17/04/25

Векторное представление слов на основе USE .

Доклад подготовил:
Поляков Михаил

Структура доклада

1. Векторные представления слов

Что такое векторные представления и какие есть способы.

2. USE

Архитектура Universal Sentence Encoder.

3. Обучение USE

Как обучался USE.

4. Применения USE

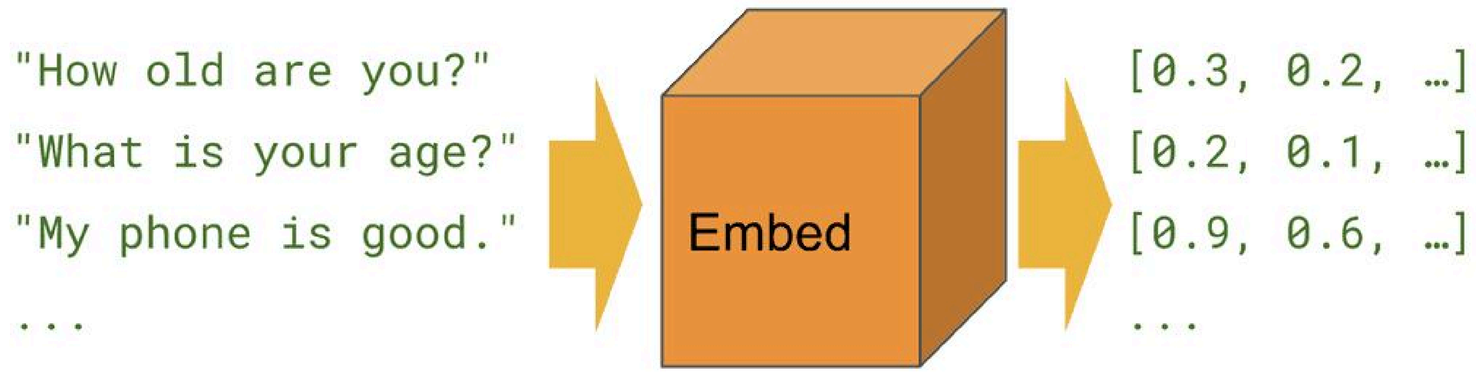
Рассмотрим где и как применяется USE.

Векторные представления слов

Как и почему?

Векторные представления слов

Векторным представлением ~ Эмбеддингом текста называется способ представления текстовых данных с помощью векторов произвольной размерности, кодирующих смысл текста с возможностью дальнейшего сравнения.



Векторизация слов

Word2vec

Способ выучивания эмбеддингов слов на основе их контекста.

GloVe

Разработанный чуть позднее Word2vec алгоритм, учитывающий контекст и глобальную статистику встречаемости.

	document	first	is	one
document	0	2	1	0
first	2	0	1	0
is	1	1	0	0
one	0	0	0	0

FastText

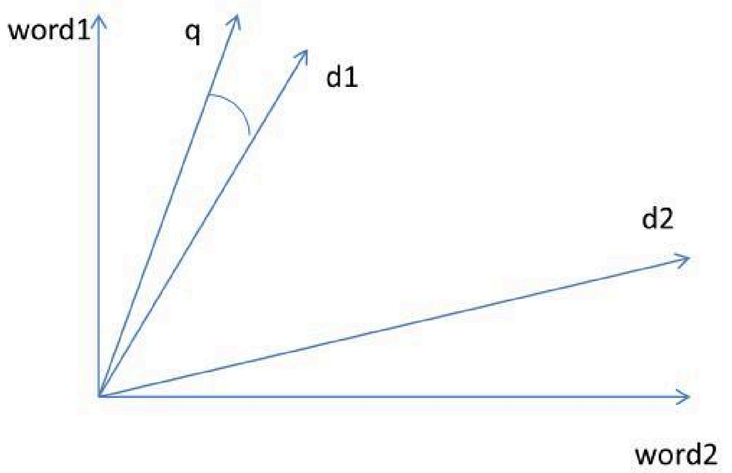
Модель разработанная в facebook, работающая с sub-word эмбеддингами.

Косинусное расстояние

Самая часто используемая мера.
Так как сравнение по углу = сравнение по направлению.
А направления имеют свои смыслы.

Косинусная мера сходства

$$\mathrm{sim}(q,d) = \frac{(q,d)}{\|q\| * \|d\|}, \qquad \|d\| = \sqrt{\sum_{k=1}^{n} d_k^2}$$



Старые подходы к обработке предложений

OHE

Представление документов вектором из 0 / 1 для слов из текста.

Bag Of Words

OHE с учётом частоты слов.

TF-IDF

Представление документов через tf-idf значения для слов.

Усреднение эмбеддингов

Эмбеддинг предложения - среднее эмбеддингов его слов.

Проблемы старых решений

Потеря контекста

Не учитывают порядок и связи между словами.

```
>>> nlp('It is cool').similarity(nlp('It'))
0.8963861908844291
```

Не учитывают семантику предложения

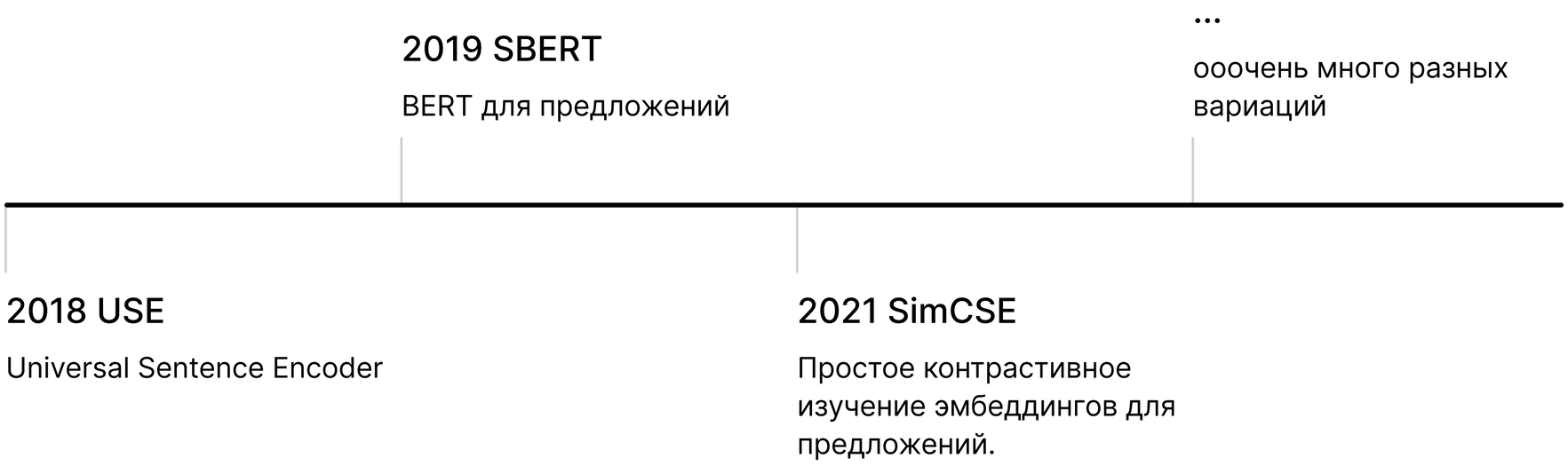
Даже усреднение word2vec-векторов часто не даёт понимания смысла текста.

```
>>> nlp('this is cool').similarity(nlp('is this cool'))
1.0
```

Тяжело производить сравнение

Сравнение таких векторов малоинформативно.

Современные подходы

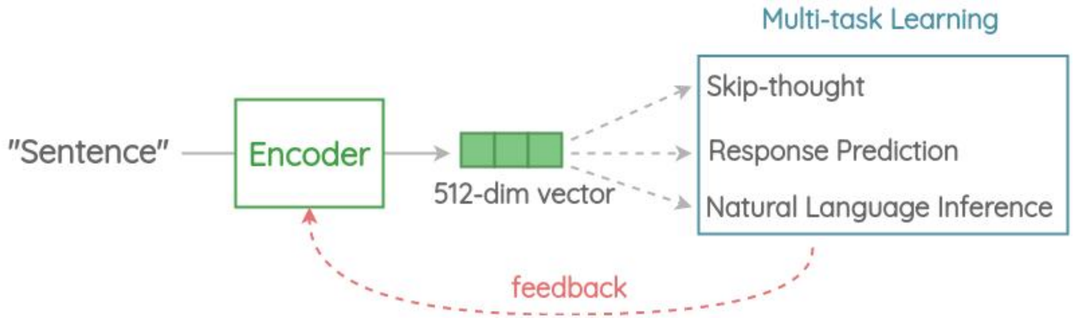


USE

Universal Sentence Encoder

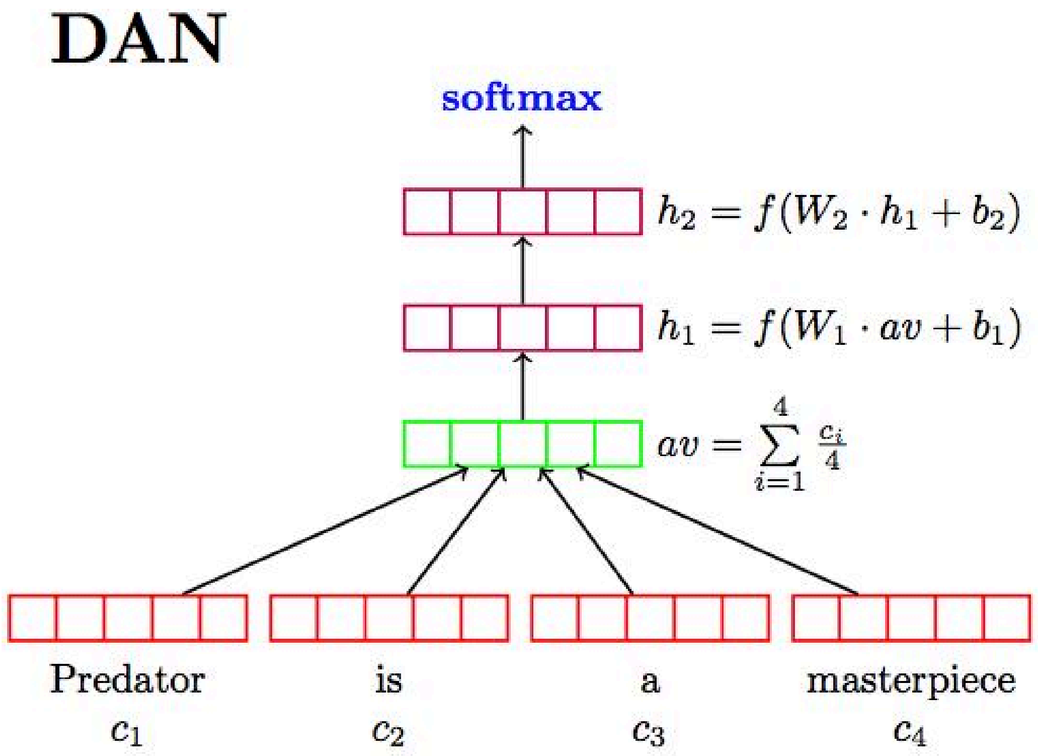
USE

USE - это модель от Google (2018), которая преобразует предложения, фразы или короткие абзацы в 512-мерные векторы.



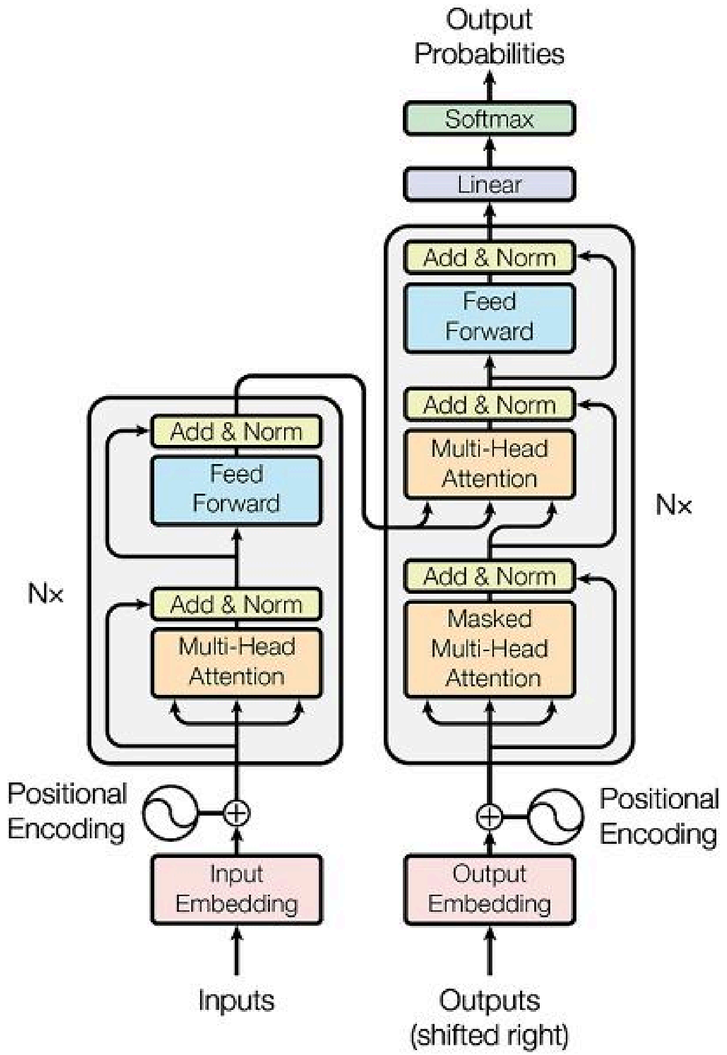
Пайплайн USE

Энкодеры USE



DAN

Deep Averaging Network



Transformer

PTB tokenizer

Алгоритм токенизации, разработанный для обработки текста в рамках проекта Penn Treebank

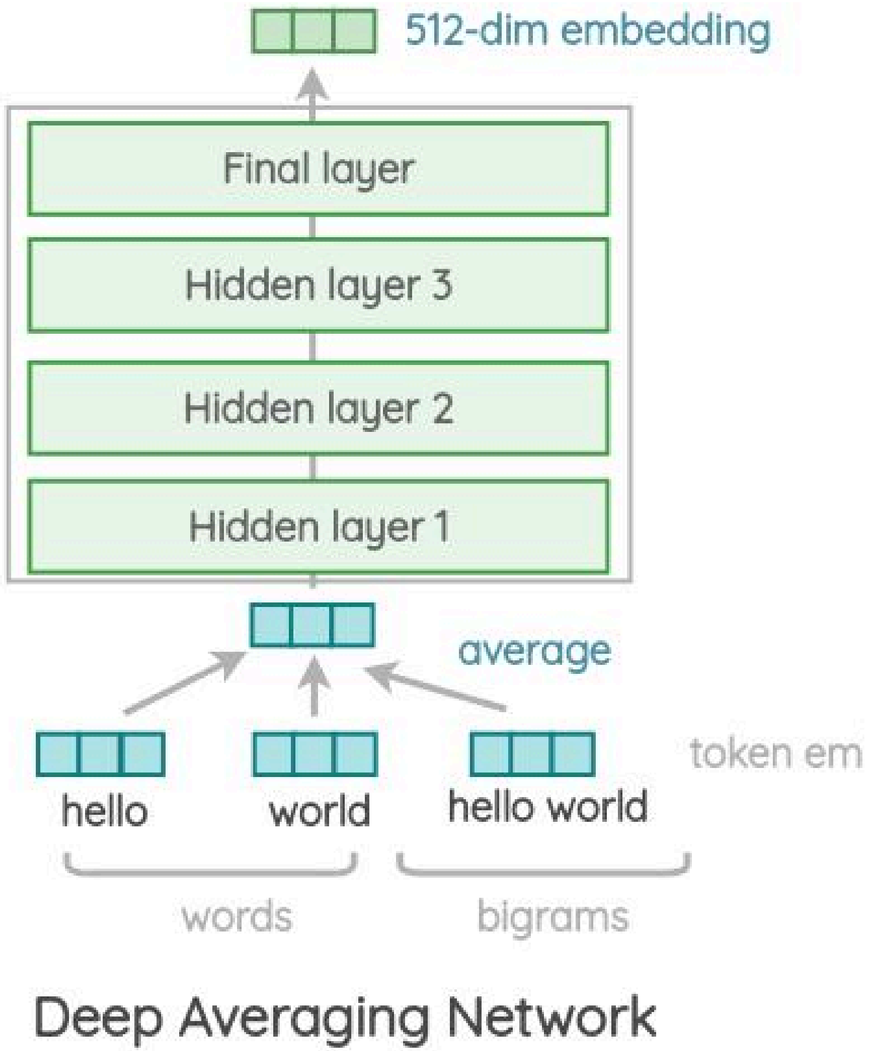
Основные правила токенизации

PTB Tokenizer применяет следующие преобразования:

1. **Разделение сокращений**:
   - `can't` → `["ca", "n't"]`
   - `I'm` → `["I", "'m"]`
2. **Отделение знаков препинания**:
   - `"Hello!"` → `["``", "Hello", "!", "''"]` (кавычки преобразуются в `` и '')
3. **Разделение валют и чисел**:
   - `$100` → `["$", "100"]`
4. **Сохранение дефисов и апострофов**:
   - `state-of-the-art` → `["state-of-the-art"]` (не разбивается)

DAN

1. Обрабатываются слова и би-граммы.
2. Усреднение исходных эмбеддингов.
3. 4 полносвязных слоя.
4. Нормализация.



Особенности

Простая архитектура

Сначала усредняются эмбеддинги слов, затем проходит через полносвязные слои.

Быстрее но менее точен

Скорость работы линейно возрастает с длинной текста.

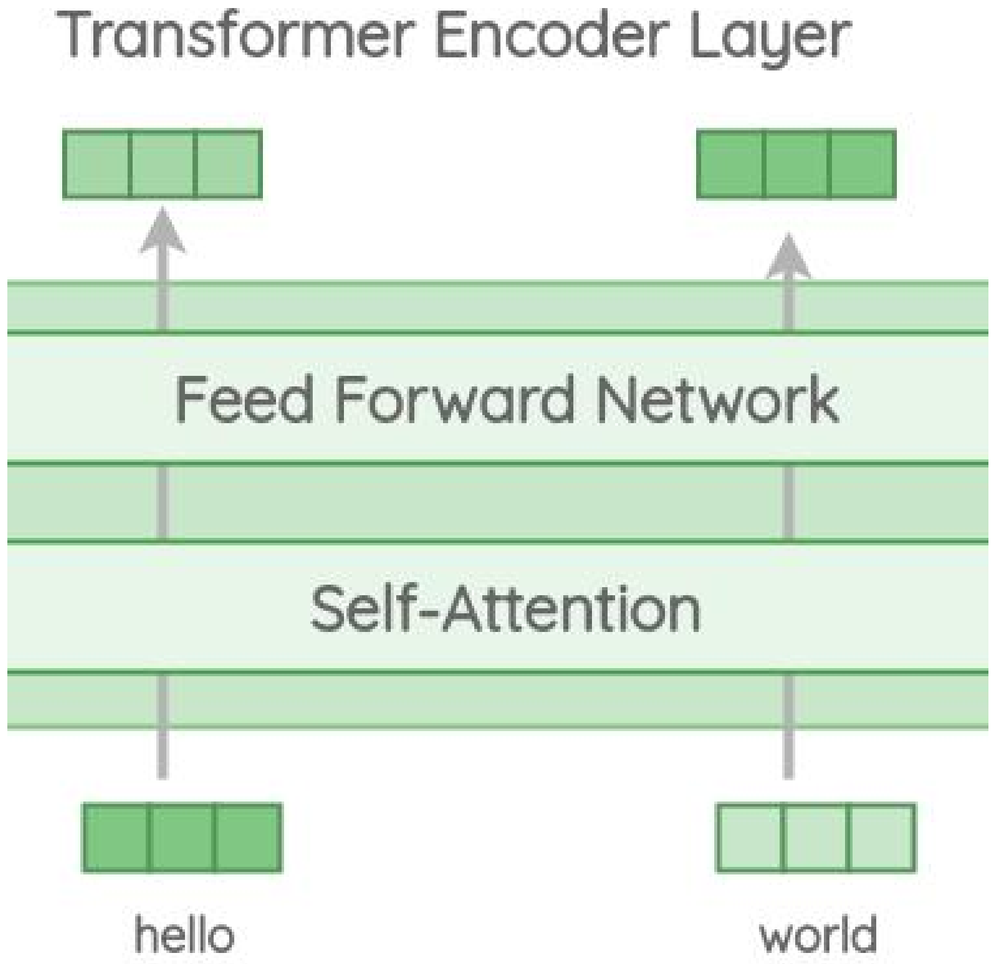
Подходит для задач в реальном времени

За счёт простоты и малых размеров.

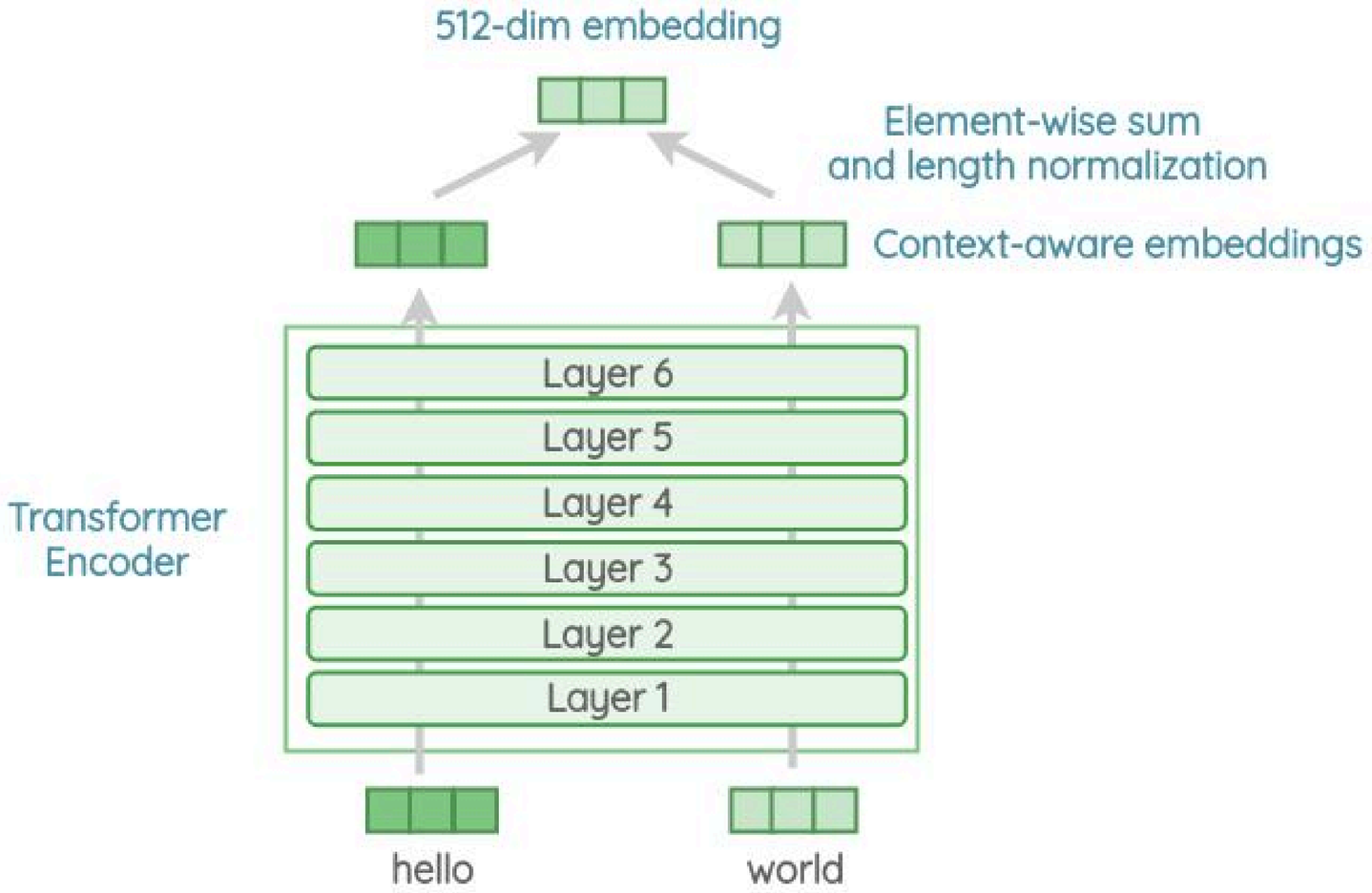
Transformer

1 слой энкодера

1. Self-attention
2. Полносвязный слой



Transformer



Архитектура энкодера

Полностью повторяет энкодер из статьи про трансформер.

Особенности

Механизм Attention

Позволяет строить контекстуализированные эмббединги, что повышает точность.

Точнее, но требует больше вычислений и памяти

Имеет квадратичную временную сложность от длины предложения.

Подходит для задач, где важна точность

позволяет добиться максимальной точности во многих задачах (была SOTA).

Обучение USE

Universal Sentence Encoder

Обучение модели

Задача 1

Modified skip-thought

Задача 2

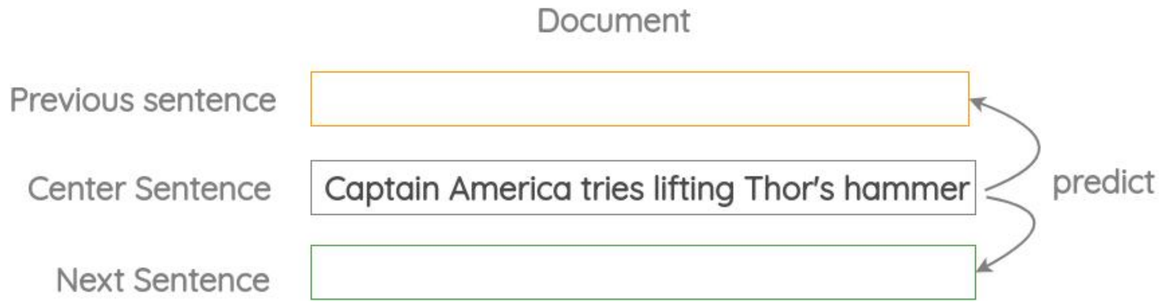
Conversational input-Response Prediction

Задача 3

Natural Language inference

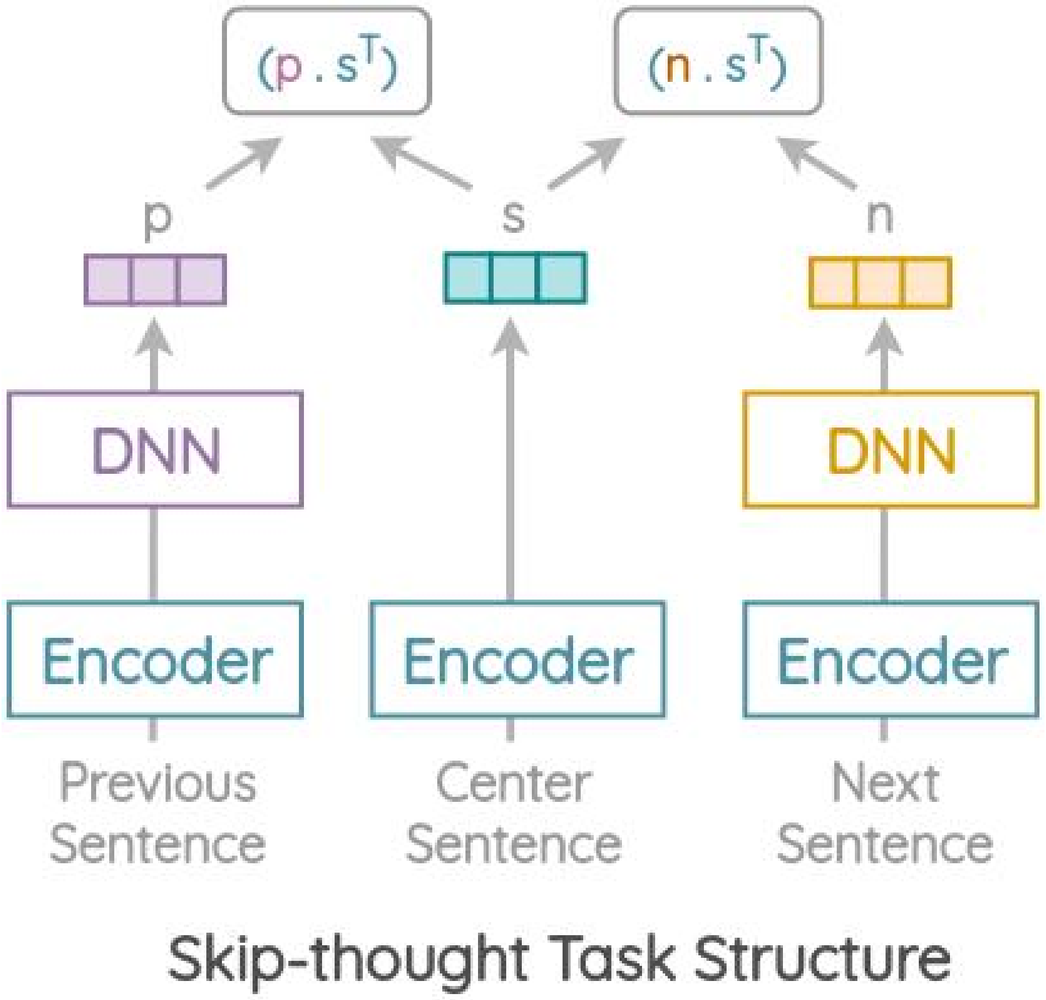
Modified skip-thought

Модель предсказывает предыдущее и следующее предложение по центральному.



https://arxiv.org/pdf/1506.06726

Схема обучения



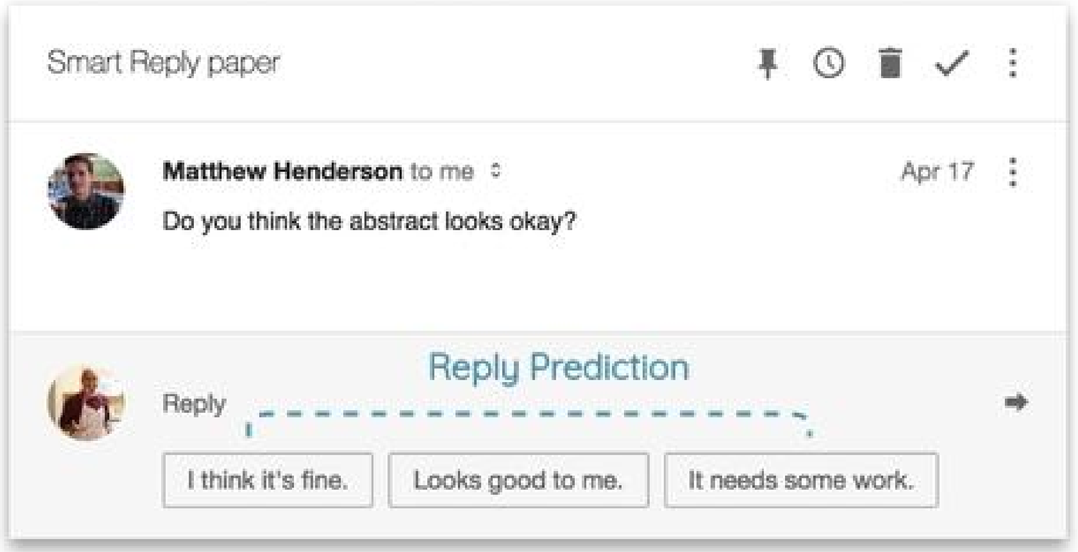
Skip-thought Task Structure

Используется **Negative Log-Likelihood (NLL)** для классификации:

$$\mathcal{L}_{\text{skip-thought}} = -\log p(s_{i-1}|s_i) - \log p(s_{i+1}|s_i)$$

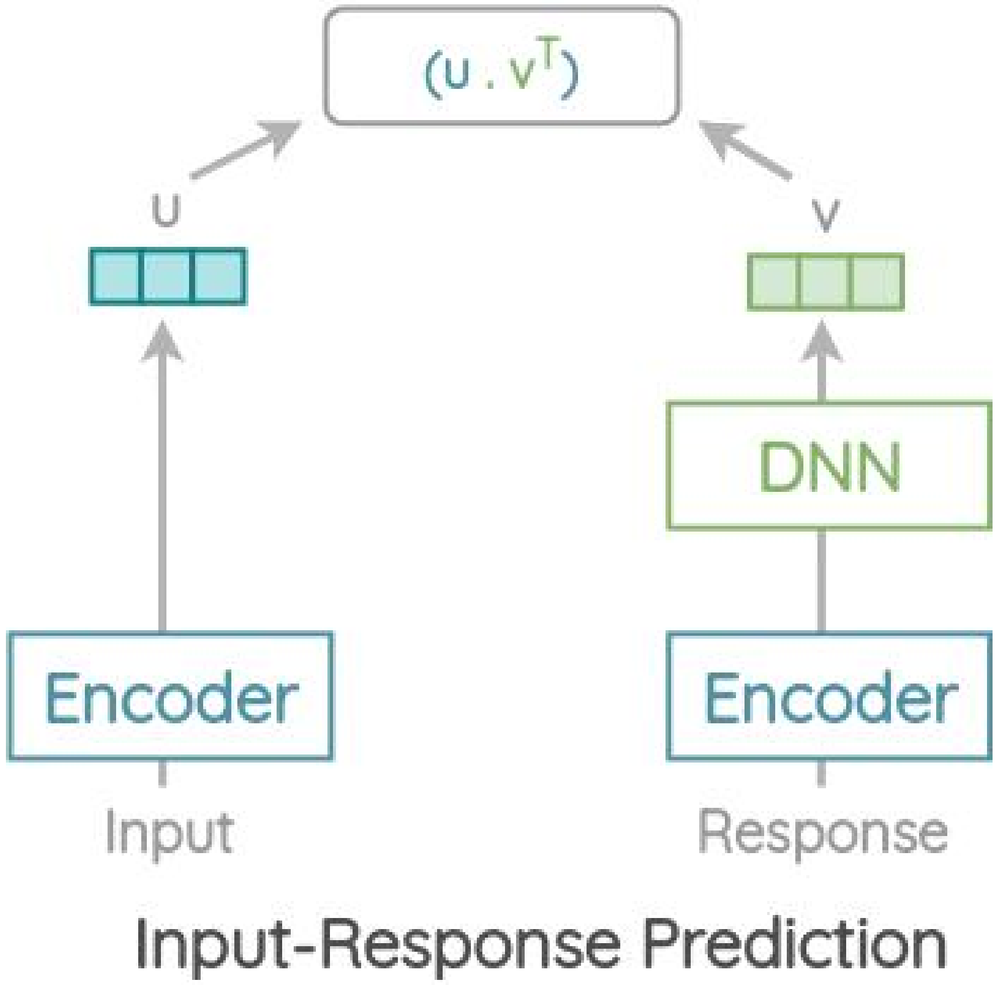
Conversational input-Response Prediction

Модель предсказывает ответ на сообщение в переписке.



https://arxiv.org/pdf/1705.00652

Схема обучения



Natural Language inference

Модель предсказывает является ли некоторое утверждение о некотором тексте

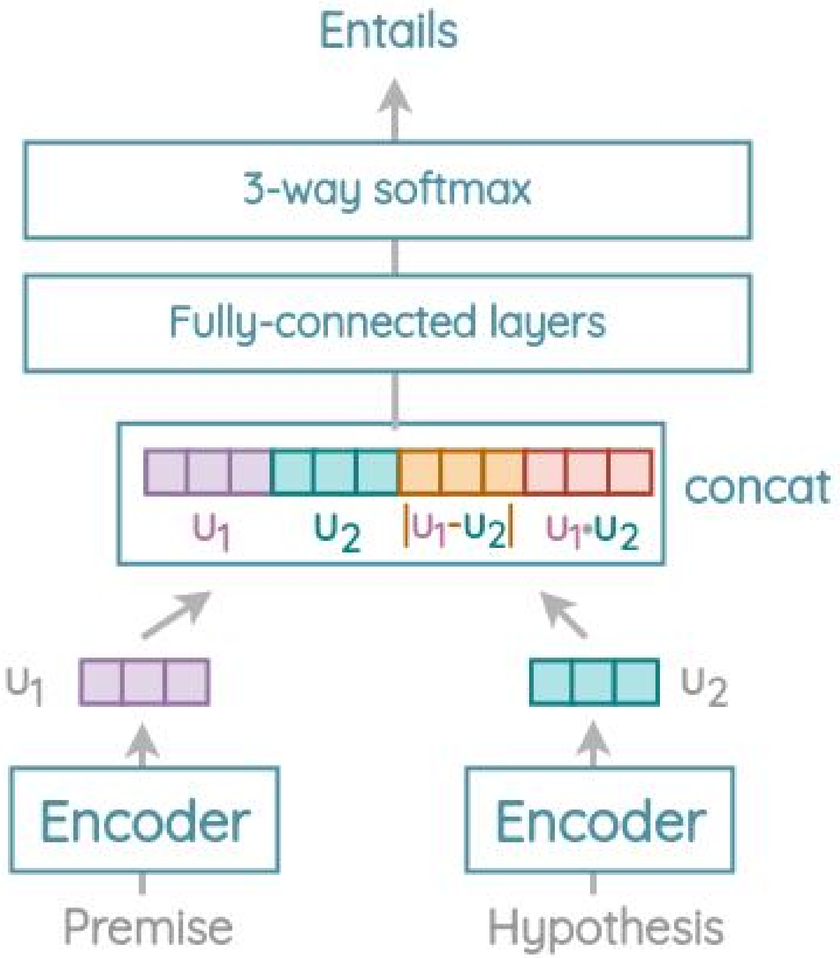
- Верным - Entailment
- Ложным - Contradiction
- Неопределённым - Neutral

Premise	Hypothesis	Judgement
A soccer game with multiple males playing	Some men are playing a sport	entailment
I love Marvel movies	I hate Marvel movies	contradiction
I love Marvel movies	A ship arrived	neutral

P[a]	A senior is waiting at the window of a restaurant that serves sandwiches.	Relationship
H[b]	A person waits to be served his food.	Entailment
	A man is looking to order a grilled cheese sandwich.	Neutral
	A man is waiting in line for the bus.	Contradiction
[a]**P**, Premise. [b]**H**, Hypothesis.		

https://arxiv.org/pdf/1705.02364

Схема обучения



Преимущества

Скорость

USE (особенно DAN-версия) обрабатывает до тысяч предложений в секунду на CPU, что критично для задач реального времени

Простота интеграции

Модель доступна в TensorFlow Hub — достаточно 2 строк кода для запуска:

Универсальность для несложных задач

Даёт приемлемое качество для поиска, сравнения текстов и классификации из коробки. Хорошее начальное решение.

```
model = hub.load("https://tfhub.dev/google/universal-sentence-encoder/4")
embeddings = model(["Текст"]).numpy()
```

Недостатки

Уступает в точности современным моделям

USE (2018) проигрывает Sentence-BERT (2019) и SimCSE (2021) в большинстве задач.

Ограниченная работа с длинными текстами

USE оптимизирован для предложений (до 50–100 слов).

Языковая и доменная ограниченность

Лучше всего работает с английским. Мультиязычные версии (USE-Multilingual) уступают современным моделям.

Применения

А для чего оно надо?

Применения USE

Семантический поиск и ранжирование.

Сравнивает вектор запроса с векторами документов через косинусную схожесть

```
query = "водонепроницаемые часы для плавания"
docs = ["часы с защитой от воды 50м", "кожаный ремешок для часов"]
query_embed = model([query])
doc_embeds = model(docs)
similarities = tf.matmul(query_embed, tf.transpose(doc_embeds))  # [0.92, 0.15]
```

Кластеризация текстовых данных.

Группирует тексты с близкими эмбеддингами.

Детекция парафраз и схожести текстов.

Нужно находить дубликаты или перефразированные тексты.

```
text1 = "Это полный отстой"
text2 = "Худший сервис в моей жизни"
sim = cosine_similarity(model([text1]), model([text2]))  # 0.89
```

Улучшение chatbots и QA-систем.

Пользователи задают вопросы по-разному, но смысл одинаков.

```
questions = ["Как восстановить карту?", "Я потерял карточку..."]
embeddings = model(questions)
if cosine_similarity(embeddings[0], embeddings[1]) > 0.9:
    print("Одинаковый интент")
```

Заключение

USE остаётся отличным решением для задач NLP, сочетая скорость обработки, простоту интеграции и достаточную точность для базовых задач.

Литература

https://arxiv.org/pdf/1803.11175

https://amitness.com/posts/universal-sentence-encoder

https://ailab.mti-vietnam.vn/blog/2021/01/31/exploring-universal-sentence-encoder-model-use/

Спасибо за внимание!