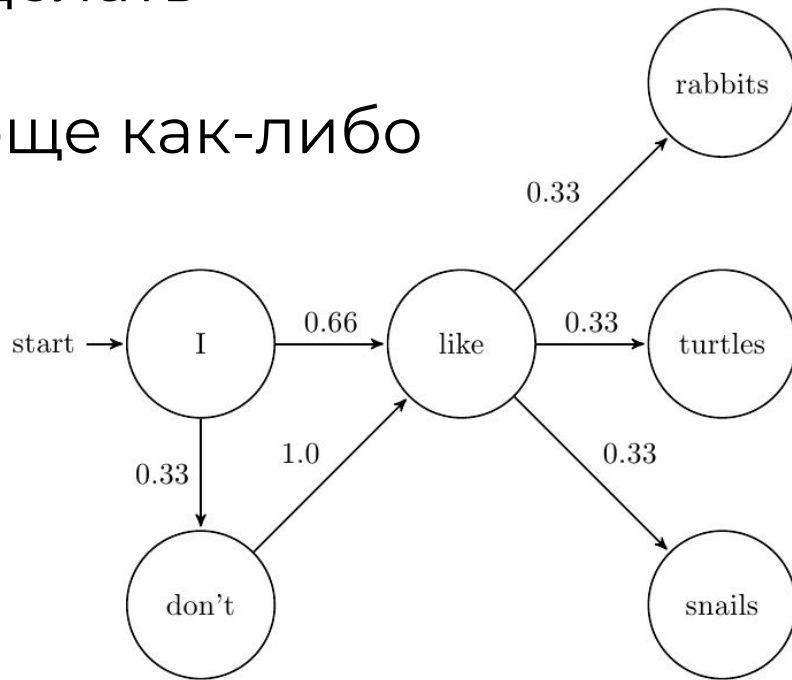
# Языковые модели

- Язык тоже можно описать марковскими цепями
- Состояние — префикс текста
- Переход — выбор следующего слова



# N-граммы

- Не обязательно все делать нейросетями
- Не обязательно вообще как-либо использовать ML



# Биграммы: обучение

```
class TextGenerator:
    '''Primitive bigram statistics based model.'''
    def __init__(self, data, lc):
        '''If given a list, processes multiple files.
        Otherwise, processes just one string.'''
        self.lc = lc
        self.bigrams = {}
        if isinstance(data, str):
            self.add(data)
        else:
            assert isinstance(data, list)
            for path in data:
                with open(path) as file:
                    for line in file:
                        self.add(line)

    def add(self, line):
        '''Trains on single line.'''
        words = re.findall(r'\p{L}+', line)
        if self.lc:
            words = list(map(str.lower, words))
        for prev_word, next_word in zip(words, words[1:]):
            self.bigrams.setdefault(prev_word, {})
            self.bigrams[prev_word].setdefault(next_word, 0)
            self.bigrams[prev_word][next_word] += 1
```

# Биграммы: генерация

```
def generate(self, length, state=None):
    assert len(self.bigrams) > 0
    sentence = []
    for _ in range(length):
        if state not in self.bigrams:
            state = choice(list(self.bigrams.keys()))
        sentence += [state]
        d = self.bigrams[state]
        words = list(d.keys())
        counts = list(d.values())
        total = sum(counts)
        probs = [x / total for x in counts]
        state = choice(words, p=probs)
    assert len(sentence) == length
    return sentence
```



# N-граммы

- Быстрое обучение: нужно просто один раз пройти по тексту
- Быстрый inference: просто сэмплирование
- Не умеют улавливать долгосрочные зависимости
- Много весят (порядка исходного текста)
- Нельзя встроить в вычислительный граф

Can you please come here ?

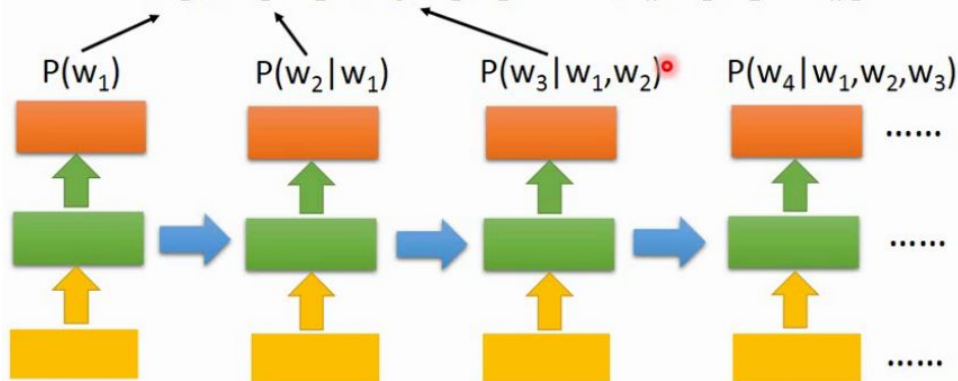


# Нейросети

- Закодируем сеткой префикс, применим к выходу softmax и будем предсказывать следующее слово
- Логично применить какую-нибудь рекуррентную сеть (LSTM, GRU)

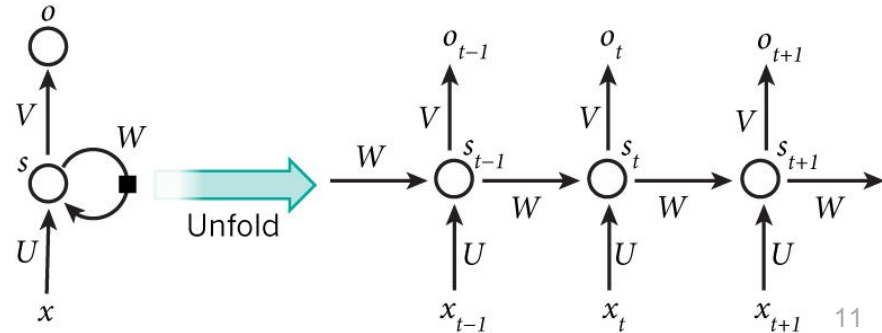
- To compute  $P(w_1, w_2, w_3, \dots, w_n)$  by RNN

$$P(w_1, w_2, w_3, \dots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \dots P(w_n|w_1, w_2 \dots w_{n-1})$$



# Нейросети: генерация

1. Поддерживаем hidden state рекуррентной сети
2. Делаем из него вероятностное распределение (linear + softmax)
3. Сэмплируем слово из этого распределения
4. Выписываем это слово, кормим **его же** эмбединг на вход ячейке RNN, обновляем hidden
5. Если не пришли в терминальное состояние (сгенерировали точку), goto 1

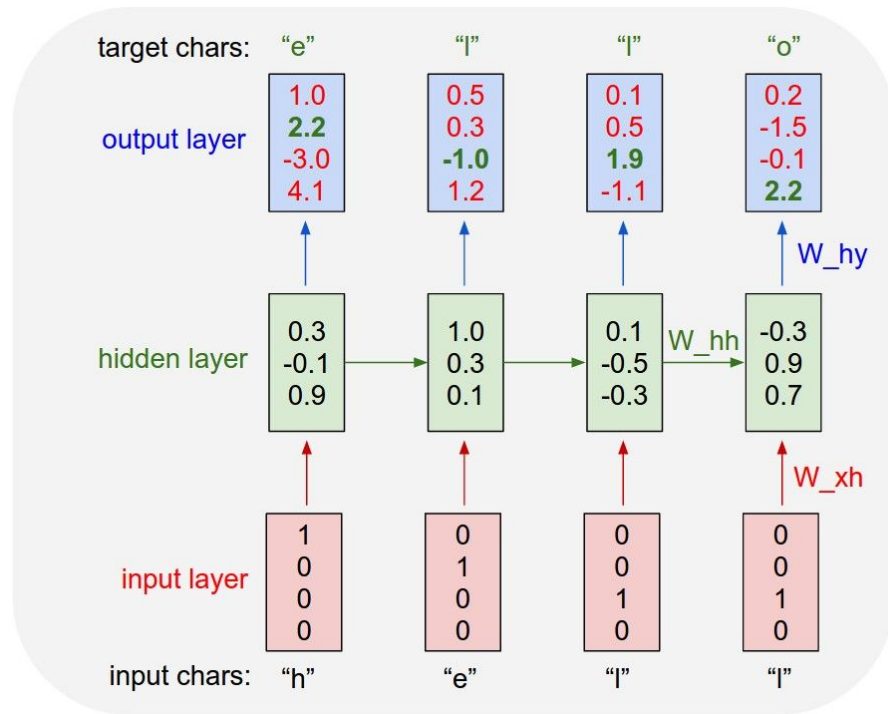


# Как их оценивать

- $P(w_n, w_{n-1}, w_{n-2}, \dots, w_1) = \prod_{i=1}^n P(w_i | w_{i-1}, w_{i-2}, \dots, w_1)$
- Учимся предсказывать  $P(w_i | w_{i-1}, w_{i-2}, \dots, w_1)$
- **Перплексия**  $PP = \hat{P}(w_1, w_2, \dots, w_m)^{-\frac{1}{m}}$
- Чем меньше — тем лучше
- Выход модели — вероятностное распределение
- По сути, мы делаем классификацию — логично использовать кроссэнтропию как функцию потерь

# Teacher forcing

- Применим лосс не только к последнему выходу, а сразу ко всем
- $\Leftrightarrow$  предсказываем собственный вход, смещенный на единицу
- Маски: после конца предложения ничего делать не надо



# Checkpoint

- Очень важно осознать всё, что было до этого слайда
- Семинар / домашка — реализовать языковую модель
- Задавайте вопросы, если вам что-либо не до конца ясно

## Ок, зачем это надо?

- Проверка орфографии
- Подсказки в поиске
- Генерация текстов
- Генеративный чат-бот
- Распознавание речи
- Машинный перевод
- Тренд 2018 года: transfer learning в NLP
- Весь мир сейчас учится делать из LM-ок что-нибудь полезное

# Генерация текстов: Шейкспир

PANDARUS:

Alas, I think he shall be come approached and the day  
When little strain would be attain'd into being never fed,  
And who is but a chain and subjects of his death,  
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,  
Breaking and strongly should be buried, when I perish  
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and  
my fair nudes begun out of the fact, to be conveyed,  
Whose noble souls I'll have the heart of the wars.

Clown:

Come, sir, I will make did behold your worship.



# Генерация текстов: ядро Linux

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_SB(in.addr);
    selector = seg / 16;
    setup_works = true;
    for (i = 0; i < blocks; i++) {
        seq = buf[i++];
        bpf = bd->bd.next + i * search;
        if (fd) {
            current = blocked;
        }
    }
    rw->name = "Getjbbregs";
    bprm_self_clearl(&iv->version);
    regs->new = blocks[(BPF_STATS << info->historidac) | PFMR_CLOBATHINC_SECONDS << 12;
    return segtable;
}
```

# Генерация текстов: arXiv

For  $\bigoplus_{n=1, \dots, m} \mathcal{L}_{m, \bullet} = 0$ , hence we can find a closed subset  $\mathcal{H}$  in  $\mathcal{H}$  and any sets  $\mathcal{F}$  on  $X$ ,  $U$  is a closed immersion of  $S$ , then  $U \rightarrow T$  is a separated algebraic space.

*Proof.* Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by  $\coprod Z \times_U U \rightarrow V$ . Consider the maps  $M$  along the set of points  $\text{Sch}_{\text{fpf}}$  and  $U \rightarrow U$  is the fibre category of  $S$  in  $U$  in Section, ?? and the fact that any  $U$  affine, see Morphisms, Lemma ?? . Hence we obtain a scheme  $S$  and any open subset  $W \subset U$  in  $\text{Sh}(G)$  such that  $\text{Spec}(R') \rightarrow S$  is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that  $f_i$  is of finite presentation over  $S$ . We claim that  $\mathcal{O}_{X, x}$  is a scheme where  $x, x', s'' \in S'$  such that  $\mathcal{O}_{X, x'} \rightarrow \mathcal{O}'_{X', x'}$  is separated. By Algebra, Lemma ?? we can define a map of complexes  $\text{GL}_{S'}(x'/S'')$  and we win.  $\square$

To prove study we see that  $\mathcal{F}|_U$  is a covering of  $\mathcal{X}'$ , and  $\bar{t}_i$  is an object of  $\mathcal{F}_{X/S}$  for  $i > 0$  and  $\mathcal{F}_p$  exists and let  $\mathcal{F}_i$  be a presheaf of  $\mathcal{O}_X$ -modules on  $\mathcal{C}$  as a  $\mathcal{F}$ -module. In particular  $\mathcal{F} = U/\mathcal{F}$  we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S, s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{\text{fpf}}^{\text{opp}}, (\text{Sch}/S)_{\text{fpf}}$$

and

$$V = \Gamma(S, \mathcal{O}) \mapsto (U, \text{Spec}(A))$$

is an open subset of  $X$ . Thus  $U$  is affine. This is a continuous map of  $X$  is the inverse, the groupoid scheme  $S$ .

*Proof.* See discussion of sheaves of sets.  $\square$

The result for prove any open covering follows from the less of Example ?? . It may replace  $S$  by  $X_{\text{spaces}, \text{étale}}$  which gives an open subspace of  $X$  and  $T$  equal to  $S_{\text{zar}}$ , see Descent, Lemma ?? . Namely, by Lemma ?? we see that  $R$  is geometrically regular over  $S$ .

**Lemma 0.1.** Assume (3) and (3) by the construction in the description.

Suppose  $X = \lim |X|$  (by the formal open covering  $X$  and a single map  $\text{Proj}_X(\mathcal{A}) = \text{Spec}(B)$  over  $U$  compatible with the complex

$$\text{Set}(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X, \mathcal{O}_X}).$$

When in this case of to show that  $\mathcal{Q} \rightarrow \mathcal{C}_{Z/X}$  is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If  $T$  is surjective we may assume that  $T$  is connected with residue fields of  $S$ . Moreover there exists a closed subspace  $Z \subset X$  of  $X$  where  $U$  in  $X'$  is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1)  $f$  is locally of finite type. Since  $S = \text{Spec}(R)$  and  $Y = \text{Spec}(R)$ .

*Proof.* This is form all sheaves of sheaves on  $X$ . But given a scheme  $U$  and a surjective étale morphism  $U \rightarrow X$ . Let  $U \cap U = \coprod_{i=1, \dots, n} U_i$  be the scheme  $X$  over  $S$  at the schemes  $X_i \rightarrow X$  and  $U = \lim_i X_i$ .  $\square$

The following lemma surjective restrocomposes of this implies that  $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{x, \dots, 0}$ .

**Lemma 0.2.** Let  $X$  be a locally Noetherian scheme over  $S$ ,  $E = \mathcal{F}_{X/S}$ . Set  $\mathcal{I} = \mathcal{I}_1 \subset \mathcal{I}_n$ . Since  $T^n \subset T^n$  are nonzero over  $i_0 \leq p$  is a subset of  $\mathcal{I}_{n,0} \circ \bar{A}_2$  works.

**Lemma 0.3.** In Situation ?? . Hence we may assume  $\mathfrak{q}' = 0$ .

*Proof.* We will use the property we see that  $\mathfrak{p}$  is the next functor (??). On the other hand, by Lemma ?? we see that

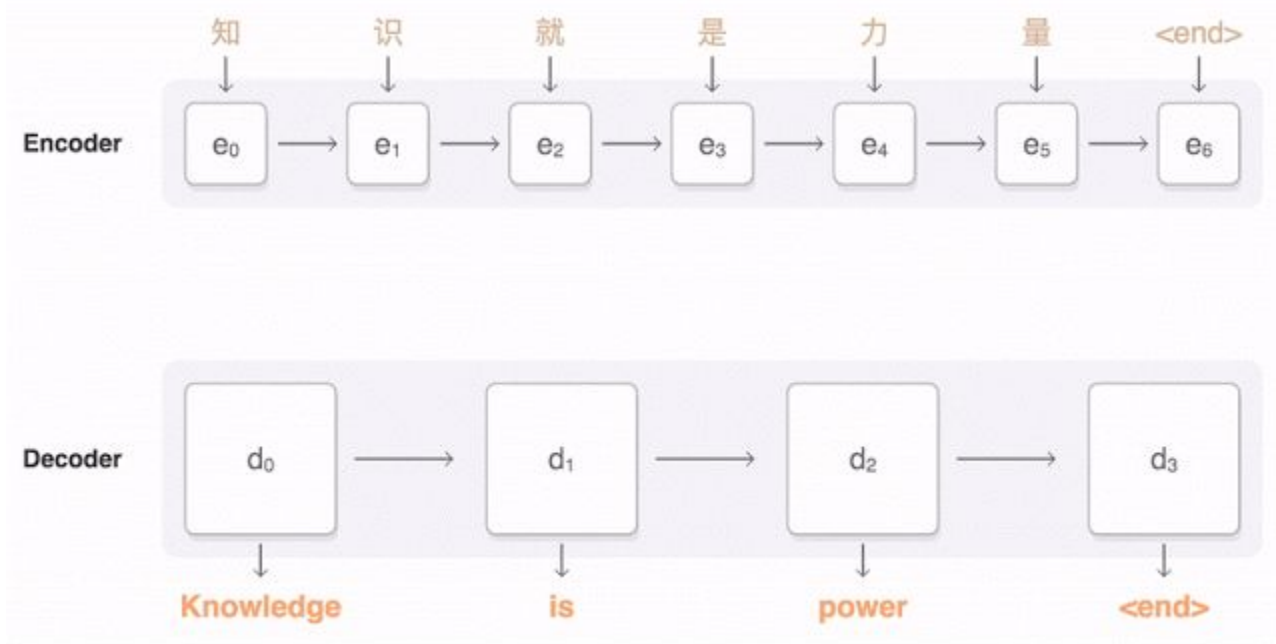
$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where  $K$  is an  $F$ -algebra where  $\delta_{n+1}$  is a scheme over  $S$ .  $\square$

# Генерация музыки



# Seq2seq

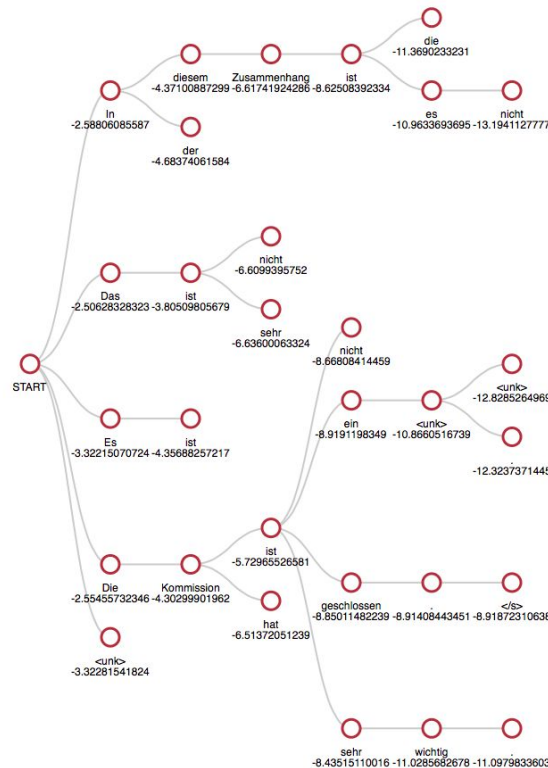


# Где нужно генерировать текст

- Распознавание речи
- Переводчик
- Image captioning
- Чат-бот
- Подсказки в поиске
- Style transfer для текстов

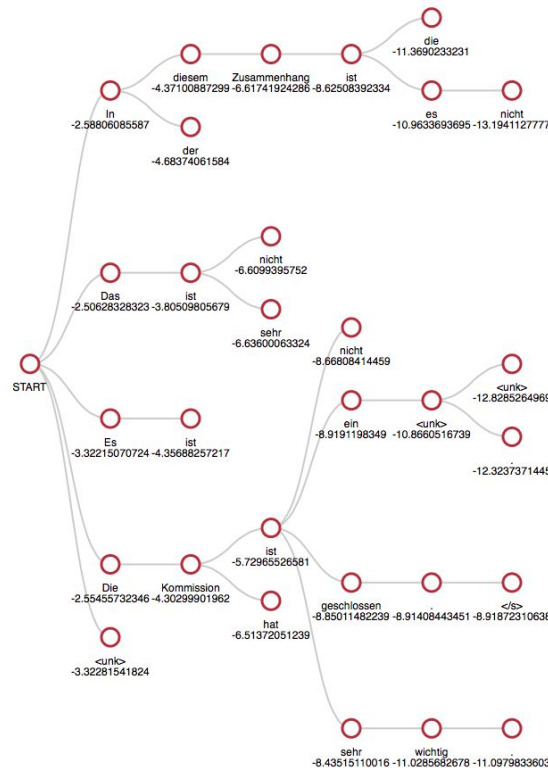
# Beam Search

- В seq2seq-задачах нам нужно сгенерировать самый вероятный ответ, а не просто из распределения
- Перебирать все возможные ответы и скорить — долго
- Каждое следующее слово брать как самое вероятное — не всегда оптимально



# Beam Search

- Будем поддерживать K самых вероятных кандидатов
- Из каждого сгенерируем N кандидатов-продолжений
- Выберем из этих KN кандидатов K самых вероятных
- После EOS ничего генерировать не надо



# Сжатие данных

- Hutter Prize — чемпионат мира по сжатию англоязычной Википедии
- Есть способ сделать из языковой модели кодировщик: арифметическое кодирование
- Впрочем, конкретно на Hutter Prize побеждают не нейросети — учитывается длина всей программы, а параметры нейронки сами весят больше википедии



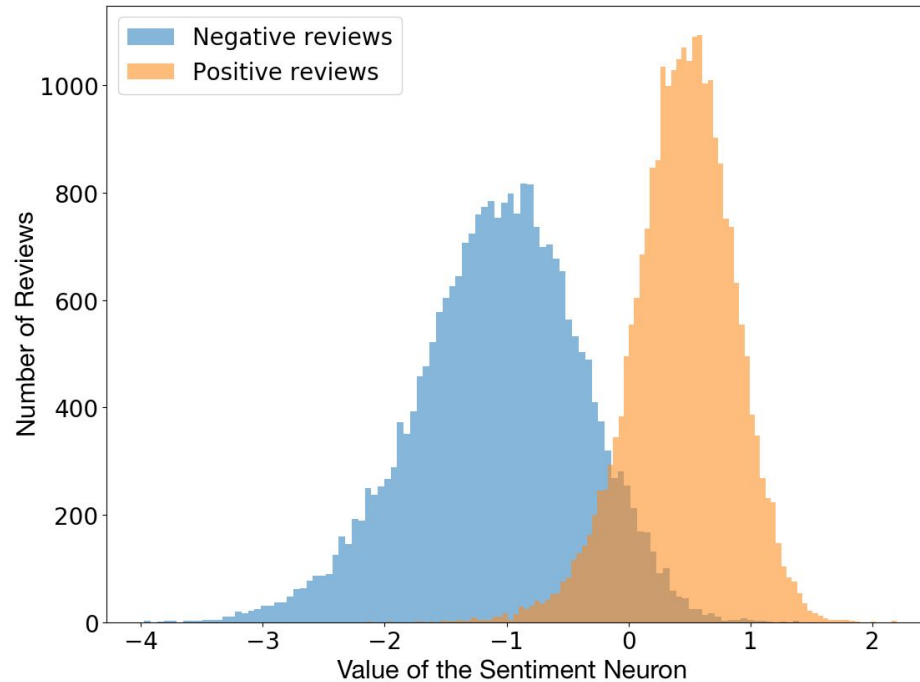
# Ансамблирование

- Моделирование естественного языка — «AI-полная» задача
- Данных для обучения **очень** много
- Обученная языковая модель хранит в себе огромную информацию о языке
- Высоковероятный по мнению LM вывод скорее всего правильный
- Её мнение можно учитывать при решении seq2seq-задач, где данных не бесконечность

# Deep и shallow fusion

- Shallow fusion: языковая модель используется только во время inference (обычное ансамблирование)
- Deep fusion: в логиты основной модели с каким-то коэффициентом прибавляются логиты LM-ки, все обучается как одна цельная сеть

# Sentiment Neuron ([OpenAI, 2017](#))



# Sentiment Neuron

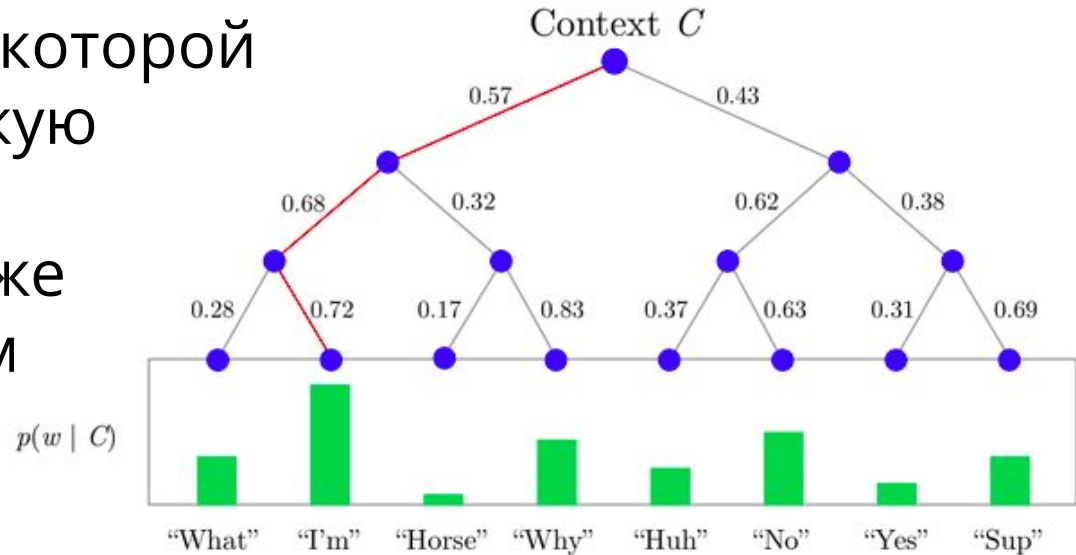
This is one of Crichton's best books. The characters of Karen Ross, Peter Ellis Munro, and Amy are beautifully developed and their interactions are exciting, complex, and fast-paced throughout this impressive novel. And about 99.8 percent of the novel got lost in the film. Seriously, the screenplay AND the directing were horrendous and clearly done by people who could not fathom what was good about the novel. I can't fault the actors because frankly, they never had a chance to make the turkey live up to Crichton's original work. I know good novels, especially those with a science fiction edge, are hard to bring to the screen in a way that lives up to the original. But this may be the absolute worst disparity in quality between novel

# Обучать state-of-the-art LM сложно

- Очень много данных
- Capacity модели почти не ограничена
- Даже если данных бесконечность, о регуляризации все равно нужно беспокоиться (в AWD-LSTM три разных дропаута)
- Очень много весов из-за необходимости делать логистическую регрессию на много слоев, их все нужно считать

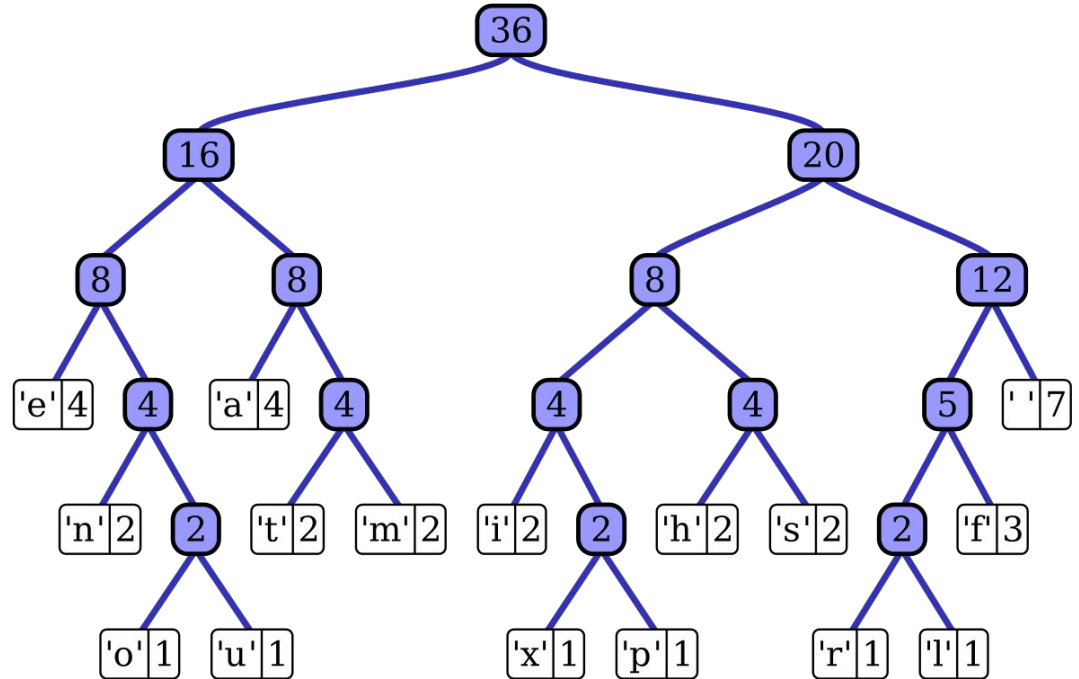
# Hierarchical softmax ([Facebook, 2017](#))

- Мысленно разделим выход softmax-а на две части
- Добавим вершину, в которой будет решаться, в какую половину пойти
- Каждую половину тоже рекурсивно разделим
- Дерево должно минимизировать ожидаемую глубину токена
- Знаете такое?



# Дерево Хаффмана

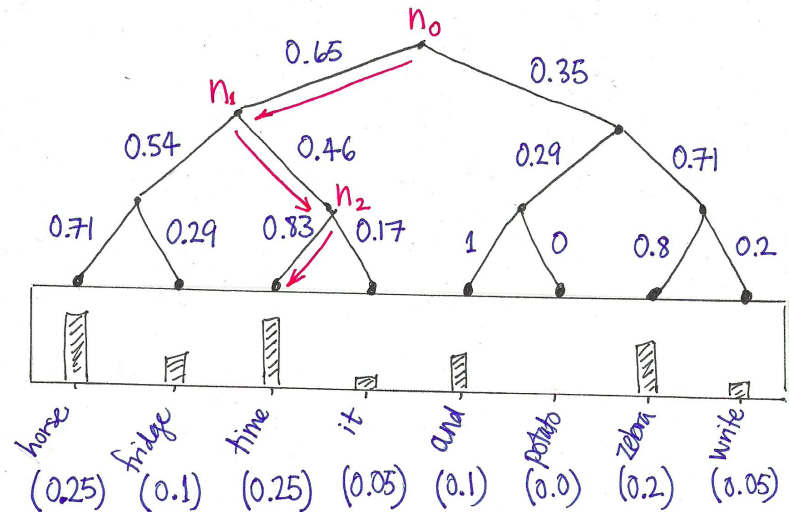
- Тоже пришло из теории информации и сжатия данных
- Минимизирует ожидаемую глубину токена



# Hierarchical softmax

- Получается аппроксимация обычного софтбокса
- Качество чуть хуже, но зато работает за  $O(\log n)$
- В обычных языковых моделях никто особо не использует, но идея прикольная

$P(\cdot | \varphi)$





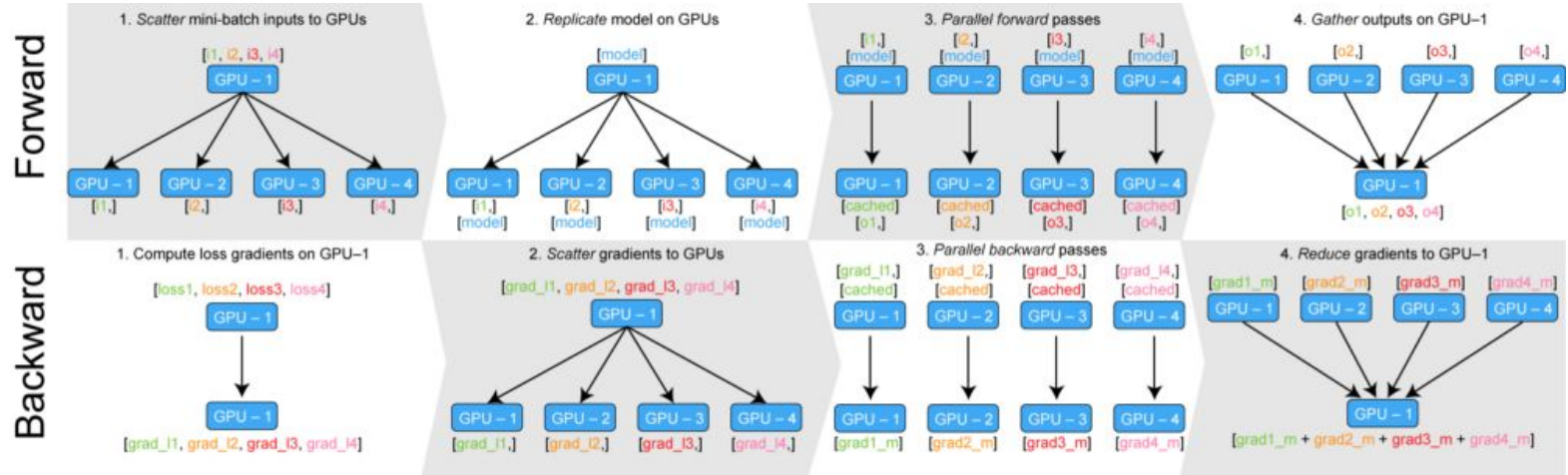
# Weight sharing

- По сути, в конце LM стоит логистическая регрессия на кучу классов
- Большинство весов в LM — эмбединги и эта логрессия
- Классов столько же, сколько и разных токенов
- Эти веса примерно об одном и том же
- Давайте объединим веса эмбедингов и коэффициенты регрессии
- Параметров стало в  $\sim 2$  раза меньше

# Breaking the softmax bottleneck ([CMU, 2018](#))

- tl;dr: давайте подумаем, что происходит, когда мы из hidden получаем логиты
- Hidden-ов очень много, зависимость между ними и распределением следующих токенов одна логрессия не уловит
- Давайте просто сделаем несколько логрессий, а предсказания как-нибудь усредним (Mixture of Softmaxes)
- Клёво, скор действительно улучшается
- В теории работает, но...

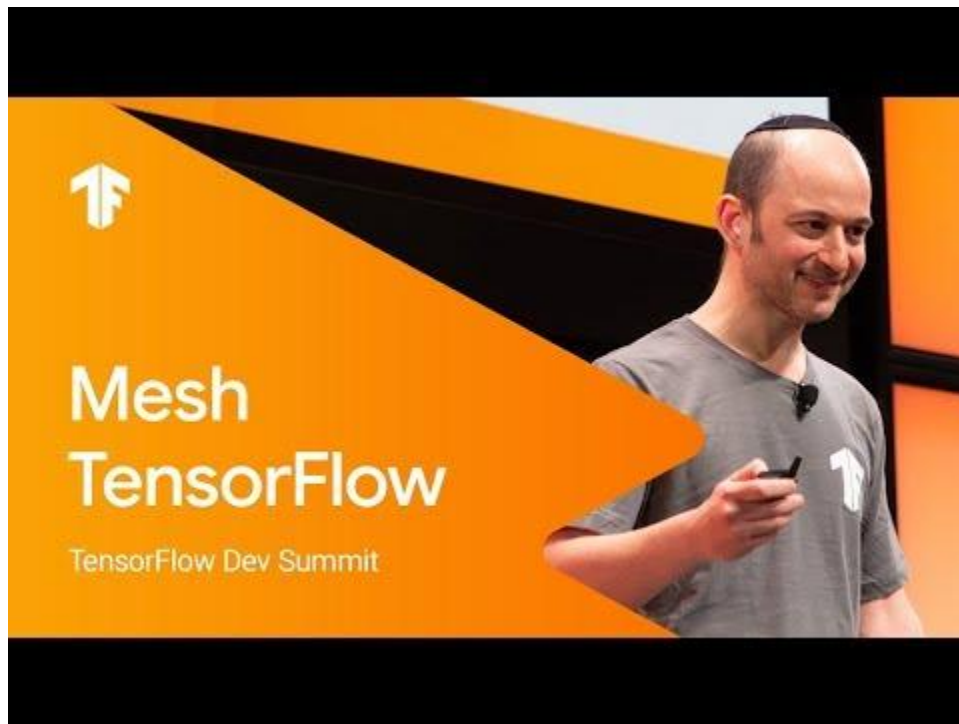
# Как работает распределенное обучение



# Память при обучении

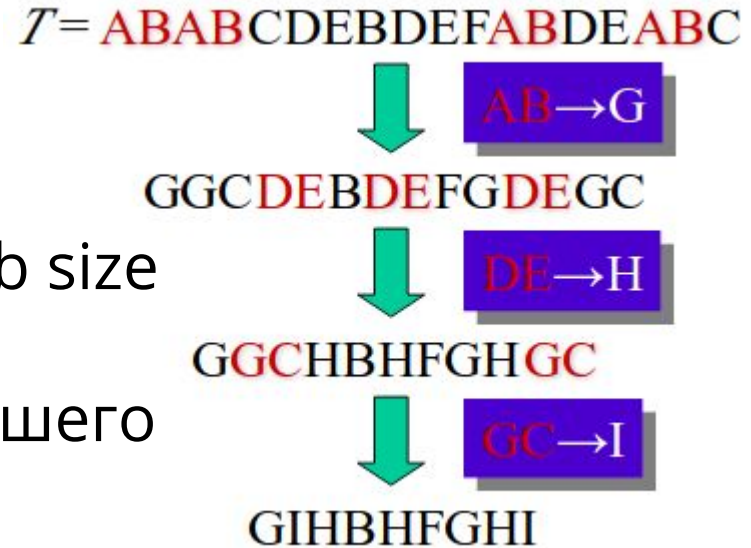
- Размер словаря = 40k
- Длины предложений = 250
- Размер батча = 32
- Таких штук нужно две (градиенты тоже)
- Берем в руки калькулятор,  
получаем 2.4гб на один только softmax
- В обычных GPU около 10гб
- **Это всё надо как минимум  
передавать по сети**

# Google хотят обучить LM на 1B параметров



# Byte-pair-encoding ([U of Edinburgh, 2016](#))

- Возьмем текст, посмотрим на все пары соседних символов
- Заменяем самую частую пару на новый символ
- Goto 1, пока не заполнили vocab size
- Подходит странных языков со сложной морфологией, типа нашего
- Заходит в машинном переводе — универсальная токенизация для любого языка



Звучит как какая-то дешевая эвристика  
[github.com/google/sentencepiece](https://github.com/google/sentencepiece)

tl;dr: давайте сделаем слишком много BPE, а потом будем выкидывать самые бесполезные токены (по мнению униграммной языковой модели)

Текущий state-of-the-art:  
[openai.com/blog/better-language-models/](https://openai.com/blog/better-language-models/)



# Практическая часть

