



Введение в обработку текста

AI Community Innopolis



План на сегодня

1. Задачи обработки текста

- Классификация / регрессия
- Сегментирование текста (Sequence Labeling)
- Моделирование языка
- Машинный перевод (Sequence-to-Sequence)
- Семантическое сходство

2. Методы обработки текста

- Токенизация
- Лемматизация
- Byte-Pair Encoding

3. Извлечение признаков из текста

- Bag-of-words
- TF-IDF
- Word2Vec
- FastText
- ELMo
- BERT и друзья
- Агрегация эмбедингов слов



Задачи обработки текста

Что такое текст?



Статья 1

Внести в Федеральный закон от 7 июля 2003 года № 126-ФЗ

"О связи" следующие изменения:

1) в статью 2 дополнить пунктом 28.5 следующего содержания:

"28.5) точка обмена трафиком – сооружения связи и (или) совокупность средств связи, с использованием которых собственник или их владелец обеспечивает возможность для соединения (включая прямое взаимодействие) и пропуска трафика между сетями связи операторов связи, собственников или владельцев технологических сетей связи, а также иных лиц, имеющих номер автономной системы;"

2) в части 2 статьи 12:

а) абзац 2 изложить в следующей редакции "определяет порядок их взаимодействия;"

б) дополнить абзацем 5 следующего содержания:

Что такое текст?



Mariya Moiseeva

15:01

Консьерж-сервис Иннополиса

Дорогие жители !!! В здании АДЦ. им. Попова проходит мероприятие с жёстким контрольно-пропус...
А как быть с тем, что доставку не пропускают в город вообще? Нам везут достаточно объемную детскую кровать, а машину не пускают даже в жк.



Екатерина Малолетова

15:01

Вот интересно, значит, это было известно. Ведь в технопарк ходят не только резиденты.



Igor Moiseev

15:02

Mariya Moiseeva

А как быть с тем, что доставку не пропускают в город вообще? Нам везут достаточно объемную детс...
никак, обратно везут уже.



Mariya Moiseeva

15:03

Как в таком случае быть? Переплачивать еще раз за доставку? или грузить на себя?



Екатерина Малолетова

15:03

Mariya Moiseeva

А как быть с тем, что доставку не пропускают в город вообще? Нам везут достаточно объемную детс...
Что за ужас! Совсем что ли уже все с ума посходили? 😡

Что такое текст?



```
class MaskedKMaxAveragePooling(nn.Module):
    """
    Performs adaptive max (or k-max) pooling.
    It takes some maximum statistics (not exactly top k elements) and averages them.
    :arg k: number of maximum elements to pick
    """
    def __init__(self, k=3):
        super().__init__()
        self.k = k

    def forward(self, x, mask):
        """
        :param x: An float tensor with shape of [batch_size, seq_len, s]
        :param mask: An byte tensor with shape of [batch_size, seq_len]
        Equals one for padding elements and zero for everything else.
        :return: An float tensor with shape of [batch_size, s]
        """
        weights = torch.ones_like(mask).float()
        weights.masked_fill_(mask, 0)
        weights = weights.unsqueeze(2)
        x = (x * weights)
        x = x.transpose(1, 2) # [batch_size, s, seq_len]
        top_k = F.adaptive_max_pool1d(x, output_size=self.k) # [batch_size, s, self.k]
        return top_k.mean(dim=2)
```



Что такое текст?

- Что-то из символов
- Символы идут по порядку
- Иногда символы группируются вместе (например, в слова)




Корпусы

Корпус – набор текстов на одном языке.

- Все сообщения в чатах Иннополиса
- Художественные произведения на русском
- Википедия
- Весь код на Python из Гитхаба
- Новости, твиты, статьи...
- Научные корпусы: НКРЯ, Тайга, OpenCorpora

Национальный корпус русского языка





НАЦИОНАЛЬНЫЙ КОРПУС
РУССКОГО
ЯЗЫКА

главная

архив новостей

поиск в корпусе

что такое корпус?

состав и структура

статистика

графики

частоты

морфология

обороты

синтаксис

семантика

параметры текстов

studiorum

форум

Частотное распределение популярных словоформ и словосочетаний (основной корпус)

[Словоформы](#) [2-граммы](#) [3-граммы](#) [4-граммы](#) [5-граммы](#) [6-граммы](#)

При подсчёте не учитывались знаки препинания и регистр.
Приведены результаты, встречающиеся не менее чем в 100 документах.
Объём всего корпуса: 192689044 словоформы.

№	Словосочетание	Документы	Частота
1	о том что	11793	37235
2	в том что	12321	36961
3	до сих пор	8947	24284
4	и т д	6916	22828
5	для того чтобы	7661	19722
6	в том числе	9206	19309
7	в то время	6072	19037
8	в это время	4456	16541
9	в то же	6130	16088
10	по крайней мере	5195	15865
11	то же время	6032	15695



Классификация текстов

Текст → категория, свойственная тексту

- Намерение пользователя
 - “Слушай Алиса, какая погода?” → /get_weather
- Определение тональности
 - “дебилы у вас там сидят” → “негатив”



Регрессия по тексту

Текст → численная характеристика

- Рейтинг отзыва
 - “супер фильм, но затянут” → 4
- Оценка стоимости тендера
 - “закупка станков” → 32 млн ₽
- Определение возраста пользователя
 - “что по матеше задали” → 14



Сегментирование текста

Текст → токены → категория токена

- Part-of-speech Tagging
 - [“я”, “люблю”, “собак”] → [“местоимение”, “глагол”, “существительное”]
- Named Entity Recognition
 - [“Путин”, “посетил”, “Иннополис”] → [“личность”, “---”, “город”]

Моделирование языка



Корпус → генеративная модель

5-gram

Can you please come **here** ?



History



Word being predicted

$$P(w_1, \dots, w_m) = \prod_{i=1}^m P(w_i \mid w_1, \dots, w_{i-1}) \approx \prod_{i=1}^m P(w_i \mid w_{i-(n-1)}, \dots, w_{i-1})$$

Моделирование языка



Позволяет провести анализ корпуса текстов, не требует разметки

Работает в подсказках в сенсорных клавиатурах

Основа для более сложных методов обработки текстов

СМИ сообщили о закрытии «Роснано»
«Газпром» объявил профинансирован перед Россией
«Агрессивный мир» отдала проводить российские компании
i 3300, train 4.546863555908203 test 4.691991806030273

В Москве задержали 16 тысяч человек за изнасилование детей
В сети появились антисесурий для российской военной авиации
Сотрудники МЧС подтвердили причастность к теракту в Дагестане
«Мы проочность, которые мы, геты жить»
В Москве пройдет фестиваль фестиваль для институтов
i 3400, train 4.516148090362549 test 4.798953056335449

Россия и Археологи сообщили о готовности Турции и Киевом
Шотландский судья отказался отпуск на должность президента по Пограничному
В Британии нашли тайный в 2017 году бомбардировщик «Амата»
IAAF объяснила отсутствие продаж в России
СМИ сообщили о планах Роскосмоса за «самую»
i 3500, train 4.5042595863342285 test 4.536854267120361

«Невозь не детям сказать»
Лавров назвал Обаму «копедо»
В Москве задержали двух подозреваемых в убийстве трех человек
В сети появилось видео убийства Вороненкова в Забайкалье
В Москве осудили поддельную помощь в заложники двух человек

Семантическое сходство (Text Similarity)



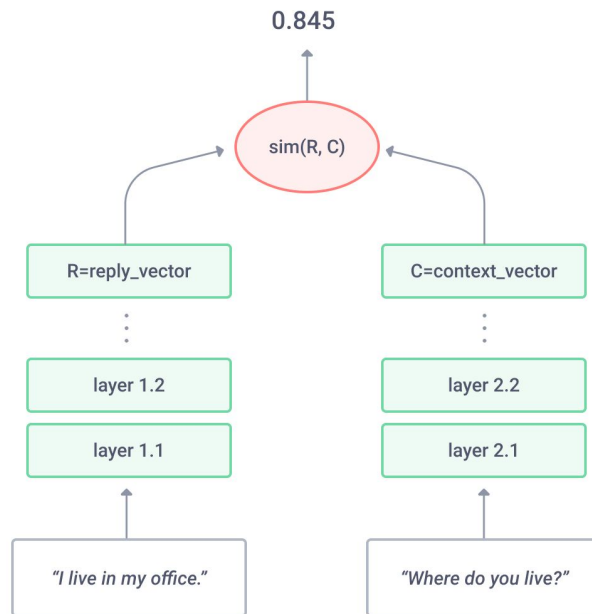
Текст_1 → эмбеddинг_текста_1

Текст_2 → эмбеddинг_текста_2

эмбеddинг_текста_1, эмбеddинг_текста_2 →
похожесть текстов

Позволяет находить схожие тексты, помогает с
поиском документов.

Эмбеddинги используются как признаки в
классификаторах.





Машинный перевод

Текст → токены → токены

- Машинный перевод (“where are my dragons” → “где мои драконы”)
- Суммаризация (текст новости → заголовок)
- Диалоговая система (“как дела” → “норм, сам как”)

По сути, это та же языковая модель, но которая принимает как вход какой-то текст.

Подход распространяется на другие задачи, которые сводятся к последовательностям чего-либо (например, речь → текст)



Методы обработки текста



Токенизация

Это такая декомпозиция задачи — переход к более понятным элементам, от символов к группам символов.

- Токенизация документа на предложения
- Токенизация предложения на слова



Токенизация на предложения

Весь текст делим на последовательность предложений.

```
>>> from rusenttokenize import ru_sent_tokenize
>>> text = "Пока везде закрыто. Ждём."
>>> ru_sent_tokenize(text)
['Пока везде закрыто.', 'Ждём.']
```



Токенизация на слова

Одно предложение делим на последовательность слов.

```
>>> from deeppavlov.models.tokenizers.ru_tokenizer import  
RussianTokenizer  
>>> tokenizer(['Пока везде закрыто.', 'Ждём.'])  
[['пока', 'везде', 'закрыто'], ['ждём']]
```

Считать ли знаки препинания словами?

Словарь



Набор токенов, которые известны алгоритму обработки текста.

Особенно важно задумываться об этом при работе с русским языком, который очень флективен:

“Му” ↔ “мой”, “моя”, “мое”, “мои”, “моим”

Если токена нет в словаре, то его пропускают или заменяют на токен <OOV> / <UNK> (out-of-vocabulary / unknown).



Лемматизация

Один из способов снизить размер словаря — привести все слова к нормальному виду (например к единственному числу, именительному падежу).

```
>>> tokenizer = RussianTokenizer(lemmas=True)
>>> tokenizer(['Пока везде закрыто.', 'Ждём.'])
[['пока', 'везде', 'закрытый'], ['ждать']]
```



Снятие (морфологической) омонимии

Обратная проблема: одно слово из словаря может иметь разные роли в предложении.

“Печь пироги” ↔ “присел на печь”

Одно из решений проблемы — определение части речи слова и приписывание ее к токену:

“Печь пироги” → [“печь_VERB”, “пирог_NOUN”]

“Присел на печь” → [“присесть_VERB”, “на_PREP”, “печь_NOUN”]



Byte-Pair Encoding

Другой способ — дробить предложение не на слова, а на подслова. Один из алгоритмов, которые находят набор фиксированного размера таких подслов — BPE.

“миша ест котлету” → [“_ми”, “ша”, “_е”, “ст”, “_кот”, “лет”, “у”]

NB: BERT активно используют именно такой подход (алгоритм SentencePiece).

Полезно: <https://github.com/huggingface/tokenizers>



Извлечение признаков из текста

One-hot Encoding (ONE)



Токены → индексы в словаре

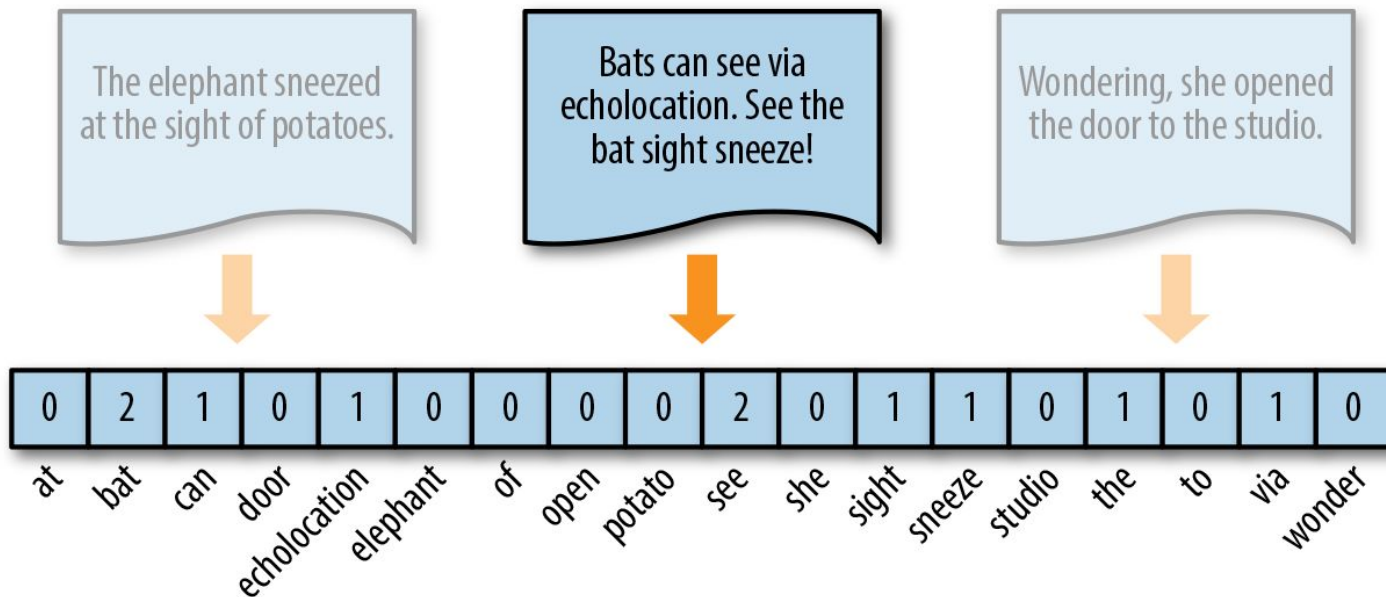
Плюсы: просто

Минусы: разреженность, не отражает похожесть токенов (синонимы и антонимы одинаково далеки)

Размер словаря

"a"	"abbreviations"	"zoology"
1	0	0
0	1	0
0	0	0
•	•	•
•	•	•
•	•	•
0	0	0
0	0	1
0	0	0

Bag-of-words



В таком представлении слов их легко агрегировать для одного предложения: достаточно сложить. Однако теряется информация о порядке слов.

TF-IDF



Не все йогурты слова одинаково полезны. Редкие слова несут больше информации, чем частые, однако в ONE они имеют равный вес.

TF-IDF — замена единице в One-hot Encoding, которая выполняет именно эту роль. Эта величина больше для редких слов и меньше для частых.

TF-IDF

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

$$\text{TF-IDF} = \text{TF}(t, d) \times \text{IDF}(t)$$

Term frequency

Number of times term t appears in a doc, d

Inverse document frequency

$$\log \frac{1 + \overset{\text{\# of documents}}{n}}{1 + \underset{\text{Document frequency of the term } t}{df(d, t)}}$$

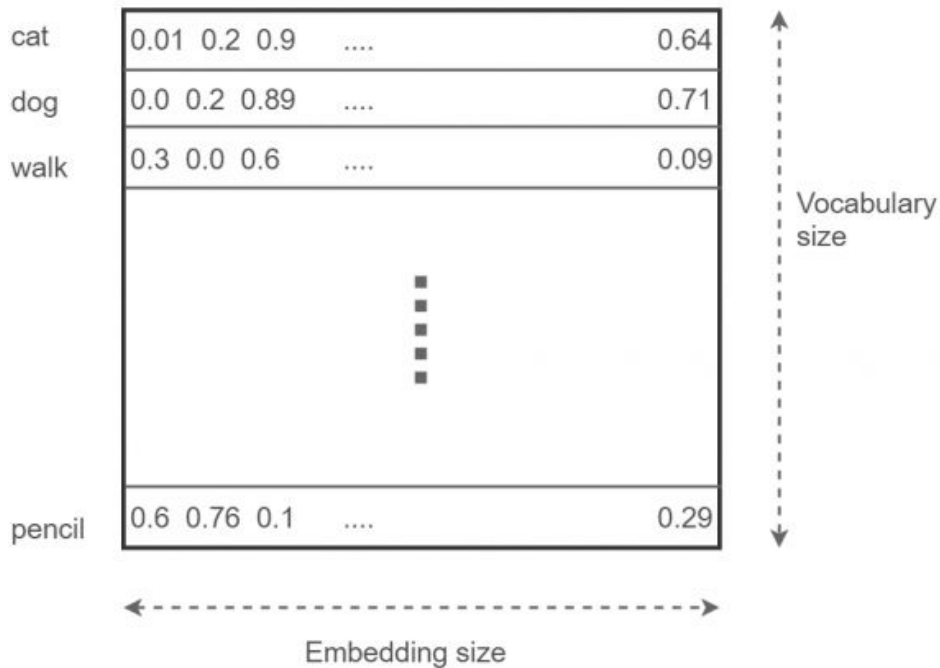
Word2Vec — эмбединги слов



Токены → векторы из семантического пространства

Плюсы: синонимы похожи, слова разных смыслов непохожи

Минусы: вектор слова не зависит от контекста (“«лист» дерева” и “«лист» бумаги” — для «лист» один вектор в обоих случаях)



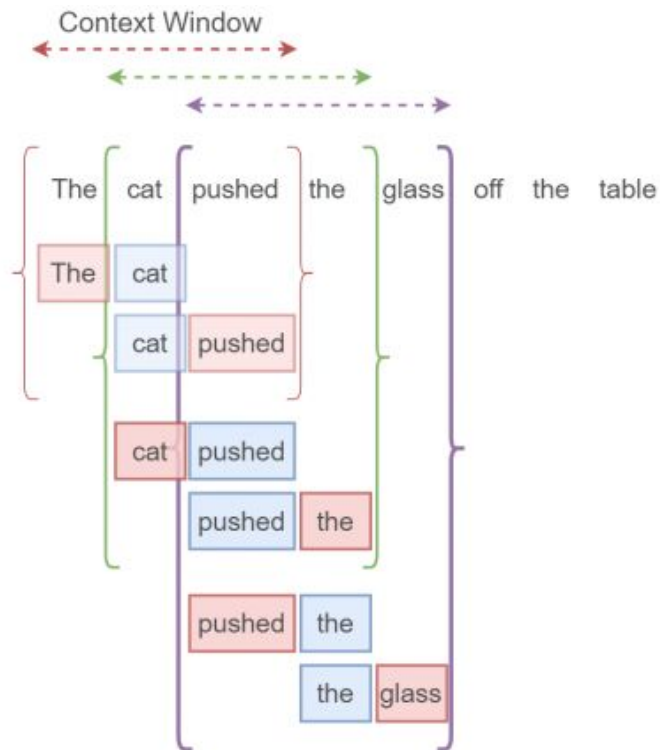
Word2Vec



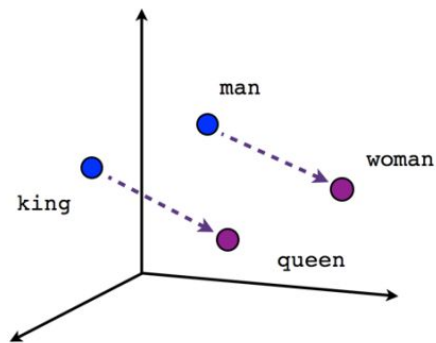
Как построить векторы для слов?

- Много текстов без разметки
- Слово → случайный вектор
- Вектор слова → вектор вероятностей слов контекста без самого слова (Skip-gram)
- Учим: знаем какие слова были в контекстах и какие не были

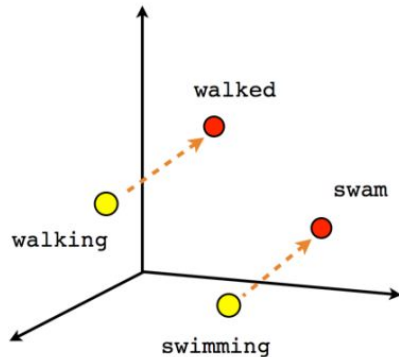
В итоге имеем логистическую регрессию, количество классов = размер словаря



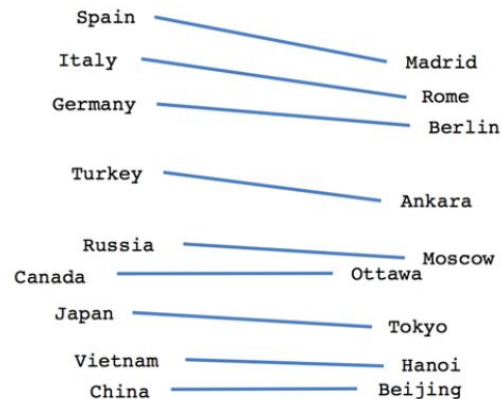
Word2Vec



Male-Female



Verb tense



Country-Capital

Дистрибутивная семантика: свойства слов выражаются векторами.

$$\text{“king”} - \text{“man”} + \text{“woman”} = \text{“queen”}$$

FastText



Слово моделируется набором посимвольных n-грамм.

Для каждой формируется отдельный вектор, вектор слова – сумма векторов n-грамм.

За счет такого построения — более стоек к опечаткам и позволяет обрабатывать слова вне словаря.

Represent a word as a bag of character n-grams, e.g. for $n = 3$:

$G_{where} : \text{\textit{_wh, whe, her, ere, re_ , _where_}}$

Model a word vector as a sum of sub-word vectors:

SGNS:

$$\textit{sim}(u, v) = \langle \phi_u, \theta_v \rangle$$

FastText:

$$\textit{sim}(u, v) = \sum_{g \in G_v} \langle \phi_u, \theta_g \rangle$$

ELMo — Embeddings from Language Models



Эмбединги формируются при помощи рекуррентной нейронной сети, строящей языковую модель.

Результат: эмбединг слова зависит от контекста.

Решается проблема омонимии.



<https://jalammar.github.io/illustrated-bert/>

ELMo — Embeddings from Language Models

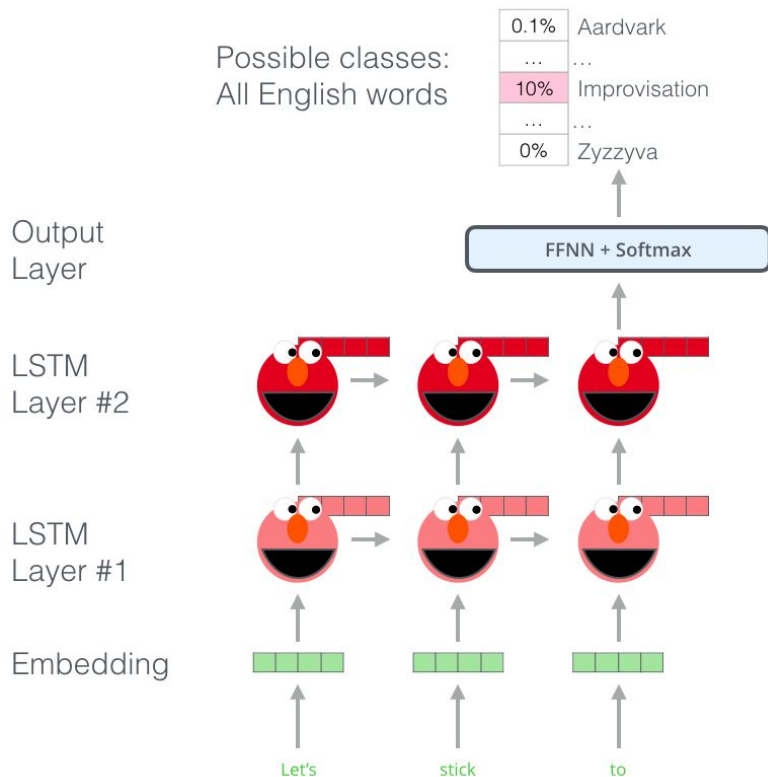


Рекуррентная нейронная сеть обучается как языковая модель.

В процессе получают скрытые состояния для каждого токена последовательности.

Эти признаки зависят как от текущего токена, так и от предыдущих.

<https://jalammar.github.io/illustrated-bert/>



ELMo — Embeddings from Language Models



На деле языковая модель строится на основе признаков, полученных в обоих направлениях.

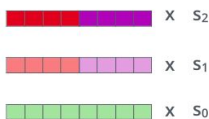
Эмбеддинг получается конкатенацией промежуточных скрытых состояний в обоих направлениях, а затем суммой с весами и по слоям векторов скрытого состояния для каждого слова.

Embedding of “stick” in “Let’s stick to” - Step #2

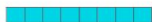
1- Concatenate hidden layers



2- Multiply each vector by a weight based on the task

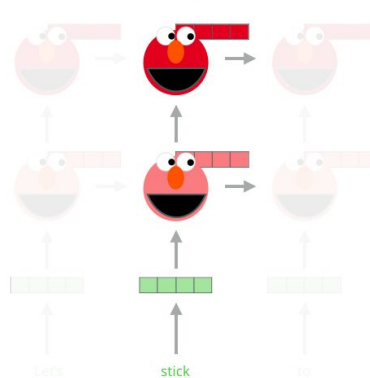


3- Sum the (now weighted) vectors

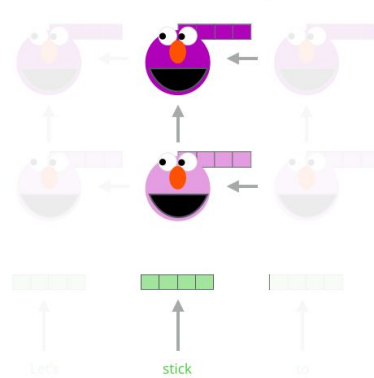


ELMo embedding of “stick” for this task in this context

Forward Language Model



Backward Language Model



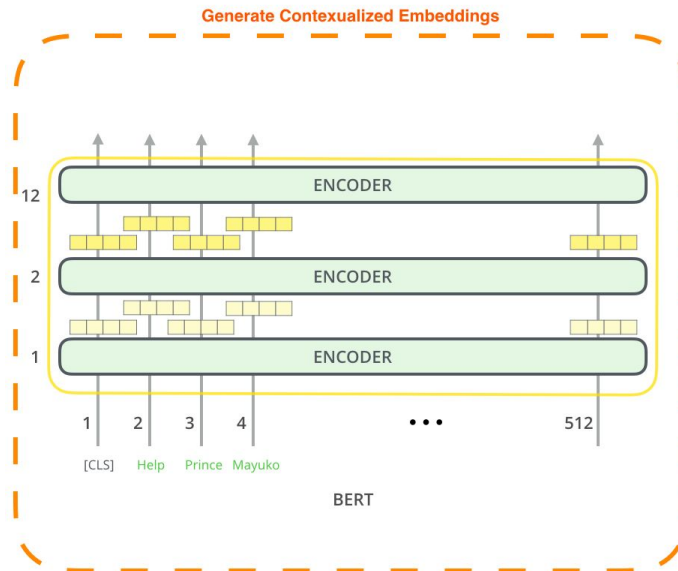
<https://jalamar.github.io/illustrated-bert/>

BERT и его друзья

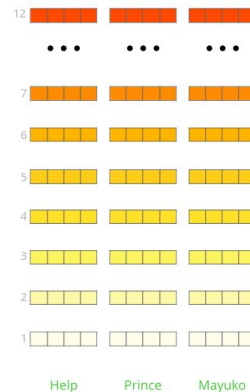


Усложнение концепций ELMo:

- Transformer + Attention + Positional Encodings вместо LSTM
- Дополнительные целевые функции
- Экзотика, которая зависит от конкретной вариации



The output of each encoder layer along each token's path can be used as a feature representing that token.



But which one should we use?

<https://jalammar.github.io/illustrated-bert/>

Агрегация эмбеддингов слов



Так, мы получили эмбеддинги для слов, как получить эмбеддинг предложения?

- Пуллинг
 - С весами TF-IDF (бейзлайн для маленьких предложений)
 - Учить веса слов как параметры или предсказывать их отдельной нейронкой
 - Попробовать различные [пуллинги](#) (усреднить == Average Pooling)
 - Сконкатенировать эмбеддинги от различных пуллингов вместе
- Модель сама возвращает эмбеддинг предложения
 - В BERT эмбеддинг токена <CLS> считается эмбеддингом всего предложения (в fine-tuning в задаче классификации)
 - В моделях skip-thought эмбеддинг предложения — эмбеддинг токена <EOS>, который добавляют в конце предложения