1.Из заданного текста копируются слова целиком и кодируются в one-hot вектор.

Далее мы загружаем высказывания из файла **text** и разобьем их на слова при помощи tokenizer, также укажем количество прогназируемых слов maxwordscount = 1000 , если обучающая выборка больше то и количество слов надо указать больше.

И так мы разбили текст на слова теперь их токенизируем при помощи метода fit\_on\_text и получаем результат список, наиболее встречающихся, слов к которым привязан свой индекс

Далее мы преобразуем текст в последовательность чисел, для этого используем метод класса text\_to\_sequennces.

1.1) 1 пункт Inp\_words переменная, которую вы вычитаем из общего количества слов и она означает минимальное количество слов при вводе.

И в тоге наша обучающая выборка будет формироваться так:

X это картеж из 3 слова идущие друг за другом

Y это one-hot вектор 4го слова из текста

Так мы перебираем весь текст.

Далее мы создаем входной слой Embedding на 256 нейронов, который должен иметь тензор (input\_dim, output\_dim, input\_length)

Далее мы создаем рекуррентный слой

Далее мы созаем полносвязный слой

При компиляции качестве оптимизатора используется Adam [6], а в качестве функции ошибки - categorial crossentropy.

И остается лишь запустить процесс обучений

Далее идет функция которая строит фразу на основе прогнозных значений, которая формирует начальные слова в One-Hot\_encoding по которым будет делать прогноз и передает их обученной нейронной сети

1.2) все как а 1 пункте только мы будем работать с файлами **text\_true, text\_false**, а точнее словами которые из файла text были разделены на плохие и хорошие, если будет замечено незнакомое слово программа его проигнорирует.

X - надо выровнять высказывания из текста до размеров входного высказывания при помощи метода pad\_sequences, который коротким предложениям добавит 0, а длинные обрежет

Y – значение высказывание 0 отрицательное, 1 положительное, которые мы перемешаем для лучшей работы нейронной сети.

Далее тем же способ обучаем нейронную сеть что и в 1 пункте.

2.Нечеткая логика

1)Anfis

import numpy as np  
from pprint import pprint  
from fuzzy\_logic.anfis import Anfis  
  
x: np.ndarray = np.array([  
 [x1, x2, x3]  
 [x11, x22, x33]  
])  
  
y: np.ndarray = np.array([y1, y2, y3])  
  
anfis: Anfis = Anfis(x, y, 3)  
anfis.train()  
pprint(f'{anfis.calculate([x1, x22])} == {x1 \* x22}')

Обучение проводится путем задания значений x и y, например если x1 = 2, x11 = 3, y1= 5 то анфис научится складывать значения

2)sugeno

from pprint import pprint  
from fuzzy\_logic.terms import Term  
from fuzzy\_logic.variables import FuzzyVariable, SugenoVariable, LinearSugenoFunction  
from fuzzy\_logic.sugeno\_fs import SugenoFuzzySystem  
from fuzzy\_logic.mf import TriangularMF  
  
  
input1: FuzzyVariable = FuzzyVariable('input1', 0, 1,

Term('mf1', TriangularMF(-0.5, 0, 0.5)),  
 Term('mf2', TriangularMF(0, 0.5, 1)),  
 Term('mf3', TriangularMF(0.5, 1, 1.5))  
)  
input2: FuzzyVariable = FuzzyVariable(  
 'input2', 0, 1,  
 Term('mf1', TriangularMF(-0.5, 0, 0.5)),  
 Term('mf2', TriangularMF(0, 0.5, 1)),  
 Term('mf3', TriangularMF(0.5, 1, 1.5))  
)  
output: SugenoVariable = SugenoVariable(  
 'output',  
 LinearSugenoFunction('mf1', {input1: 0.1, input2: 0.3}, 0.5),  
 LinearSugenoFunction('mf2', {input1: 0.5, input2: 0.5}, 0.5),  
 LinearSugenoFunction('mf3', {input1: 1, input2: 0}, 1)  
)  
  
mf: SugenoFuzzySystem = SugenoFuzzySystem([input1, input2], [output])  
mf.rules.append(mf.parse\_rule('if (input1 is mf1) and (input2 is mf1) then (output is mf1)'))  
mf.rules.append(mf.parse\_rule('if (input1 is mf2) and (input2 is mf2) then (output is mf2)'))  
result = mf.calculate({input1: 0.4 , input2: 0.4})  
pprint(result)

3)mamdani

from fuzzy\_logic.terms import Term  
from fuzzy\_logic.variables import FuzzyVariable  
from fuzzy\_logic.mamdani\_fs import MamdaniFuzzySystem  
from fuzzy\_logic.mf import TriangularMF

t1 = Term('mf1', TriangularMF(-0.5, 0, 0.5))  
t2 = Term('mf2', TriangularMF(0, 0.5, 1))  
t3 = Term('mf3', TriangularMF(0.5, 1, 1.5))  
input: FuzzyVariable = FuzzyVariable('input1', 0, 1, t1, t2, t3)  
  
output = FuzzyVariable(  
 'output', 0, 1,  
 Term('mf1', TriangularMF(-0.5, 0, 0.5)),  
 Term('mf2', TriangularMF(0, 0.5, 1)),  
 Term('mf3', TriangularMF(0.5, 1, 1.5))  
)  
  
mf: MamdaniFuzzySystem = MamdaniFuzzySystem([input], [output])  
mf.rules.append(mf.parse\_rule('if (input1 is mf1) then (output is mf1)'))  
mf.rules.append(mf.parse\_rule('if (input1 is mf2) then (output is mf2)'))  
mf.rules.append(mf.parse\_rule('if (input1 is mf3) then (output is mf3)'))  
result = mf.calculate({input: 1]})

2 и 3 пунк работает довольно схоже в них мы устанавливаем промежуток в input случае от 0 до 1 а так же термы

Далее мы создаем output в котором так же настраиваем термы

Остается лишь задать значения правил нечеткого вывода

Так выгдядят термы в матлабе

