

НИЯУ “МИФИ”

НИРС

по теме: “Классификация медицинских текстов на наличие побочных эффектов при помощи методов машинного обучения”

Группа: М19-117

Студент: Кайгородов Александр

Преподаватели: Сбоев А.Г. , Рыбка Р.Б.

Москва, 2020

Введение	2
Данные для решения задачи	3
Опыт решения задач	5
Материалы	7
Предварительная обработка текста.	7
Подходы на базе методов машинного обучения	8
Подходы на основе нейронных сетей.	8
Метрика качества	9
Модели	10
Классические модели	10
Модели на основе нейронных сетей	11
LSTM	12
CNN	14
LSTM+CNN	15
Результаты	17
Настройка гиперпараметров в Hyperopt и Hyperas	18
CNN уровень символов + hyperas	18
CNN уровень символов + hyperopt	19
LSTM+CNN и hyperopt	20
Заключение	20

Введение

Тематика представленных наборов твитов склоняется к написанию отзыва о каком-либо медицинском препарате, побочные эффекты которого нужно занести в базу данных для фармакологических служб. Правильное извлечение огромного числа выявленных побочных эффектов может значительно улучшить разработку новых лекарств, но проблема заключается в работе со сложной структурой информации представленной в виде естественного языка.

Для решения данной задачи можно прибегать к использованию машинного обучения, в частности нейронных сетей. Использование классических базовых методов МЛ допустимо, но использование глубинного обучения представляется наиболее приоритетным в данном случае, т.к позволяет получать большую точность при предсказании.

Данная научно-исследовательская работа посвящена исследованию методов машинного обучения с сравнением эффективности работы как классических методов так и методов с использованием нейронных сетей для решения задач классификации текстов.

Для исследования методов поставлены следующие цели:

1. Добыть данные и провести их предобработку
2. Построить модели нейросетевых и классических подходов машинного обучения
3. Обучить модели
4. Получить точности работы моделей
5. Сравнить полученные точности и обозначить наилучшие модели

В качестве данных для исследования были взяты материалы открытого соревнования проводимого Пенсильванским медицинским институтом.

О соревновании

“Social Media Mining for Health Applications (#SMM4H) Shared Task 2020”

Задания данного соревнования включают в себя задачи тематики NLP, распространяющиеся на практические цели из сферы здравоохранения. Общая суть указанных заданий данного соревнования окружена вопросами извлечения информации из больших наборов твитов, написанных пользователями в сети.

Основная группа задач данного соревнования сводится к следующим пунктам.

1. Классификация твита на наличие или отсутствие побочного эффекта (ADR)
2. Выделение спана для конкретного побочного эффекта
3. Нормализация побочного эффекта (строгое формулирование)
4. Соотнесение полученного ADR с идентификатором базы данных MedDRA.

В данной работе для исследования выбрана задача из программы соревнования 2020 года:

(названия и описания тасков оставляются в оригинальном виде)

- Task 2: Automatic classification of multilingual tweets that report adverse effects
 - Training data: 25,672 tweets (2,374 “positive” tweets; 23,298 “negative” tweets)
 - Test data: ~5,000 tweets. (! в действительности полученных данных меньше)

Оригинальная ссылка на соревнование: [#SMM4H Shared Task 2020](#)

Данные для решения задачи

Данные, предоставленные организаторами соревнований, содержат посты пользователей твиттера - “твиты”, и разбиты на 2 датасета. Первый содержит N твитов и разметку твитов по классам:

- Есть побочный эффект действия препарата (ADR) указанного в твите
- Побочный эффект действия препарата отсутствует

Второй датасет содержит информацию о локальном присутствии ADR в приложенном твите. Используется термин SPAN, которые подразумевает границы написанного ADR-а в виде информации об индексе символов начала и конца описанного эффекта.

Пример твита из второго датасета:

@fibby1123 are you on paxil .. i need help - ADR отсутствует

@flatchests it has nothing to do with any of that~ apparently seroquel makes you gain alot of weight - ADR присутствует

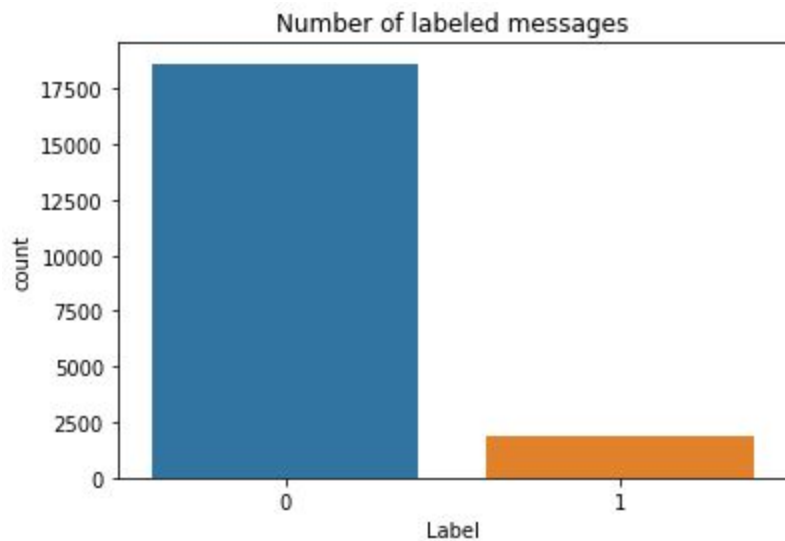
Пример твита из третьего датасета:

dis lady has the worst cough i wish i had a lozenge - ADR отсутствует

@luckystubbs reppin zoloft&seroquel since last november. i'm hella gainin weight too awesome i'm fat and can't cum i own - ADR присутствует (fat), спан = [101, 104]

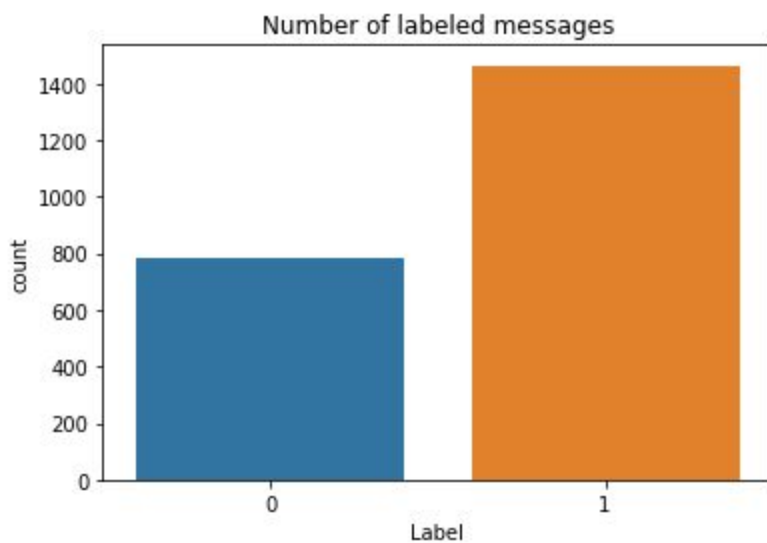
Факт дисбаланса данных является очень важным при обучении модели. Проблема существует и в одном из действительных датасетов.

Анализ баланса классов для второго датасета показал:



Во втором датасете присутствует дисбаланс (довольно выраженный) в сторону меток об отсутствии adr в твите. Этот факт важно это учитывается при оценке модели.

Анализ баланса классов для второго датасета показал:

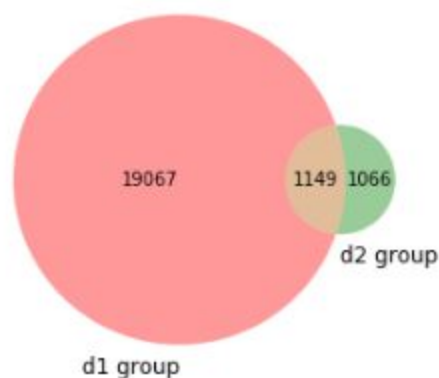
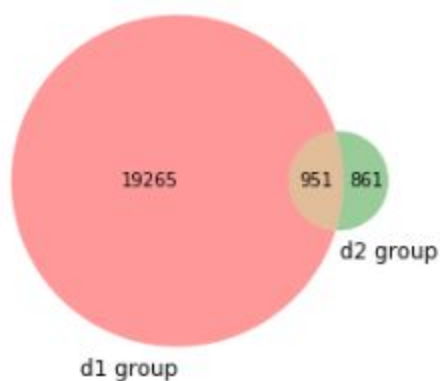


В дальнейших исследованиях, второй датасет будет использован в качестве дополнения ко второму.

Проведем анализ этих датасетов относительно друг друга.
(на первой картинке data2 and data3_train, на второй data2+(data3_train+data3_validation))

Size of d1: 20216
 Size of d2: 1812
 Not in d2: 19265
 Not in d1: 861
 Intercept: 951

Size of d1: 20216
 Size of d2: 2215
 Not in d2: 19067
 Not in d1: 1066
 Intercept: 1149



В итоге имеется следующее распределение:

Dataset	ADR	No ADR
Dataset 2	1903	18641
Dataset 3	1464	778

Опыт решения задач

Организаторы предоставили ссылку на сборник статей с воркшопа предыдущего года, на котором можно легко ознакомиться с работами других участников. Задания остались в той же тематике, но слегка изменили свои формулировки и все подходы и методы, описанные там не распространяются на текущее соревнование. Тем не мене, работы опубликованные в этом сборнике могут оказаться весьма полезными для ознакомления с работающими подходами и при оценке своей модели относительно полученных ранее. Это даст лучшее представление об эффективности проделанной работы.

LINK -> [WORKSHOP](https://www.aclweb.org/anthology/W19-32.pdf) (<https://www.aclweb.org/anthology/W19-32.pdf>)

Наибольший интерес представляют подходы победителей соревнований предыдущих лет. Я выбрал работу одного из участников <https://www.aclweb.org/anthology/W19-3207.pdf>

В работе говорится об эффективности применения классических методов при решении задачи классификации.

В отдельной части работы описывается подход при решении задачи классификации твитов обозначенной как TASK 1.

В качестве используемых методов выбраны:

- SVM (далее будут изложены подробности метода). Для представления слов использован метод представления мешка слов (BOW). Также был применен метод `sent2vec` для численного представления твитов.
- В качестве главного решения, в работе выдвигается использование классификатора на основе архитектуры BERT которая была получена при помощи архитектуры трансформера и логистической регрессии в качестве классификатора.

Для оценки результатов использования моделей использовались критерии качества: f1-measure (F1), precision (P), recall (R). Полученные результаты оценок представлены в таблице.

В работе приведены следующие оценки качества выполненной модели:

Run name	F1	P	R
KFU NLP, BERT	57.38	69.14	49.04
KFU NLP, SVM	51.64	56.2	47.76
Average scores	50.19	53.51	50.54

Table 1: Text classification results on the Task 1 test set.

Данные оценки взяты в качестве мерки сравнения полученной мной модели.

Материалы

Предварительная обработка текста.

На этапе подготовки твитов для использования в моделях проводилась процедура предобработки данных. В качестве основных подходов при токенизации и векторизации были применены следующие методы:

- TF-IDF
- FastText

TF-IDF (TF — term frequency, IDF — inverse document frequency) статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса. Вес некоторого слова пропорционален частоте употребления этого слова в документе и обратно пропорционален частоте употребления слова во всех документах коллекции.

FastText - библиотека разработанная компанией Facebook для получения эмбедингов слов. Классификатор FastText библиотеки gensim использует по умолчанию следующие параметры обработки: model: Архитектура CBOW, Skipgram. (CBOW по умолчанию).

По существу, FastText обрабатывает текст тем же подходом что и метод word2vec, но его ключевое отличие заключается в том что обработка уже идет на уровне символов. Это дает возможность использовать в качестве векторного представления слова которые ранее не оказывались в словаре.

Иллюстрация скользящего окна word2vec.

Thou shalt not make a machine in the likeness of a human mind

thou	shalt	not	make	a	machine	in	the	...
thou	shalt	not	make	a	machine	in	the	...
thou	shalt	not	make	a	machine	in	the	...
thou	shalt	not	make	a	machine	in	the	...
thou	shalt	not	make	a	machine	in	the	...

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine
a	not
a	make
a	machine
a	in
machine	make
machine	a
machine	in
machine	the
in	a
in	machine
in	this
in	likeness

Подходы на базе методов машинного обучения

Метод опорных векторов (SVM, support vector machine) — набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа. Принадлежит семейству линейных классификаторов. (https://en.wikipedia.org/wiki/Support_vector_machine)

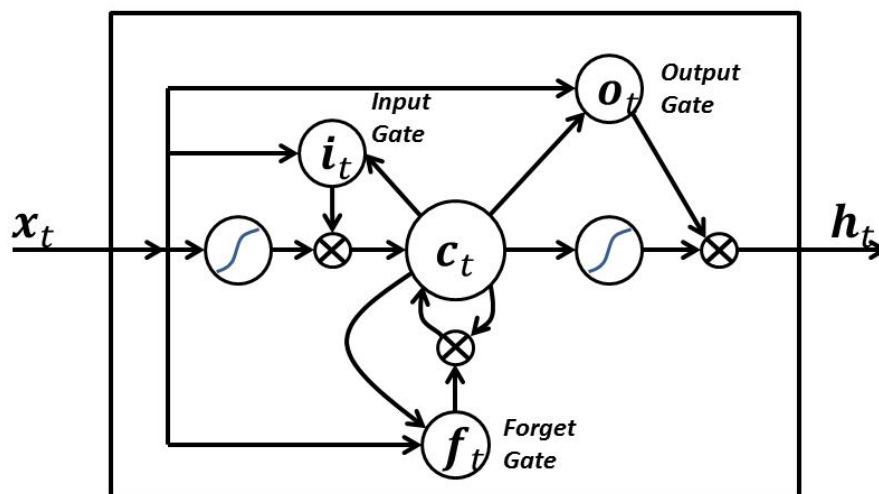
Многослойный перцептрон (MLP) - это класс искусственных нейронных сетей прямого распространения, состоящих как минимум из трех слоев: входного, скрытого и выходного. За исключением входных, все нейроны используют нелинейную функцию активации. (<https://wiki.loginom.ru/articles/multilayered-perceptron.html>)

Градиентный бустинг (GBM) — это техника машинного обучения для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений. (<https://neurohive.io/ru/osnovy-data-science/gradienty-busting/>)

TPOT - библиотека самостоятельно осуществляющая поиск подходящей модели с оптимизированными параметрами. Данный подход является очень универсальным и простым для программиста, но требует значительных вычислительных мощностей. Поиск решения осуществлялся для третьего датасета и сравнивался с наилучшей моделью обработавшей на втором датасете. Подробнее о TPOT <http://epistasislab.github.io/tpot/>

Подходы на основе нейронных сетей.

LSTM - это искусственная нейронная сеть, содержащая LSTM-модули вместо или в дополнение к другим сетевым модулям. LSTM-модуль — это рекуррентный модуль сети, способный запоминать значения как на короткие, так и на длинные промежутки времени. Ключом к данной возможности является то, что LSTM-модуль не использует функцию активации внутри своих рекуррентных компонентов. Позволяет выявлять временные зависимости.



CNN - сверточная нейронная сеть. Свое название получила из-за применения операции свертки. Широко используется в сфере CV где рабочее пространство представлено двумерными картинками, но также может быть применена в случае одномерной свертки.

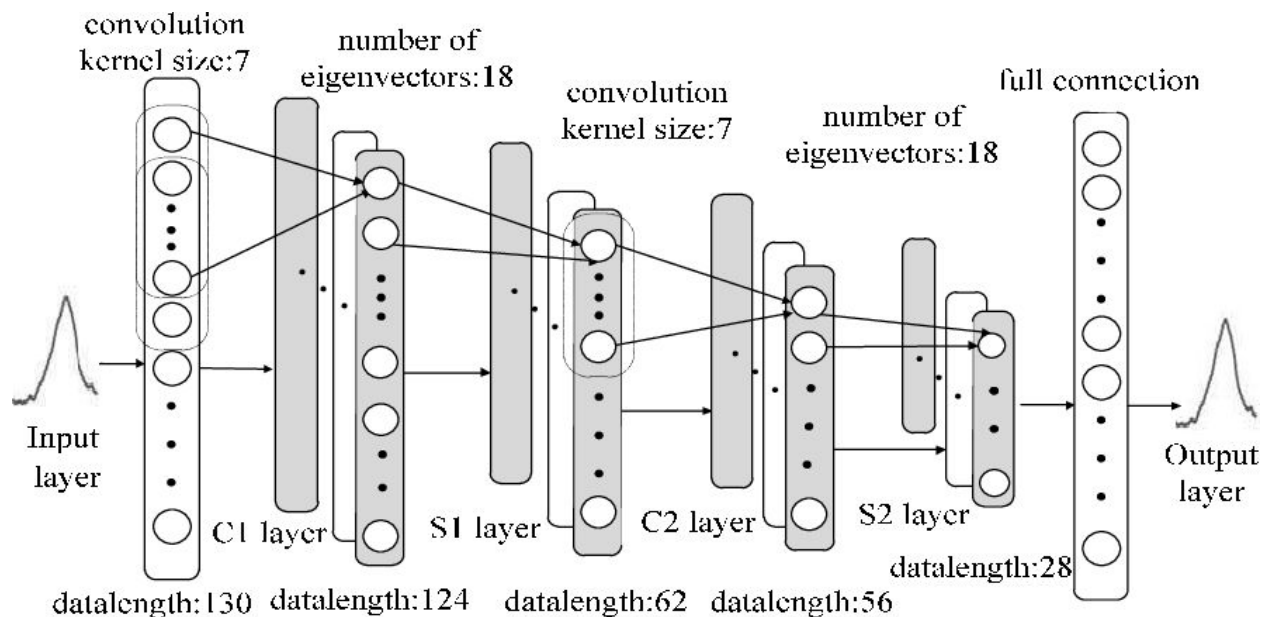


Рис. 4. Структура сверточной нейронной сети

Подробнее о технологии можно узнать тут:

https://en.wikipedia.org/wiki/Convolutional_neural_network

Метрика качества

В связи с тем что данные несбалансированные по классам, то оценка моделей по параметру ассигасу уже не будет являться объективной. В данном случае, для оценки качества модели применяется метрика f1-score (F-мера).

Данная оценка основана на вычислении двух следующих параметров:

- Точность (precision)
- Полнота (recall)

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

В полученной таблице содержится информация сколько раз система приняла верное и сколько раз неверное решение по документам заданного класса. А именно:

- TP - истинно-положительное решение;
- TN — истинно-отрицательное решение;
- FP — ложно-положительное решение;
- FN — ложно-отрицательное решение.

Категория i		Экспертная оценка	
		Положительная	Отрицательная
Оценка системы	Положительная	TP	FP
	Отрицательная	FN	TN

Модели

Классические модели

В качестве изучаемых классических моделей были построены следующие связки:

- TF-IDF + SVM
- TF-IDF + MLP
- TF-IDF + GMB
- FastText + SVM
- TPOT

Обучение описанных моделей проводилось на представленных датасетах. Топология применяемых методов и логика построения решения в целом представлена в приложениях к работе с кодом моделей.

Реализация представленных классических моделей исполнялась при использовании библиотеки scikit-learn.

При исследовании работоспособности классических моделей был применен метод кросс-валидации по фолдам. Данный подход позволяет получить более объективную оценку модели за счет использования имеющихся данных полностью. Полученные оценки моделей при таком подходе лучше отражают ее способность к решению задачи.

Во время исследований, среди используемых с TF-IDF моделей SVM, MLP и GBM было выявлено что в связке TFIDF + SVM лучшее сочетание качества и скорости обучения. Данная модель была применена при обучении на третьем датасете и сравнена с полученным результатом TPOT. Такое решение было принято в связи с слабыми вычислительными мощностями, которых не хватит для применения TPOT на втором датасете.

В результате исследования получены следующие значения качества используемых моделей.

MODEL	F1 score, %	Std
TFIDF + SVM	64.328	6.011
TFIDF + MLP	65.988	5.821
TFIDF + GBM	57.575	3.853
FastText + SVM	86.925	2.338
TFIDF + SVM (d3)	74.321	-
TFIDF + TPOT (d3)	75.177	-

Выводы: среди проведенных оценок, наилучшей на втором датасете оказалась модель с многослойным персептроном (MLP), но вычислительная способность такой модели не выглядит привлекательной, именно поэтому в качестве передовой модели мной была выбрана модель опорных векторов (SVM). Использование векторизации текста по методам FastText оказалось эффективней метода TFIDF.

Модели на основе нейронных сетей

В процессе подготовки текста к работе с сетью был использован набор эмбедингов "British National Corpus". Обработка текста происходила в следующем порядке. Токенизация -> формирование матрицы размерностью Число твитов * Макс длина предложения (наполнение паддингом) -> выравниваем эмбединга по количеству участвующих слов в вычислениях

При загрузке в модель, каждое слово будет встречать слой эмбединга которые паддингами будем извлекать вектора из представленного эмбединга для каждого слова и использовать в тренировке сети.

Тренировка и оценка моделей проходили на следующих размерах данных.

n_items_for_train = 500
n_items_for_test = 100
max_words = 2000

LSTM

В качестве принятой структуры нейронной сети использовались попытки реализовать два типа сетей, с классическим LSTM и двунаправленным LSTM (BI-LSTM).

- LSTM

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 36, 300)	600300
lstm_2 (LSTM)	(None, 128)	219648
dense_3 (Dense)	(None, 1)	129

Total params: 820,077
Trainable params: 219,777
Non-trainable params: 600,300

None

- двунаправленная LSTM

```
Model: "model_1"
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 36)	0
embedding_1 (Embedding)	(None, 36, 300)	600300
bidirectional_1 (Bidirection	(None, 128)	186880
dropout_1 (Dropout)	(None, 128)	0
batch_normalization_1 (Batch	(None, 128)	512
dense_1 (Dense)	(None, 32)	4128
dropout_2 (Dropout)	(None, 32)	0
batch_normalization_2 (Batch	(None, 32)	128
dense_2 (Dense)	(None, 1)	33

Total params: 791,981
Trainable params: 191,361
Non-trainable params: 600,620

lstm_10_10_0.21_0.18

Embedding может находится в состоянии Freez что означает что при тренировке сети, вектора слов изменяться не будут. В противном случае происходит дообучение уже имеющихся векторов представления слов.

В качестве попыток улучшения были применены следующие действия.

- Добавить второй слой эмбедингов которые уже будет обучаться
- Использовать batch_generator

В первой модификации использовались те-же значения размеров данных
Конфигурация модели:

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 36)	0	
input_2 (InputLayer)	(None, 36)	0	
embedding_1 (Embedding)	(None, 36, 300)	1200300	input_1[0][0]
embedding_2 (Embedding)	(None, 36, 300)	1200300	input_2[0][0]
bidirectional_1 (Bidirectional)	(None, 128)	186880	embedding_1[0][0] embedding_2[0][0]
concatenate_1 (Concatenate)	(None, 256)	0	bidirectional_1[0][0] bidirectional_1[1][0]
dropout_1 (Dropout)	(None, 256)	0	concatenate_1[0][0]
batch_normalization_1 (BatchNor	(None, 256)	1024	dropout_1[0][0]
dense_1 (Dense)	(None, 32)	8224	batch_normalization_1[0][0]
dropout_2 (Dropout)	(None, 32)	0	dense_1[0][0]
batch_normalization_2 (BatchNor	(None, 32)	128	dropout_2[0][0]
dense_2 (Dense)	(None, 1)	33	batch_normalization_2[0][0]
Total params: 2,596,889			
Trainable params: 1,396,013			
Non-trainable params: 1,200,876			
lstm_10_10_0.16_0.17			

В качестве улучшения двунаправленной модели LSTM был применен метод тренировки с использованием батч-генератора. Для этого создан отдельный класс генератора данных для тренировки и реализован через метод библиотеки keras fit_generator. Код модели и класса генератора в приложении.

CNN

В качестве реализации сверточной сети были опробованы два подхода.

- Сверточная сеть работающая на уровне символов
- Сверточная сеть работающая на уровне слов

При предобработке данных для сверточной сети на уровне символов была использована простейшая прямая токенизация по символам всех твитов предварительно обработанных и представленных через лемматизатор который был применен для LSTM.

Топология сверточной сети для уровня символов:

Model: "model_1"

Layer (type)	Output Shape	Param #
input (InputLayer)	(None, 200)	0
embedding_1 (Embedding)	(None, 200, 69)	4830
conv1d_1 (Conv1D)	(None, 194, 256)	123904
activation_1 (Activation)	(None, 194, 256)	0
max_pooling1d_1 (MaxPooling1	(None, 64, 256)	0
conv1d_2 (Conv1D)	(None, 58, 256)	459008
activation_2 (Activation)	(None, 58, 256)	0
max_pooling1d_2 (MaxPooling1	(None, 19, 256)	0
conv1d_3 (Conv1D)	(None, 13, 256)	459008
activation_3 (Activation)	(None, 13, 256)	0
conv1d_4 (Conv1D)	(None, 7, 256)	459008
activation_4 (Activation)	(None, 7, 256)	0
max_pooling1d_3 (MaxPooling1	(None, 2, 256)	0
flatten_1 (Flatten)	(None, 512)	0
dense_1 (Dense)	(None, 200)	102600
dropout_1 (Dropout)	(None, 200)	0
dense_2 (Dense)	(None, 200)	40200
dropout_2 (Dropout)	(None, 200)	0
dense_3 (Dense)	(None, 2)	402
Total params: 1,648,960		
Trainable params: 1,648,960		
Non-trainable params: 0		

Сверточная нейронная сеть на уровне слов использует ту же самую логику что и сверточная сеть на уровне символов, только на этот раз токенизация происходит на уровне слов по тому же самому обработанному набору.

Топология сверточной сети для уровня слов.

Model: "model_2"

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	(None, 36)	0
embedding_3 (Embedding)	(None, 36, 300)	1200300
conv1d_7 (Conv1D)	(None, 34, 300)	270300
dropout_8 (Dropout)	(None, 34, 300)	0
max_pooling1d_7 (MaxPooling1	(None, 17, 300)	0
conv1d_8 (Conv1D)	(None, 13, 300)	450300
dropout_9 (Dropout)	(None, 13, 300)	0
max_pooling1d_8 (MaxPooling1	(None, 6, 300)	0
conv1d_9 (Conv1D)	(None, 2, 300)	450300
dropout_10 (Dropout)	(None, 2, 300)	0
max_pooling1d_9 (MaxPooling1	(None, 2, 300)	0
flatten_3 (Flatten)	(None, 600)	0
dropout_11 (Dropout)	(None, 600)	0
batch_normalization_2 (Batch	(None, 600)	2400
dense_1 (Dense)	(None, 32)	19232
dropout_12 (Dropout)	(None, 32)	0
batch_normalization_3 (Batch	(None, 32)	128
dense_2 (Dense)	(None, 1)	33
Total params: 2,392,993		
Trainable params: 1,191,429		
Non-trainable params: 1,201,564		

CNN

LSTM+CNN

Совместное применение сверточной нейронной сети и сети LSTM. Предобработка данных осуществлялась аналогичным образом как и в предыдущих двух подпунктах.

Топология сети:

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
input_CHARS (InputLayer)	(None, 200)	0	
embedding_3 (Embedding)	(None, 200, 69)	4830	input_CHARS[0][0]
conv1d_1 (Conv1D)	(None, 194, 256)	123904	embedding_3[0][0]
activation_1 (Activation)	(None, 194, 256)	0	conv1d_1[0][0]
max_pooling1d_1 (MaxPooling1D)	(None, 64, 256)	0	activation_1[0][0]
conv1d_2 (Conv1D)	(None, 58, 256)	459008	max_pooling1d_1[0][0]
activation_2 (Activation)	(None, 58, 256)	0	conv1d_2[0][0]
max_pooling1d_2 (MaxPooling1D)	(None, 19, 256)	0	activation_2[0][0]
conv1d_3 (Conv1D)	(None, 13, 256)	459008	max_pooling1d_2[0][0]
activation_3 (Activation)	(None, 13, 256)	0	conv1d_3[0][0]
conv1d_4 (Conv1D)	(None, 7, 256)	459008	activation_3[0][0]
activation_4 (Activation)	(None, 7, 256)	0	conv1d_4[0][0]
max_pooling1d_3 (MaxPooling1D)	(None, 2, 256)	0	activation_4[0][0]
flatten_1 (Flatten)	(None, 512)	0	max_pooling1d_3[0][0]
dense_1 (Dense)	(None, 200)	102600	flatten_1[0][0]
input_1 (InputLayer)	(None, 36)	0	
input_2 (InputLayer)	(None, 36)	0	
dropout_1 (Dropout)	(None, 200)	0	dense_1[0][0]
embedding_1 (Embedding)	(None, 36, 300)	600300	input_1[0][0]
embedding_2 (Embedding)	(None, 36, 300)	600300	input_2[0][0]
dense_2 (Dense)	(None, 200)	40200	dropout_1[0][0]
bidirectional_1 (Bidirectional)	(None, 128)	186880	embedding_1[0][0] embedding_2[0][0]
dropout_2 (Dropout)	(None, 200)	0	dense_2[0][0]
concatenate_1 (Concatenate)	(None, 456)	0	bidirectional_1[0][0] bidirectional_1[1][0] dropout_2[0][0]
dropout_3 (Dropout)	(None, 456)	0	concatenate_1[0][0]
batch_normalization_1 (BatchNormalizatio	(None, 456)	1824	dropout_3[0][0]
dense_3 (Dense)	(None, 32)	14624	batch_normalization_1[0][0]
dropout_4 (Dropout)	(None, 32)	0	dense_3[0][0]
batch_normalization_2 (BatchNormalizatio	(None, 32)	128	dropout_4[0][0]
dense_4 (Dense)	(None, 1)	33	batch_normalization_2[0][0]
Total params: 3,052,647			
Trainable params: 2,451,371			
Non-trainable params: 601,276			

Результаты

В отличие от таблицы результатов представленной для результатов работы с классическими моделями, в данной таблице отсутствует показатель СКО для оценки, т.к. использование кросс-валидации по фолдам представляется очень затратной. Взамен представлены значения матриц сопряженности получаемых результатов (confusion matrix).

Данная матрица описана в пункте “Метрика качества” и продемонстрирована в конце.

LSTM or CNN	F1-score	Confusion matrix
LSTM Embedding (freez=True)	0.73	114/32 49/105
LSTM Embedding (freez=False)	0.74	111/39 39/111
LSTM Embedding (freez=False) + Embedding (freez=True)	0.73	116/37 43/104
CNN char_lvl	0.42	72/84 90/54
CNN word_lvl	0.71	100/55 31/114
LSTM + CNN	0.77 (0.79 extended data 1200\400)	99/53 16/132

Вывод: наилучший результат получен на модели с совместным использованием LSTM сети и сверточной сети. Данное явление можно описать увеличением емкости модели и как следствие улучшение ее обобщающей способности.

Настройка гиперпараметров в Hyperopt и Hyperas

Получение лучших гиперпараметров используемых в сети определяющих как ее топологию так и свойства слоев в ней является одной из самой важных и трудоемких задач при проектировании сети. Метод определения этих гипер параметров зачастую сводится к простому перебору наиболее вероятных значений которые способны улучшить способность сети к обобщению с наименьшей ошибкой.

Для более комфортного подбора параметров можно использовать вспомогательные библиотеки hyperopt и hyperas. Первая является базой разработки, вторая является модифицированным и упрощенным вариантом первой специально для работы вместе с библиотекой keras.

Данные библиотеки автоматизируют поиск наилучших параметров с контролем качества получаемых сетей. На выходе получаем перечень наиболее подходящих значений которые в дальнейшем следует использовать на полном наборе данных для получения лучшего качества сети.

Контрольные значения на данный момент.

Model	F1-score
Supported vector machine	0.69
2EMB + LSTM + CNN	0.80

CNN уровень символов + hyperas

Получены следующие оптимальные параметры

'Dense': 2, 'Dense_1': 2, 'Dropout': 0.6108763092812357, 'Dropout_1': 0.7371698374615214, 'activation': 0, 'activation_1': 0, 'batch_size': 1, 'optimizer': 2	'Dense': hp.choice('Dense', [256, 512, 1024]), 'activation': hp.choice('activation', ['relu', 'sigmoid']), 'Dropout': hp.uniform('Dropout', 0, 1), 'Dense_1': hp.choice('Dense_1', [256, 512, 1024]), 'activation_1': hp.choice('activation_1', ['relu', 'sigmoid']), 'Dropout_1': hp.uniform('Dropout_1', 0, 1), 'Optimizer': hp.choice('optimizer', ['rmsprop', 'adam', 'sgd']), 'Batch_size': hp.choice('batch_size', [32, 64, 128]),
--	---

После проверки полученных подстановкой в изначальную модель, точность классификации увеличилась на 2 процента.

CNN уровень символов + hyperopt

Расчет сильно усложнен требуемыми вычислениями, при минимальной обработке получены параметры ценность которых сомнительна, но доказывает стремление к оптимизации

('batch_size', 1) ('dropout1', 0.6334371211063357) ('dropout2', 0.36283182783153534) ('dropout3', 0.6297297987936828) ('dropout4', 0.6357503507027977) ('dropout5', 0.36563410317891387) ('hidden_activation', 2) ('layers', 2) ('loss', 0) ('optimizer', 1) ('pooling_size1', 0) ('pooling_size2', 0) ('pooling_size3', 1) ('pooling_size4', 0) ('pooling_size5', 1)	'choice': hp.choice('layers', [1, 2, 3, 4]), 'pooling_size1': hp.choice('pooling_size1', [3]), 'pooling_size2': hp.choice('pooling_size2', [3, -1]), 'pooling_size3': hp.choice('pooling_size3', [3, -1]), 'pooling_size4': hp.choice('pooling_size4', [3, -1]), 'pooling_size5': hp.choice('pooling_size5', [3, -1]), 'dropout1': hp.uniform('dropout1', .25,.75), 'dropout2': hp.uniform('dropout2', .25,.75), 'dropout3': hp.uniform('dropout3', .25,.75), 'dropout4': hp.uniform('dropout4', .25,.75), 'dropout5': hp.uniform('dropout5', .25,.75), 'batch_size' : hp.choice('batch_size', [32, 64]), 'hidden_activation': hp.choice('hidden_activation', ['relu', 'sigmoid', 'tanh', 'selu']), 'optimizer': hp.choice('optimizer',['adadelata', 'adam', 'rmsprop']), 'loss': hp.choice('loss', ['binary_crossentropy']), 'nb_epochs' : 15, 'activation': 'relu', 'patience': 10
---	---

После подстановки, точность модели не изменилась. Для получения изменений стоит применить большее число экспериментов, что требует больше вычислительных мощностей.

LSTM+CNN и hyperopt

Наиболее сложная реализация с вычислительной точки зрения

<pre>('batch_size', 1) ('dense_size', 0) ('dropout1', 0.39461651019515437) ('dropout2', 0.7396057604413669) ('dropout3', 0.4408370824285548) ('hidden_activation', 1) ('hidden_activation2', 1) ('loss', 0) ('optimizer', 2) ('pooling_size1', 0) ('pooling_size2', 0)</pre>	<pre>'pooling_size1': hp.choice('pooling_size1', [3, 5]), 'pooling_size2': hp.choice('pooling_size2', [3, 5]), 'dropout1': hp.uniform('dropout1', .25,.5), 'dropout2': hp.uniform('dropout2', .25,.5), 'dropout3': hp.uniform('dropout3', .25,.5), 'dense_size': hp.choice('dense_size', [32, 64, 128]), 'batch_size': hp.choice('batch_size', [32, 64]), 'hidden_activation': hp.choice('hidden_activation', ['relu', 'sigmoid', 'tanh', 'selu']), 'hidden_activation2': hp.choice('hidden_activation2', ['relu', 'sigmoid', 'tanh', 'selu']), 'optimizer': hp.choice('optimizer', ['adadelta', 'adam', 'rmsprop']), 'loss': hp.choice('loss', ['binary_crossentropy']), 'epochs': 15, 'activation': 'relu', 'patience': 5</pre>
--	---

Точность модели оказалась меньше. Решение может быть найдено увеличением числа экспериментов.

Заключение

По итогам проделанной работы , можно сделать несколько заключений, которые приведены в списке:

- Использование классических методов в задачах классификации текста возможно и имеет свои преимущества в виде более быстрых процессов обучения и детерминированности работы
- Для более качественных результатов рекомендуется применение технологий глубинного обучения
- Для применения глубинного обучения требуется большее количество исходных данных
- Для работы с глубинным обучением разумно использовать графические карты, в ином случае тренировка на обучающих данных может занять весьма продолжительное время
- Возможно совместное применение методов классического м.о. и глубинного.
- Предварительная обработка данных для обучений является крайне важным этапом, при ответственном подходе к которому можно улучшить качество полученного результата и ускорить обучение
- Возможно применение уже предобученных эмбеддингов

Лично для меня оказалось полезным более тесное ознакомление с применяемыми библиотеками, такими как scikit-learn, keras, tensorflow, pandas. Осуществлено практическое ознакомление с функциями и методами библиотек, были предприняты попытки модификации подходов. Изучены методы обработки естественного языка.