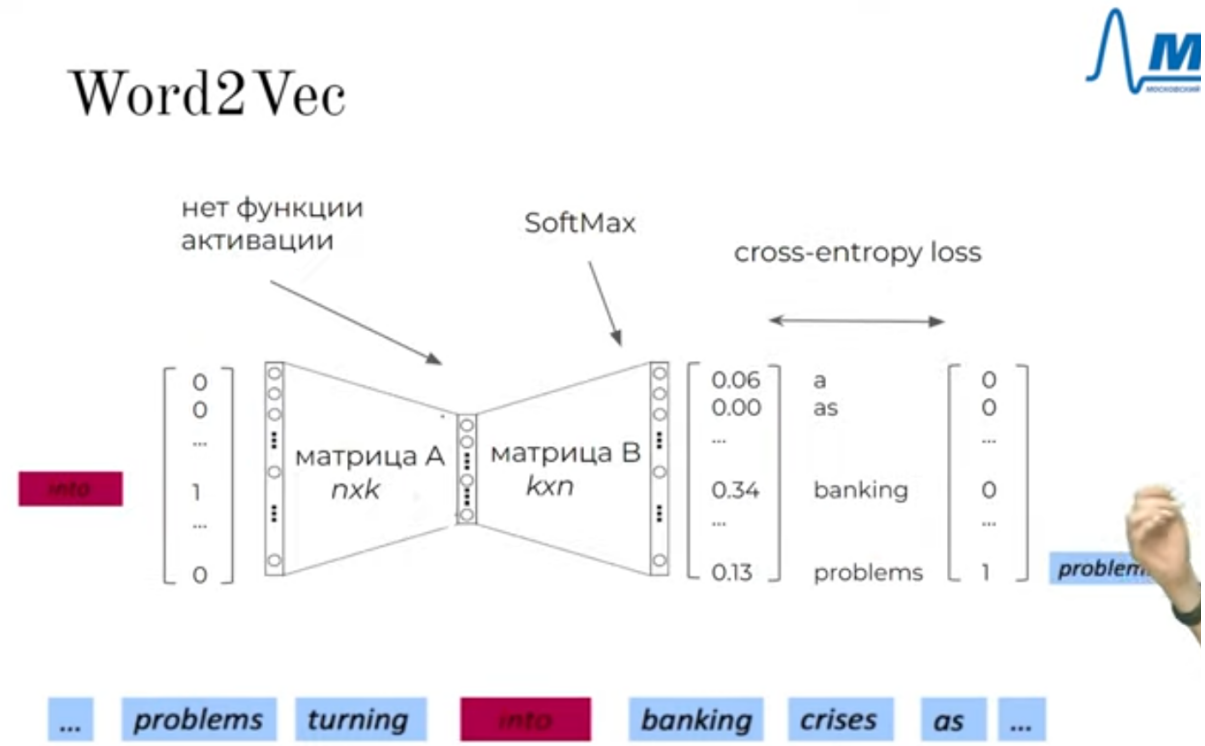
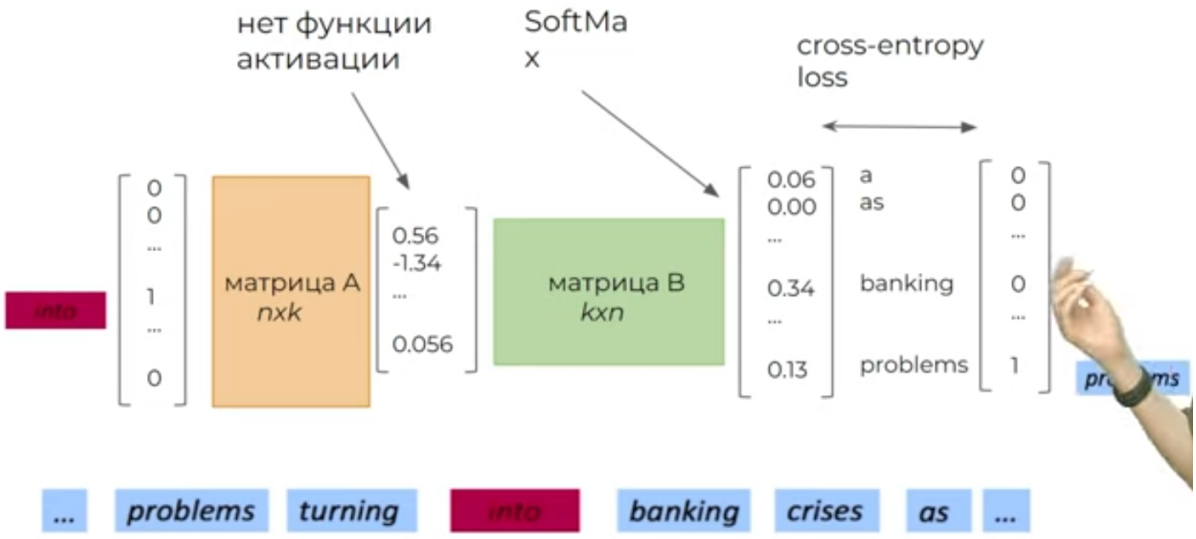
Embeddings (from DLS_NLP)

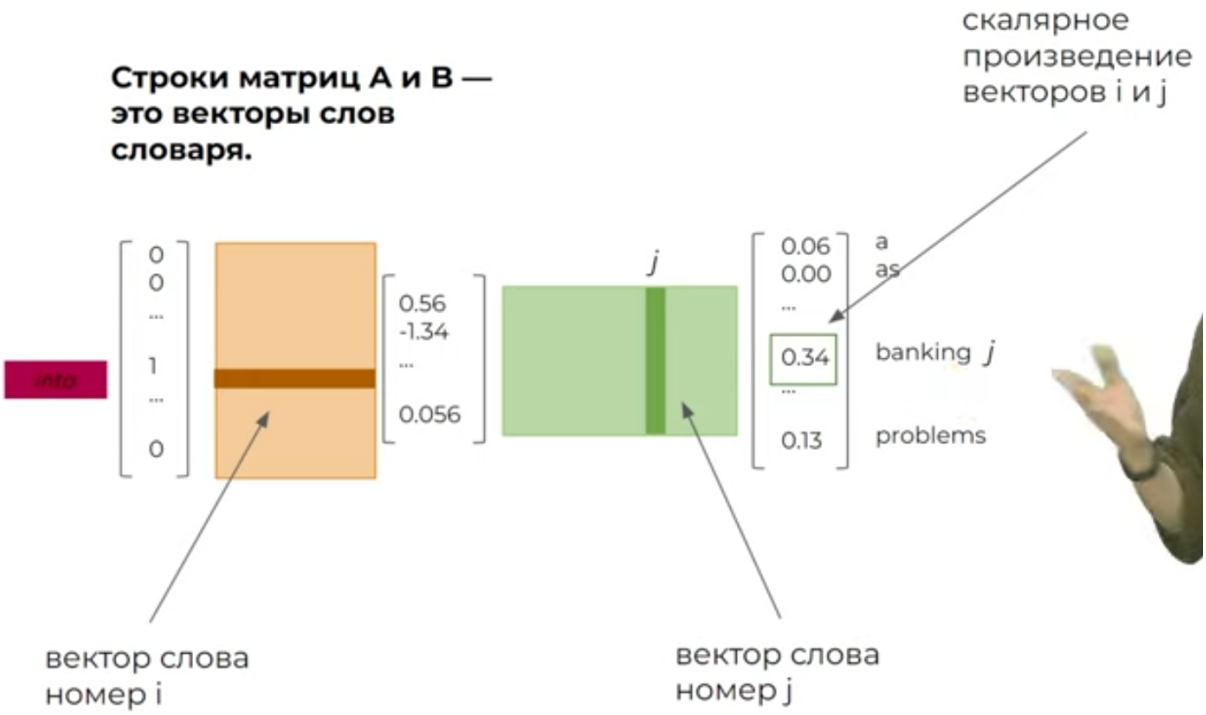


Word2Vec

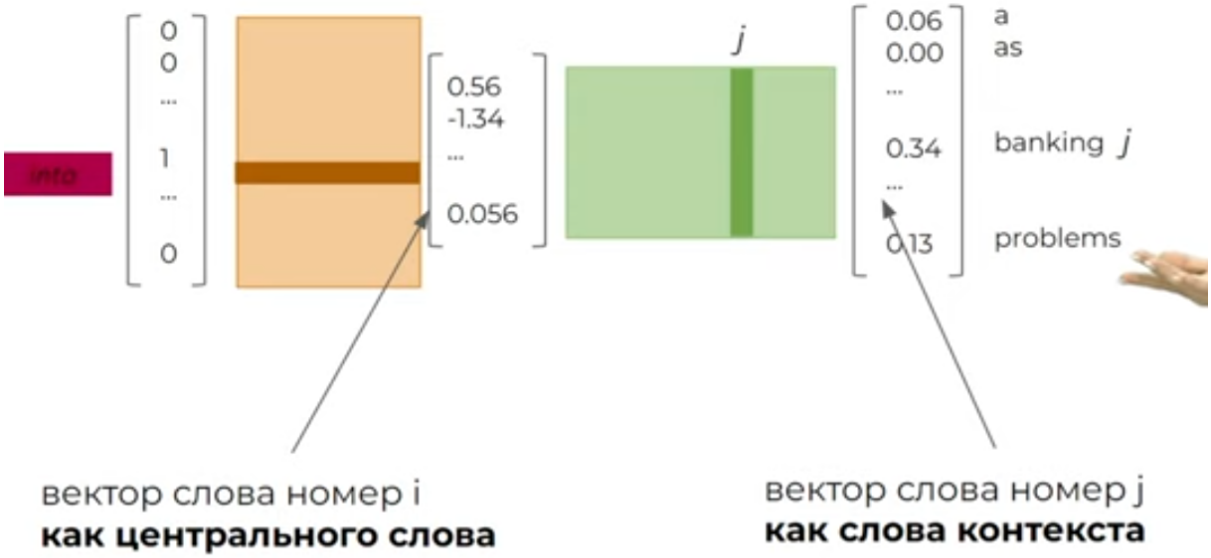


Word2Vec

Строки матриц A и B — это векторы слов словаря.



Word2Vec



вектор слова номер i
как центрального слова

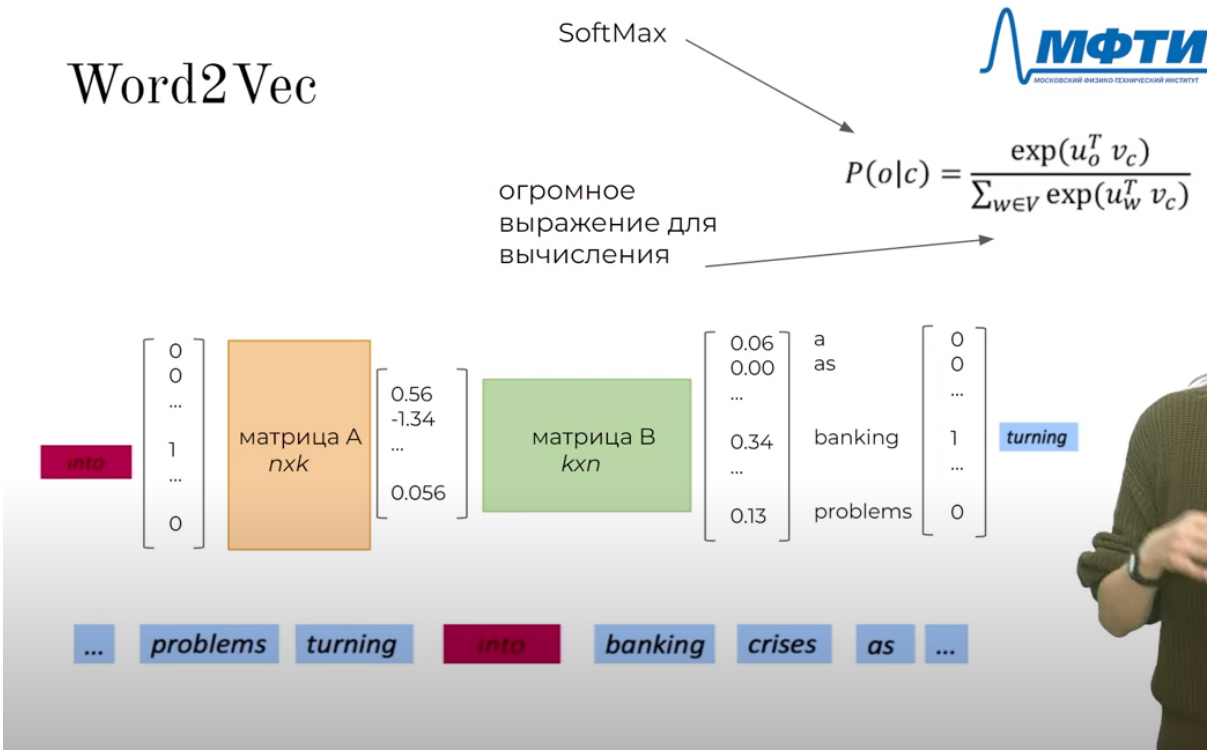
вектор слова номер j
как слова контекста

Word2Vec

После обучения сети мы получаем векторы размера k для всех слов в словаре.
Размер k мы можем задавать сами.

Эти векторы содержат смысл слов. Их можно сравнивать между собой с помощью косинусного расстояния.
Косинусное расстояние — это нормализованное скалярное произведение двух векторов.

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

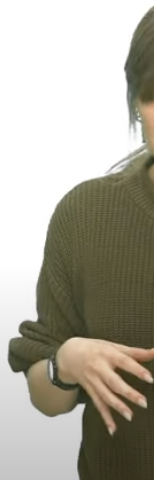


Word2Vec

SoftMax

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

огромное выражение для вычисления

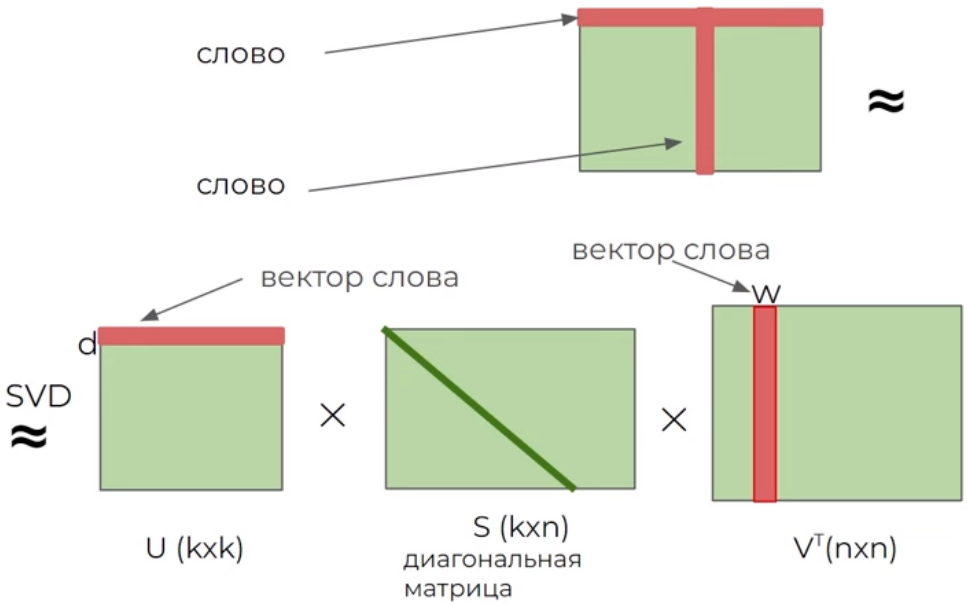


Идеи решения проблемы:

- Иерархический SoftMax;
- Negative sampling;

Word2Vec vs SVD

Word2vec с negative sampling получает эмбеддинги, похожие на эмбеддинги из SVD-разложения



FastText

Идея — будем строить векторы для частей слов, а не для целых слов.

- Делим слова на n-граммы по буквам:
  apple = <ap, ppl, ple, le>
- Учим векторы для n-грамм;
- Вектор слова получаем как сумму векторов его n-грамм.

Плюсы:

- Можно получить более адекватные эмбеддинги для редких и неизвестных слов;

Недостатки:

- n-грамм может быть очень много. Требуется больше вычислительных ресурсов.

GloVe (Global Vectors)

GloVe использует статистическую информацию о частоте встречаемости слов и фраз в тексте, чтобы улучшить обучение эмбеддингов редких слов.

Подробнее можно почитать тут:
https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010

Эмбеддинги предложений

Что мы делали со словами:

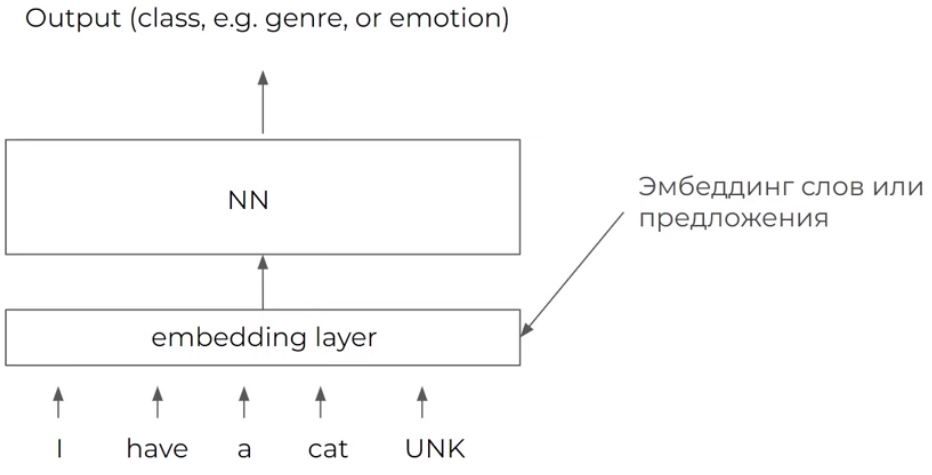
- Предсказывали следующее/предыдущее слово.

Что можно делать для предложений:

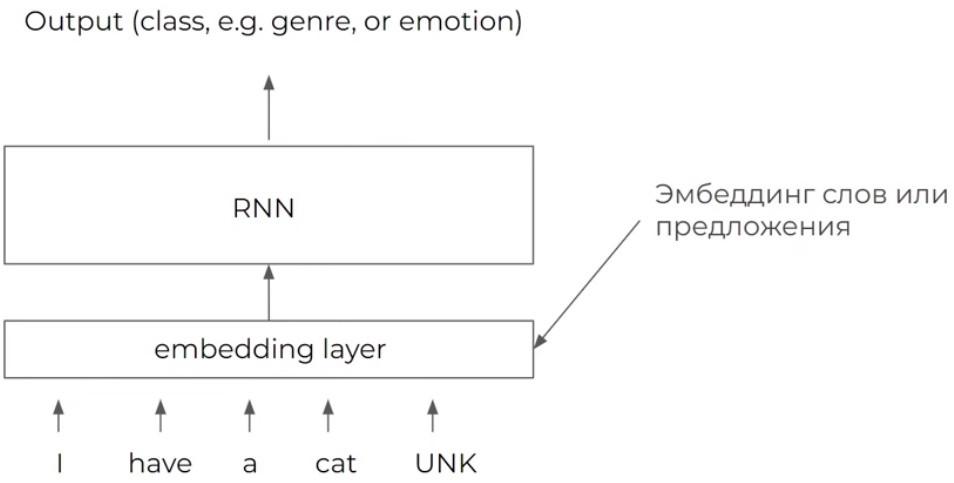
- Предсказывать порядок следования двух предложений (бинарная классификация);
- Может ли предложение A идти после предложения B? (бинарная или многоклассовая классификация)
- Предсказание соединительного слова между двумя предложениями.

Обучаясь на подобные задачи, нейросеть выучивает некую информацию о предложениях.

Классификация текста с помощью эмбеддингов



Классификация текста с помощью эмбеддингов

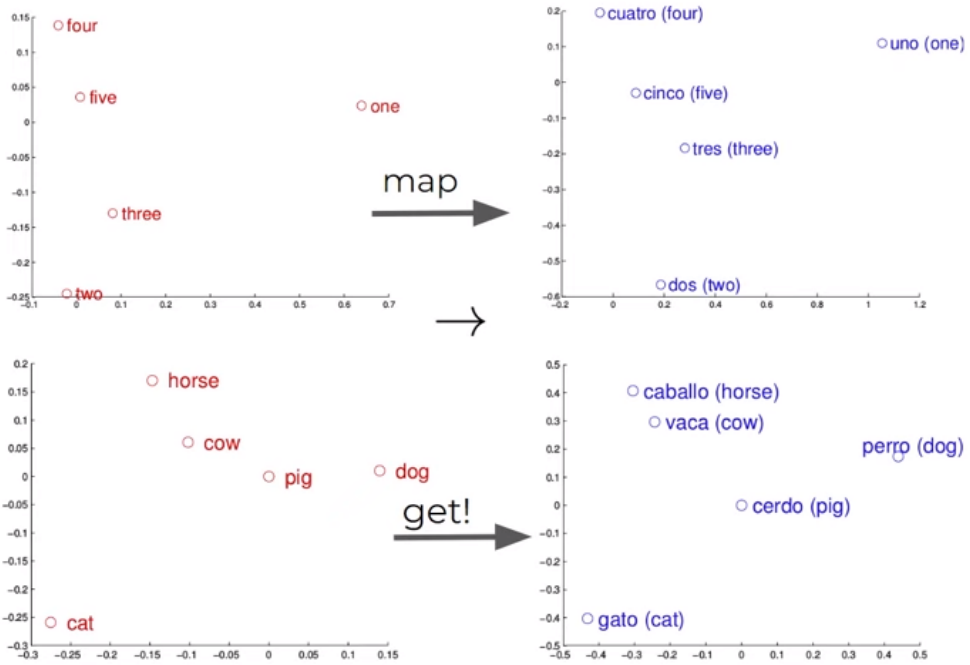


Карта языка

Пусть у нас есть тексты на неизвестном языке,
которые мы хотим научиться понимать.
Как это можно сделать:

- Обучаем эмбеддинги для слов английского языка;
- Обучаем эмбеддинги для неизвестного языка
- Находим преобразование f, которое переводит эмбеддинги английского языка в эмбеддинги неизвестного языка.

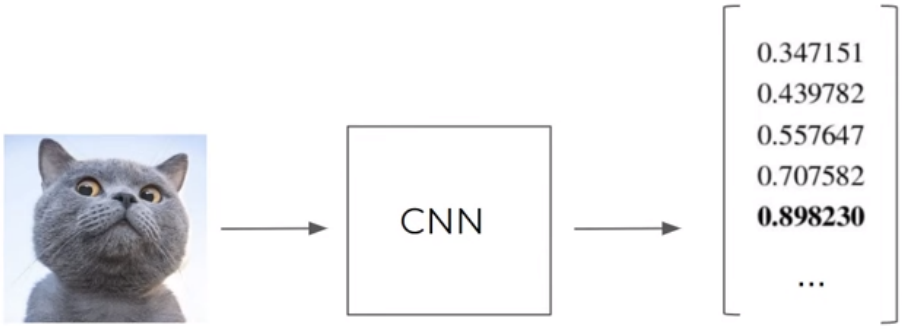
Карта языка


four
five
one
three
two
map
cuatro (four)
uno (one)
cinco (five)
tres (three)
dos (two)
horse
cow
pig
dog
cat
get!
caballo (horse)
vaca (cow)
perro (dog)
cerdo (pig)
gato (cat)

Эмбеддинги в общем смысле

Эмбеддинг — векторное представление объекта, которое отражает информацию об объекте.

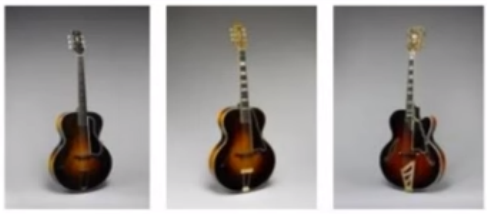
Выходы слоев моделей, обученных под какую-либо задачу, тоже можно считать эмбеддингами.



Эмбеддинги в общем смысле

Пример: поиск похожих изображений

- Берем предобученную VGG16 на ImageNet;
- Прогоняем все картинки датасета через модель, получаем выходы (эмбеддинги) предпоследнего слоя модели;
- Для конкретной картинки ищем похожие, сравнивая эмбеддинг этой картинки с эмбеддингами других картинок датасета по MSE/косинусному расстоянию.



```
import spacy

# load the English language model
nlp = spacy.load("en_core_web_sm")

# create a pipeline with a vectorizer component
nlp.add_pipe("vectorizer", last=True)

# create a list of documents
documents = [
    "This is the first document.",
    "This is the second second document.",
    "And the third one.",
    "Is this the first document?",
]

# process the documents with the pipeline
vectors = [doc.vector for doc in nlp.pipe(documents)]

# print the vectors
print(vectors)
```

To delete stop words, punctuation, and tokenize with spaCy, follow these steps:

1. Install spaCy if you haven't already
2. Download the language model for English

```
pip install spacy
python -m spacy download en_core_web_sm
```

3. Load the language model and create a function to process the text

```
import spacy
# Load the language model
nlp = spacy.load('en_core_web_sm')

def process_text(text):
  # Create a spaCy document
  doc = nlp(text)
  # Tokenize, remove stop words and punctuation
  tokens = [token for token in doc if not (token.is_stop or token.is_punct)]
  # Convert tokens back to text
  processed_text = " ".join([token.text for token in tokens])
  return processed_text
```

4. Use the process_text function to process your text

```
text = "This is an example sentence with stop words and punctuation!"
processed_text = process_text(text)
print(processed_text)
```

This will output

```
example sentence stop words punctuation
```

In this example, the stop words and punctuation are removed from the input text, and the remaining words are tokenized using spaCy stackoverflow.com.