**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](#_f5rbemvn1m98) **2**

[Данные для решения задачи](#_st87o6j6x8yl) **3**

[Опыт решения задач](#_m3svg35s3j8l) **5**

[Материалы](#_jld9bvv7lazw) **7**

[Предварительная обработка текста.](#_8gcj93jyekr5) 7

[Подходы на базе методов машинного обучения](#_jnc8mxrlrr6t) 8

[Подходы на основе нейронных сетей.](#_p9i94a4zja6u) 8

[Метрика качества](#_bo2ro1u0rivn) 9

[Модели](#_xede9qj76k7) **10**

[Классические модели](#_p3bzlpoqbhkd) 10

[Модели на основе нейронных сетей](#_rpig75v5oup6) 11

[LSTM](#_554nko4hjmkh) 12

[CNN](#_987p1xlr6fm8) 14

[LSTM+CNN](#_7fhx3llkmin7) 15

[Результаты](#_56s3mq95fgg9) 17

[Настройка гиперпараметров в Hyperopt и Hyperas](#_nlvwbx6tr97) **18**

[CNN уровень символов + hyperas](#_b3hdupq2sfak) 18

[CNN уровень символов + hyperopt](#_etw80loznqdt) 18

[LSTM+CNN и hyperopt](#_ebchz9apei12) 19

[**Заключение**](#_88yzrdxxasts) **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: 20,216 tweets (1,903 “positive” tweets; 18,641 “negative” tweets)
* Task 3: Automatic extraction and normalization of adverse effects in English tweets
  + 1,812 tweets (1,464 “positive” tweets; 778 “negative” tweets)

Оригинальная ссылка на соревнование:  [#SMM4H Shared Task 2020](https://healthlanguageprocessing.org/smm4h-sharedtask-2020/) (https://healthlanguageprocessing.org/smm4h-sharedtask-2020/)

# **Данные для решения задачи**

Данные, предоставленные организаторами соревнований, содержат посты пользователей твиттера - “твиты”, и выделены в два различных датасета для каждой задачи.

Датасет для task2 содержит N твитов и разметку твитов по классам:

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

Датасет для task3 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. Также в датасете для task3 присутствуют твиты не имеющие ADR.

Пример твита из датасета для task2:

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

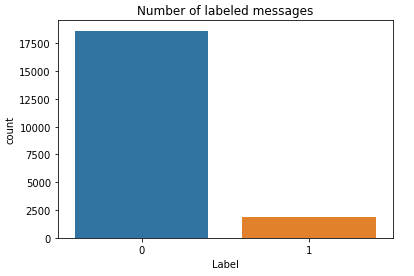
|  |
| --- |
| @flatchests it has nothing to do with any of that~ apparently seroquel makes you gain alot of weight - ADR присутствует |

Пример твита из третьего датасета для task3:

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

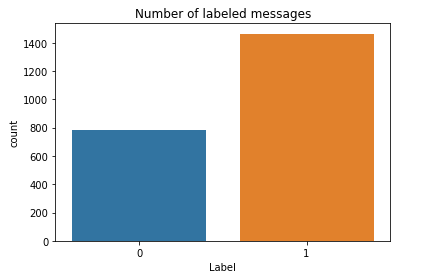
|  |
| --- |
| @luckystubbs reppin zoloft&amp;seroquel since last november. i'm hella gainin weight too awesome i'm fat and can't cum i own - ADR присутствует (fat), спан = [101, 104] |

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

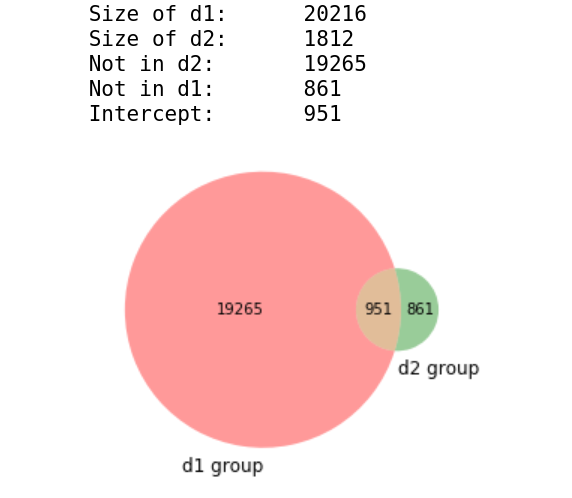
Анализ баланса классов для датасета task2 показал:

Во датасете task3 присутствует дисбаланс (довольно выраженный) в сторону меток об отсутствии adr в твите. Этот факт важно это учитывается при оценке модели.

Анализ баланса классов для датасета task3 показал:



Проведем анализ этих датасетов относительно друг друга. При сравнении используется диаграмма Венна между тренировочными примерами данных task2 и task3.



В итоге имеется следующее распределение:

|  |  |  |
| --- | --- | --- |
| Dataset | ADR | No ADR |
| Dataset task\_2 | 1903 | 18641 |
| Dataset task\_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 свой 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

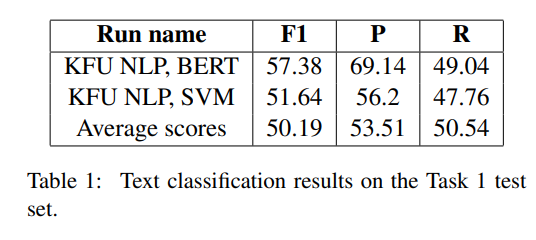
В работе говорится об эффективности применения классических методов при решении задачи классификации.

В отдельной части работы описывается подход при решении задачи классификации твитов обозначенной как Task 1. В действительности, набор данных схож по составу набора данных для task2.

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:

Данные оценки взяты в качестве мерки сравнения полученной мной модели. Данные результаты были похожи на схожих данных, но имеющих меньший объем. Следовательно данные значения не подвергаются прямому сравнению, а представляются как “отсечка”.

# **Материалы**

## **Предварительная обработка текста.**

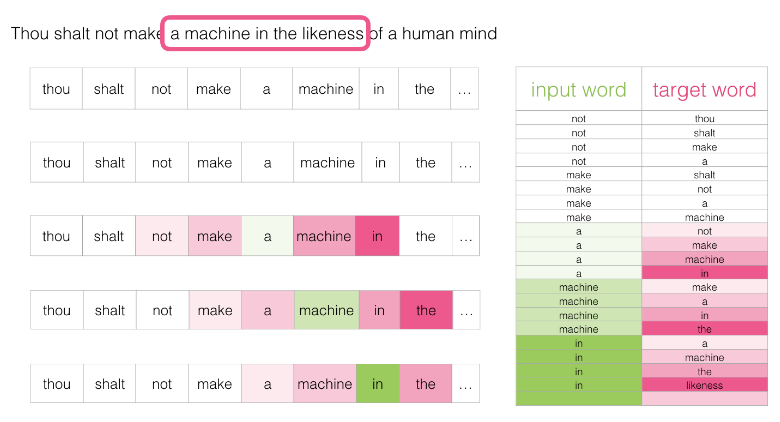
На этапе подготовки твитов для использования в моделях проводилась процедура предобработки данных. В качестве основных подходов при токенизации и векторизации были применены следующие методы:

* 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 эмбеддингов слов. Классификатор FastText библиотеки gensim использует по умолчанию следующие параметры обработки: model: Архитектура CBOW, Skipgram. (CBOW по умолчанию).

По существу, FastText обрабатывает текст там 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.

## **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>)

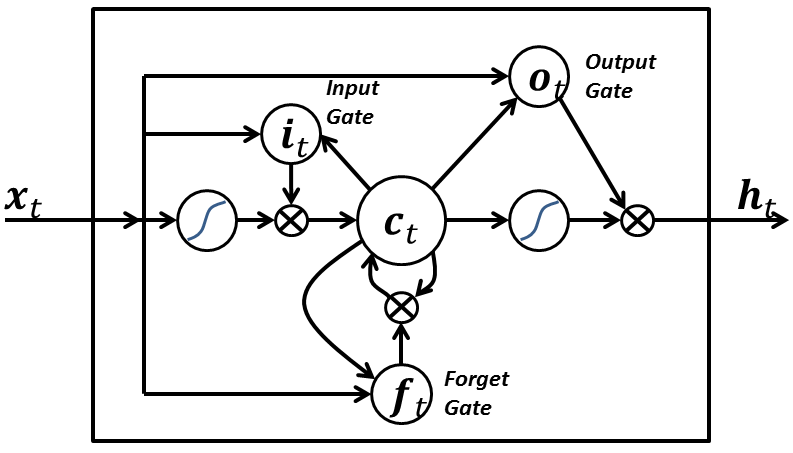
Многослойный перцептрон (MLP) - это класс искусственных нейронных сетей прямого распространения, состоящих как минимум из трех слоев: входного, скрытого и выходного. За исключением входных, все нейроны используют нелинейную функцию активации. (<https://wiki.loginom.ru/articles/multilayered-perceptron.html>)

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/gradientyj-busting/>)

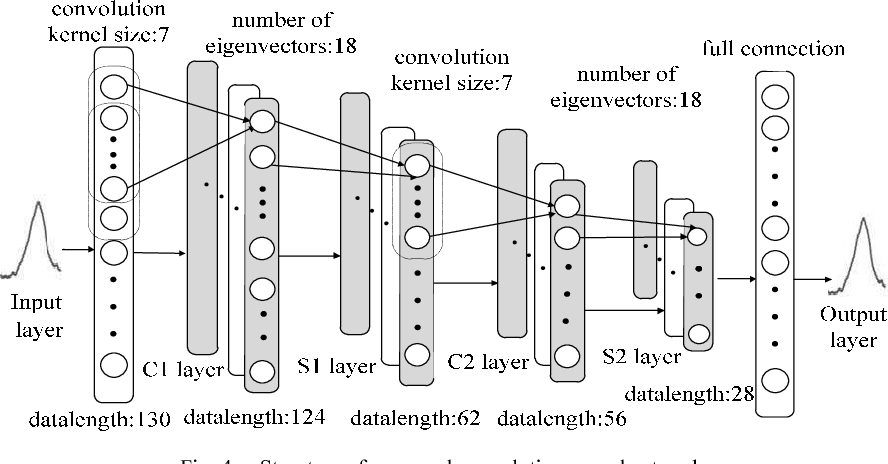
TPOT - библиотека осуществляющая самостоятельный поиск подходящей модели с оптимизированными параметрами. Данных подход является очень универсальным и простым для программиста, но требует значительных вычислительных мощностей. Поиск решения осуществлялся для третьего датасета и сравнивался с наилучшей моделью отработавшей на втором датасете. Подробней о ТРОТ <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 двухмерными картинками, но также может быть применена в случае одномерной свертки.



Подробней о технологии можно узнать тут:

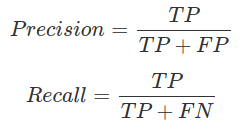
<https://en.wikipedia.org/wiki/Convolutional_neural_network>

## **Метрика качества**

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

Данная оценка основана на вычислении двух следующих параметров:

* Точность (precision)
* Полнота (recall)



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.



# **Модели**

## **Классические модели**

В качестве изучаемых классических моделей были построены следующие связки:

* TF-IDF + SVM
* TF-IDF + MLP
* TF-IDF + GMB
* FastText + SVM
* ТРОТ

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 ТРОТ. Такое решение было принято в связи с слабыми вычислительными мощностями, которых не хватит для применения ТРОТ 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 векторизацией через модуль FastText и моделью SVM, но вычислительная скорость такой модели не оказалась привлекательной, именно поэтому в качестве центральной классической модели мной была выбрана модель опорных векторов (SVM) с предварительной обработкой при помощи подхода TF-IDF.

## **Модели на основе нейронных сетей**

В процессе подготовки текста к работе с сетью был использован набор эмбеддингов “British National Corpus”. Обработка текста происходила в следующем порядке. Токенизация -> формирование матрицы размерностью   
*Число\_твитов \* Макс\_длина\_предложения* (наполнение паддингом) *->* выравнивае эмбеддинга по количеству участвующих слов в вычислениях

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

Тренировка и оценка моделей проходили на следующих размерах данных.  
n\_items\_for\_train = 500

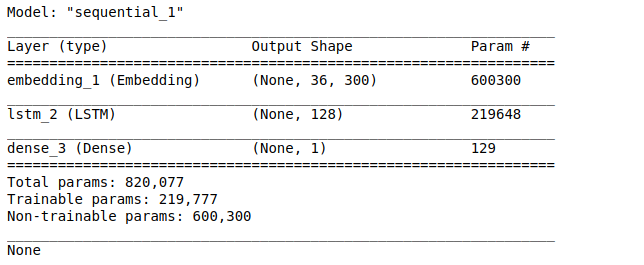
n\_items\_for\_test = 100

max\_words = 2000

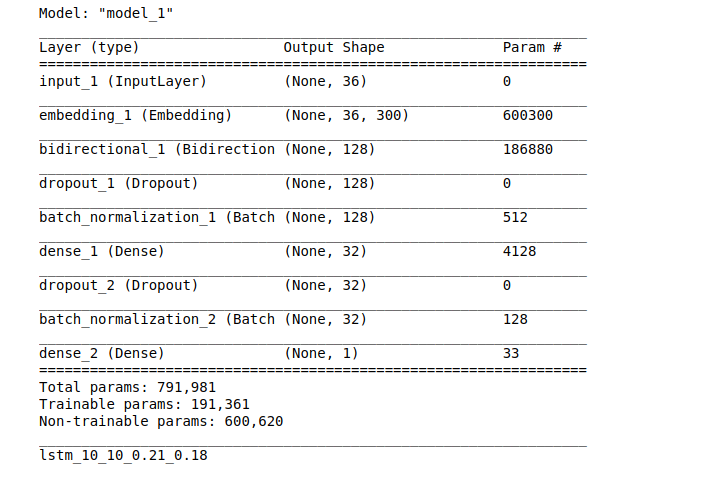
### **LSTM**

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

* LSTM



* двунаправленная LSTM



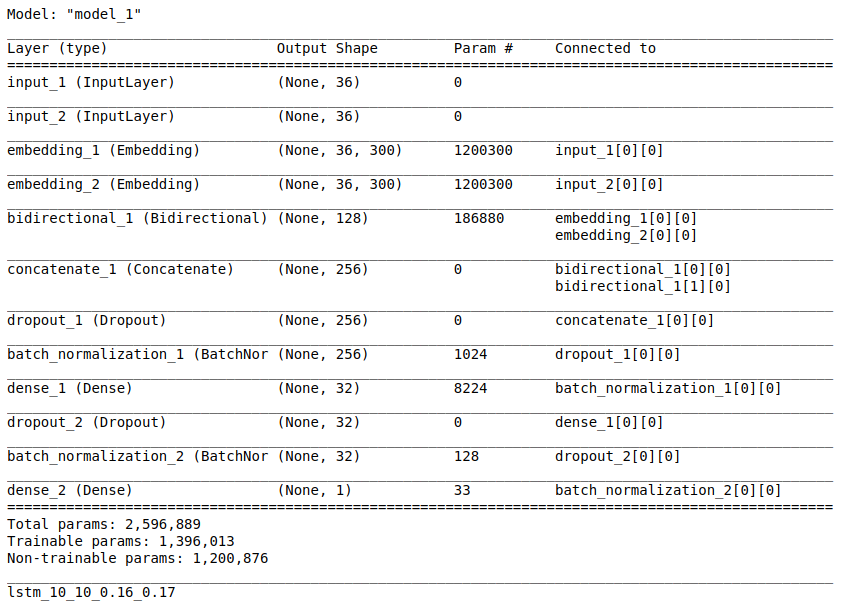
Embedding может находится в состоянии Freez что означает что при тренировке сети, вектора слов изменяться не будут. В противном случае происходит дообучение уже имеющихся векторов представления слов.

В качестве попыток улучшения были применены следующие действия.

* Добавить второй слой эмбеддингов которые уже будет обучаться
* Использовать batch\_generator

В первой модификации использовались те-же значения размеров данных

Конфигурация модели:



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.

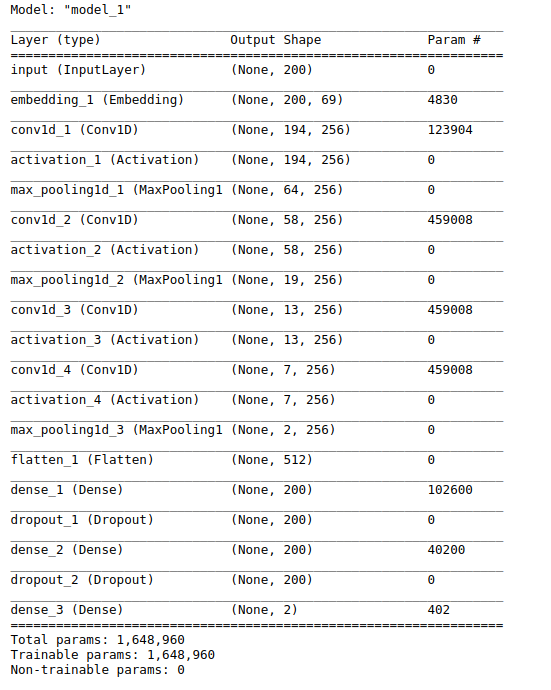
### **CNN**

В качестве реализации сверточной сети были опробованы два подхода.

* Сверточная сеть работающая на уровне символов
* Сверточная сеть работающая на уровне слов

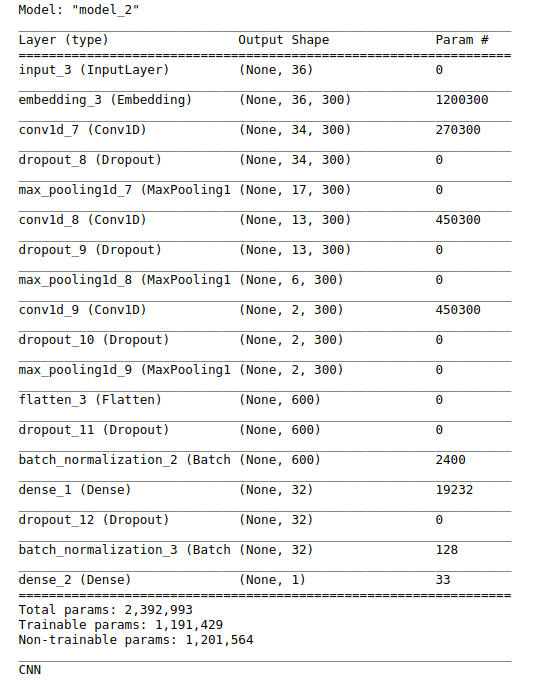
При предобработке данных для сверточной сети на уровне символов была использована простейшая прямая токенизация по символам всех твитов предварительно обработанных и представленных через лемматизатор который был применен для LSTM.

Топология сверточной сети для уровня символов:



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.

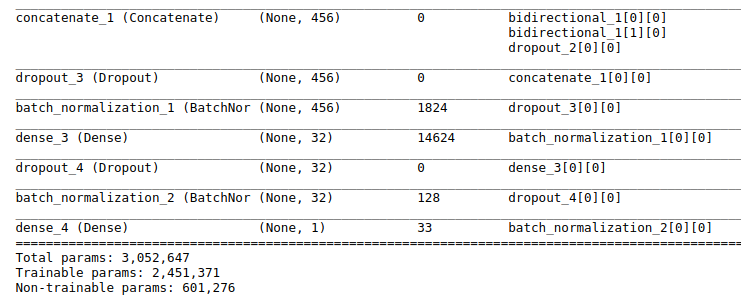
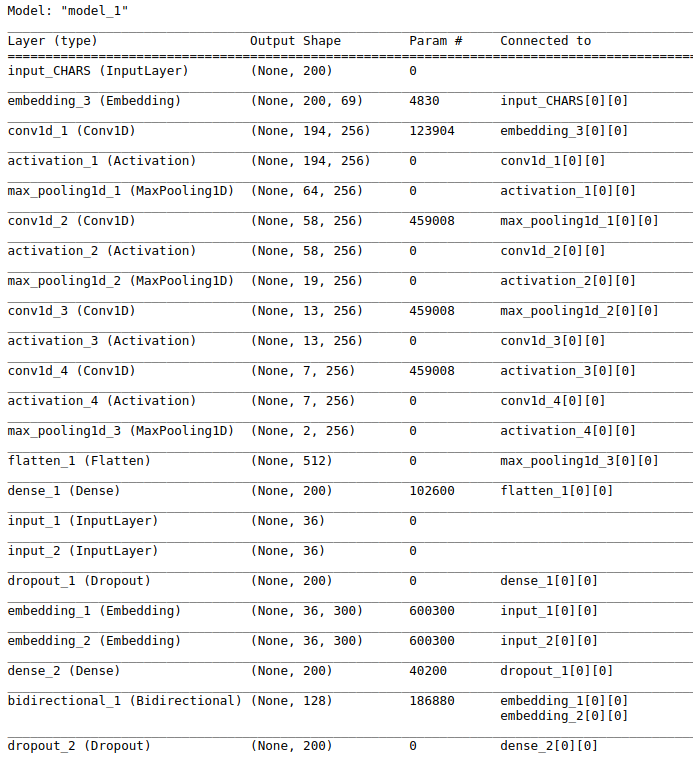
Топология сверточной сети для уровня слов.



### **LSTM+CNN**

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

Топология сети:



### **Результаты**

В отличии 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).

Для сравнения показания классической модели был взят результат работы TF-IDF на том же самом, но уже урезанном наборе данных датасет task2. Качество такой модели получилась ожидаемо меньше чем на полном наборе и теперь равняется 0.48.

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 |
| TF-IDF + SVM | 0.48 | - |

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.

В дальнейшем, при увеличении объема тренировочных и тестировочных данных, мной ожидается увеличение качества предсказаний данной модели которое, вероятно, окажется лучше чем у модели FastText + SVM.

# **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.

## **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',['adadelta', '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

|  |  |
| --- | --- |
| ('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) | '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 |

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.