Санкт–Петербургский государственный университет

ШЕРГИН Тимур Олегович

Выпускная квалификационная работа

Методы детектирования искусственных новостей

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2016 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Информационные системы»

Научный руководитель:
доцент, кафедра технологии программирования,
к.т.н. Блеканов Иван Станиславович

Рецензент:
доцент, кафедра информационных систем,
д.ф. - м.н. Матросов Александр Васильевич

Санкт-Петербург
2020 г.

Содержание

Введение 3

Постановка задачи 4

Глава 1. Обзор существующих решений 5

Глава 2. Реализация методов 6

2.1. Датасет[10] 6

2.2. Предварительная обработка данных 6

2.2.1 Удаление стоп-слов 6

2.2.2 Удаление чисел 7

2.2.3 Удаление иноязычных слов 7

2.2.4 Удаление знаков препинания и специальных символов . 7

2.3. Распределение данных 8

2.3.1 Графики полярности настроения 8

2.3.2 Графики речевого распределения 9

2.3.3 Униграмма и биграмма 10

2.4. Pre-training models 11

2.4.1 Word2Vec 11

2.4.2 TF-IDF Vectoraizer 13

2.5. Classic machine learning based methods 14

2.5.1 SVM 14

2.5.2 Random Forest 16

2.6. Deep learning based methods 18

2.6.1 LSTMs (Long Short Term Memory networks) 18

2.6.2 Трансформеры 20

2.6.3 BERT 22

2.6.4 XLNet 24

Глава 3. Анализ результатов 26

Вывод 30

Список литературы 31

Введение

Появление новых информационных технологий привело к появлению множества новых способов потребления, таких как онлайн-покупки, мультимедийные развлечения, игры, реклама или обучение. В последние годы произошел отток пользователей из более традиционных средств массовой информации, таких как газеты, радио и телевидение, в новые форматы: социальные сети, YouTube, подкасты, онлайн-журналы, новостные приложения и т. д.. Основной причиной распада информационных средств является растущая благодаря Интернету возможность мгновенного и бесплатного доступа к широкому спектру источников информации, не говоря уже о многочисленных услугах, позволяющих делиться новостями с миллионами людей по всему миру. В результате СМИ начали реагировать на изменения. Некоторые, например, начали расставлять приоритеты своего присутствия в Интернете или решили начать использовать новые каналы распространения, такие как видео или подкасты. Большинство этих средств массовой информации решили начать монетизацию своего контента с помощью рекламы, встроенной в их статьи, видео и т. д. Одним из наиболее частых методов является публикация статей с яркими заголовками и фотографиями, предназначенными для использования в социальных сетях (так называемые кликбейты), чтобы пользователи переходили на их сайты, таким образом максимизируя доходы таких СМИ. Однако такой подход может привести к опасным ситуациям. Большой объем информации, к которой люди получают доступ, обычно не проверяется и обычно считается достоверным. Именно в этот момент возникает термин поддельные новости.

В наши дни методы детектирования искусственных новостей развиваются быстро, но это все еще очень сложная проблема, требующая дальнейшего изучения.

Постановка задачи

В данной работе я сосредоточился на анализе содержания новостей, поэтому основная цель данной работы найти связь в словах и контекст, в котором слова появляются в тексте, и как он может быть использован для классификации текстов новостей. Для достижения цели были поставлены следующие задачи:

- Провести обзор существующих решений
- Реализовать методы основанные на классических алгоритмах машинного обучения (SVM, Random Forest)
- Реализовать методы основанные на глубоком обучении (LSTM, BERT, XLNet)
- Проанализировать полученные результаты

Глава 1. Обзор существующих решений

Исследования в области обнаружения поддельных новостей были интенсивными в последние годы. Тем не менее, большая часть работ в этой области сосредоточена на идее изучения и обнаружения мистификаций в ее основном канале распространения: социальных сетях. Примерами этого являются [1] или [2], где вероятность того, что данный пост будет ложным, изучается с использованием его собственных характеристик, таких как «лайки», «подписчики», «поделиться» и т. д., с помощью классических методов машинного обучения (деревья классификации, SVM, и т. д.). Применяя такого рода приближения, наилучшие результаты получены в [3], где обнаруживаются click-bait новости, достигающие результатов с точностью 93В других работах, таких как [4], используются аппроксимации на основе графов для изучения отношений между пользователями, которые делятся новостями, и путей, которыми следует контент, для того чтобы остановить его распространение и смягчить его потенциально обманное воздействие. Хотя общая тенденция состоит в том, чтобы проанализировать способы распространения мистификаций, уже начали появляться другие альтернативы, сосредоточенные на анализе содержания новостей. Так в статье [5], помимо пользовательских функций, которые делятся новостями, используется текст новости для распознавания поддельных новостей, аналогично статье [6], где новости изучаются с целью выявления ложных фактов в содержании. Что касается использования современных алгоритмов глубокого обучения, компания Fabula.ai использует метод, который учитывает как содержание новостей, так и функции, извлеченные из социальных сетей, достигая результатов 0,93 AUC [7]. В [8] сравнивается эффективность нескольких алгоритмов (как классического, так и глубокого обучения), при котором новости классифицируются по категориям «истинный» и «поддельный», что дает результаты с 95Наконец, используя только содержание новостей [9], предлагается метод на основе сверточной нейронной сети для обнаружения поддельных новостей с использованием заголовков и изображения заголовка, получая результаты с точностью 92

Глава 2. Реализация методов

2.1 Датасет[10]

Набор обучающих данных имеет пять функций: идентификатор, заголовок, автор, текст и метка. Идентификатор однозначно идентифицXLNetирует новостную статью. Заголовок и автор - это соответственно заголовок и автор новостной статьи. Текст является содержанием статьи. Метка указывает является ли статья надежной (реальной) или нет:

$$f(x) = \begin{cases} 0, & \text{если новость реальная} \\ 1, & \text{если новость фейковая} \end{cases}$$

Набор обучающих данных содержит 20800 новостных статей.

2.2 Предварительная обработка данных

2.2.1 Удаление стоп-слов

Стоп-слова - это список наиболее распространенных слов в языке (в нашем случае английском “a”, “an”, “be”, “the”, . . .), которые часто лишены смысла, но не всегда. Иногда информация, которую мы ищем, может быть включена в стоп-слова, которые мы удалили. Например в большинстве случаев языкового перевода, где важно, чтобы мы оставляли все стоп-слова. Однако в данной работе я использую семантику текста для принятия решений, следовательно я предполагаю, что удаление этих слов может дать мне несколько преимуществ:

- Уменьшение шума, поскольку, удаляя стоп слова, мы можем сосредоточится на более значимом содержании.
- Уменьшает количество функций для обучения моделей, так как сокращается огромное количество текста.

2.2.2 Удаление чисел

В контексте заголовка и текста новостной статьи числа просто не несут большого количества информации, то есть смысл текста не изменится при их удалении. Поэтому лучше убрать все числа, чтобы минимизировать шум в наших данных. Для этого используем строковую константу string.digits («0123456789») в Python, а также методы из строкового модуля Python str.maketrans(in
outtab) (возвращает таблицу перевода, которая отображает каждый символ в строке intab на символ в той же позиции в строке outtab. Затем эта таблица передается в функцию str.translate) и соответственно str.translate(table[, deletechars]) (возвращает копию строки, в которую все символы были переведены с использованием таблицы созданной методом maketrans при желании удаляя все символы, найденные в строке deletechars), для эффективного преобразования всех цифр в пустую строку.

2.2.3 Удаление иноязычных слов

Далее появилась необходимость убрать все иноязычные слова, так как в данной работе рассматривается задача выявления фейковых новостей только на английском языке. Для этого используется пакет langid в Python для определения языка всех текстов и последующее удаление всех строк с иностранными символами.

2.2.4 Удаление знаков препинания и специальных символов

Для этого использую модуль string.punctuation
(!"#$% &’()*+,-./:;<=>?@[\]^_’{|}~«») в Python, чтобы найти все знаки препинания и удалить все эти знаки из каждого слова в текстах, за исключением символов "#"и "@". Потому что эти символы используются для хэштегов и упоминаний в Твиттере и часто добавляются, чтобы получить больше результатов поиска и актуальности, но часто отвлекают от общего смысла самого новостного контента. Так как в задаче я прежде всего заинтересован в словах и в их основном контекстуальном значении, использующемся в тексте, то предполагаю, что это ненужные символы. Чтобы обнаружить хэштеги и упо-

минания, просто используются регулярные выражения, чтобы удалить весь текст после хэштег ("#") или символ ("@"), и прекратить удаление текста, когда мы достигнем следующего пробела. Также регулярные выражения используются для обработки тире ("—"), потому что оно используется в различных лингвистических конструкциях, таких как объединение независимых предложений, поэтому заменяем его на пробел. После чего возникает вероятность того, что в нашем тексте появились два или более пробела подряд, их мы также заменяем на одиночный пробел.

2.3 Распределение данных

После предварительной обработки было проанализировано и представлено распределение данных (текст) в нескольких различных аспектах. Я проанализировал данные, построив график их полярности настроения, наиболее популярных униграмм и биграмм, а также рассмотрев распределение типов слов.

2.3.1 Графики полярности настроения

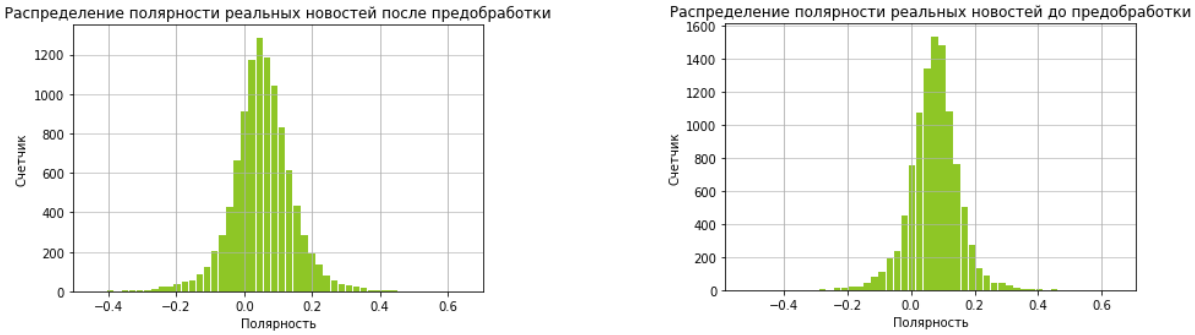


Рис. 1: Полярность реальных новостей

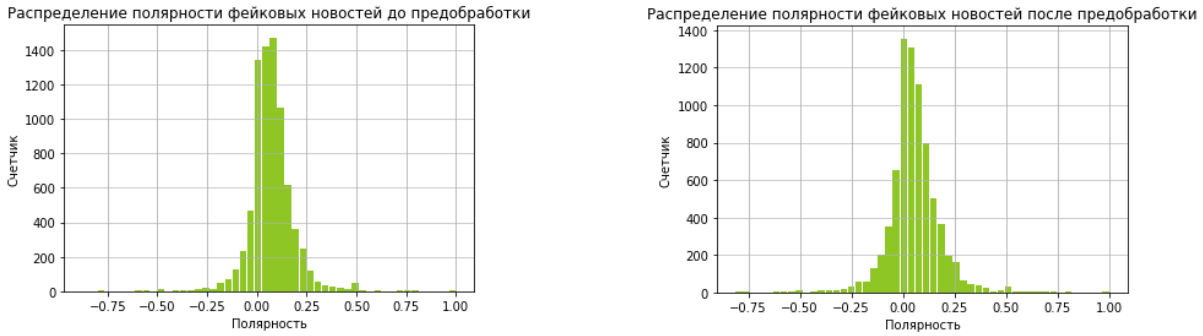
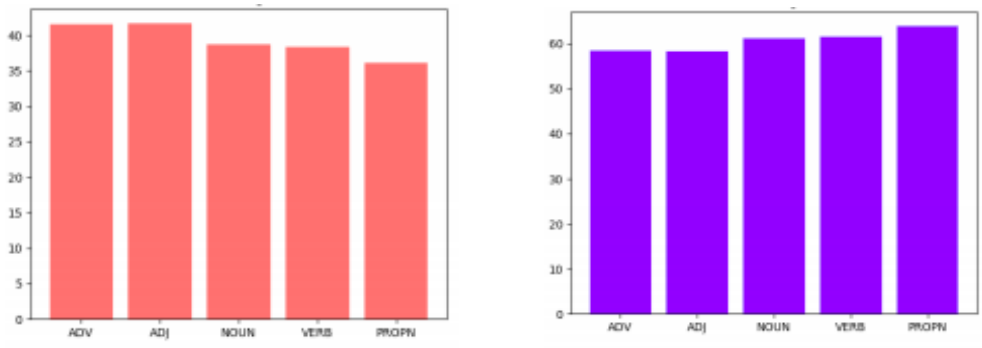


Рис. 2: Полярность фейковых новостей

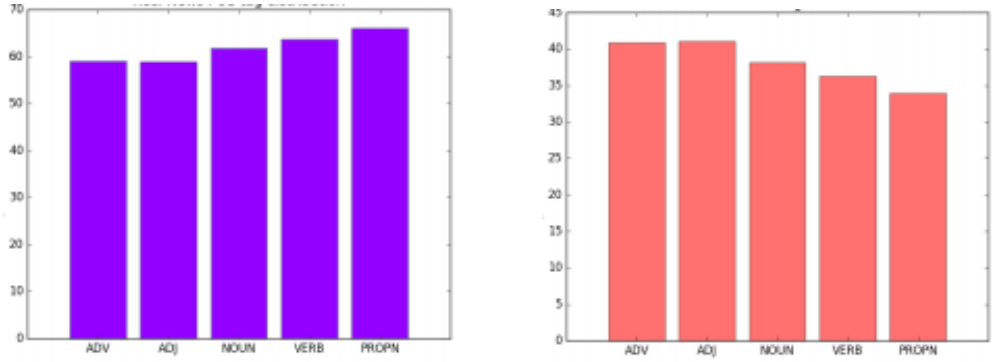
Как до, так и после предварительной обработки, распределение полярности настроения поддельных новостей и настроения реальных новостей в основном одинаково. Как для фейковых, так и для реальных новостей, можно заметить, что есть немного больше позитивных новостей, чем негативных. Также можно заметить, что, хотя и ненамного, но поддельные новости более полярны, чем реальные новости. Присутствует больше выбросов, и данные более распространены.

2.3.2 Графики речевого распределения



(a) Фейковые новости **(b)** Реальные новости

Рис. 3: Распространение POS-тегов до предобработки



(a) Фейковые новости **(b)** Реальные новости

Рис. 4: Распространение POS-тегов после предобработки

Как можно увидеть различия между реальными и поддельными новостями незначительны. В поддельных новостях процент наречий и прилагательных выше, чем все остальные части речи, в то время как процент местоимений ниже; однако в реальной новости, присутствует более высокий процент местоимений. Возможно, это указывает на то, что в подделке новости чаще используют наречия и прилагательные, чтобы приукрашивать свои предложения, в то время в реальных новостях используют больше местоимений, чтобы акцентировать внимание на их законности.

2.3.3 Униграмма и биграмма

Таблица 1: Unigram

Реальные новости		*Фейковые новости*	
До	*После*	*До*	*После*
the	nt	the	nt
to	trump	to	trump
of	people	of	hillary
and	clinton	and	like
in	hillary	for	people
that	said	it	time
for	like	is	clinton
on	new	in	new

Таблица 2: Bigram

Реальные новости		*Фейковые новости*	
До	*После*	*До*	*После*
of the	mr trump	of the	hillary clinton
in the	united states	in the	donald trump
to the	new york	to the	united states
on the	mr trumps	on the	white horse
mr trump	white horse	and the	new york
at the	donald trump	that the	bill clinton
and the	mrs clinton	to be	clinton campaign
that the	said mr	for the	clinton foundation
to be	york times	it is	secretary state
he said	islamic state	with the	nt know

Сравнение результата униграммы и биграммы до и после предварительной обработки показывает, что наше решение удалить стоп-слова является правильным выбором. Поскольку после удаления стоп-слов нетрудно заметить, что униграмма и биграмма становятся более конкретными.

2.4 Pre-training models

2.4.1 Word2Vec

Word2Vec — это набор моделей, принимающих на вход текст и получающих в результате работы представление слов в векторном пространстве на основе контекста. Эти модели (ContinousBagOfWords и skipgram) представляют собой нейронную сеть, задачей которой является реконструкция контекста слов. Так, задача CBOW – предсказание слова на основании контекста, а задача skipgram – предсказать контекст на основе единственного

слова. Их архитектуру можно посмотреть на рисунке 5

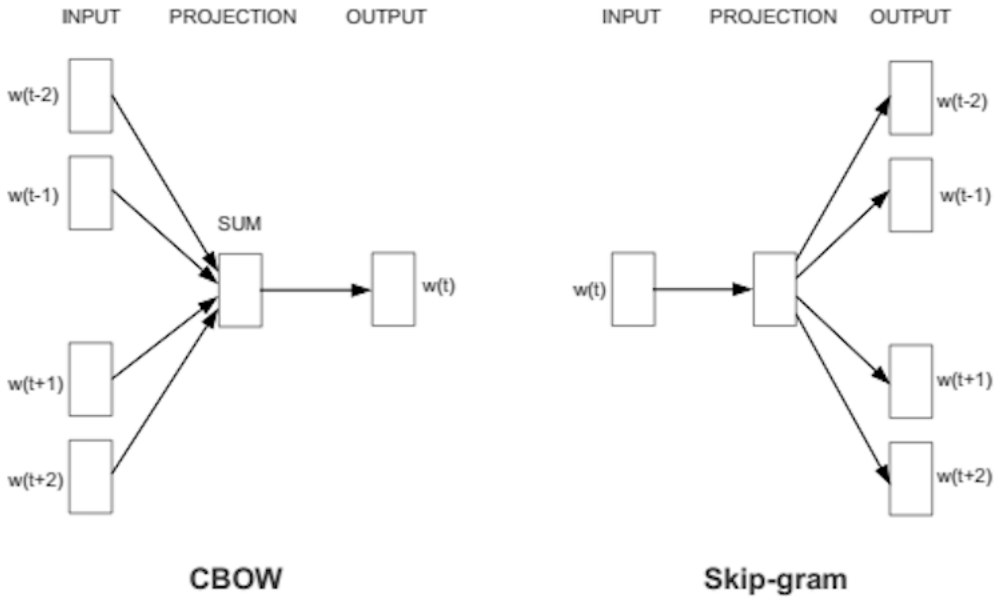


Рис. 5: CBOW и skip-gram архитектура.

Размер векторного пространства $\mathbb{R}^n$ задается вручную, обычно n находится в диапазоне от 100 до 400. Таким образом, данный метод имеет неоспоримое преимущество в виде небольшой размерности векторов. В отличие, например, от методов, которые работают со словарями, где размерность может достигать нескольких тысяч. Принцип работы Word2Vec: максимизация косинусной близости для векторного представления слов, которые появляются в похожих контекстах, и, наоборот, её минимизация для слов, не встречающихся в похожих контекстах. После того, как векторные представления получены, появляется возможность, например, находить близость между двумя словами, получать список наиболее близких элементов в векторном пространстве и так далее. Кроме того, можно получать вектора для целых предложений, используя, например, усредненный вектор всех слов в нём.

Для обучения модели word2vec выбирались новости количество слов в которых было больше среднего числа слов по всем новостям, полагая, что тексты, которые короче, чем средняя длина, имеют меньший контекст, и поэтому для более качественного обучения стоит от них отказаться. В качестве количества функций было выбрано количество по умолчанию, которое равняется

100, поскольку хотелось проанализировать небольшое количество функций. Для этого проекта был выбран очень простой и понятный подход. Получив вектор для каждого предложения, то есть суммируя все векторные представления для каждого слова в предложении (только если слово принадлежит модели word2vec), далее сумма делится на количество слов в предложении, так как нужно было убедиться, что размер текста не влияет на вложения вектора, и, следовательно, было нормализовано наше вложение word2vec.

2.4.2 TF-IDF Vectoraizer

Хотя TF-IDF является старым алгоритмом, его просто и эффективно использовать на этапе предварительной подготовки. Вычисление TfidfVectorizer включает в себя вычисление произведения частоты слова на обратную частоту документа. Как следует из этого термина, TF-IDF вычисляет значения для каждого слова в документе посредством обратной пропорции частоты слова в конкретном документе к проценту документов.

Частота слова $tf(t, d)$ вычисляет сколько раз термин $t \in V(d)$ появляется в документе d. Словарь построен документом d. Частота слова имеет формулу:

$$tf(t, f) = \frac{n(t, d)}{V(d)},$$

где $n(t, d)$ - вхождение слова t в документ d.

Для заданной коллекции документов D, обратная частота документа $idf(t, D)$ представляет собой $\log$ количества документов N деленный на $df(t, D)$, количество документов $d \in D$, содержащих термин t. В результате обычные слова в D будут иметь низкую частоту, в то время как редкие слова будут иметь высокую частоту. Таким образом, термин «частота» с большой вероятностью будет отделять поддельные новости, в которых часто встречаются менее распространенные слова (даже не грамматические), от реальных новостей, которые обычно состоят из общих слов.

Таким образом, оценка $w(d, t)$ для слова увеличивается с количеством повторений, но будет нейтрализована, если слово встречается в слишком

большом количестве документов.

$$w(t,d) = tf(t,d) \cdot idf(t,D)$$

2.5 Classic machine learning based methods

2.5.1 SVM

SVM - это набор контролируемых алгоритмов машинного обучения, в которых создается гиперплоскость для разделения и классификации функций. Оптимальная гиперплоскость обычно рассчитывается путем создания опорных векторов на обеих сторонах гиперплоскости, в которых каждый вектор должен максимизировать расстояние между собой. Другими словами, чем больше расстояние между каждым вектором вокруг гиперплоскости, тем точнее будет граница принятия решения между категориями объектов. В данной работе я использовал только алгоритм SVC(C-Support Vector Classification), поэтому остановимся на нем поподробней.

Математическая формулировка SVC:

Даны векторы $x_i \in \mathbb{R}^p$ обучения, $i = 1, \ldots, n$, принадлежащие двум $y \in \{-1, 1\}^n$ классам, и вектор $\omega \in \mathbb{R}^p$, наша цель — найти $sign(\omega^T \varphi(x) + b)$ такие, что данное предсказание верно для большинства объектов.

Основная задача которую решает SVC :

$$\min_{\omega, b, \zeta} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n}\zeta_i$$

при условии: $y_i(\omega^T\varphi(x_i) + b) \geq 1 - \zeta_i, \zeta_i \geq 0$

Интуитивно, мы пытаемся максимально увеличить зазор между гиперплоскостью и объектами классов (минимизируя $\|\omega\|^2 = \omega^T\omega$), в то время как наложение штрафа происходит при неправильной классификации объекта или за нахождение его в пределах границы классов. В идеале значение $y_i(\omega^T\varphi(x_i) + b)$ было бы ≥ 1 для всех объектов, что указывает на идеальный прогноз. Но данные, как правило, не всегда идеально разделяются с помощью гиперплоскости. поэтому разрешается некоторым объектам находиться на расстоянии ζ_i от границы их класса. Штрафной параметр контролирует силу

штрафа, и в результате действует как обратный параметр регуляризации.

Также большой интерес представляет двойственная задача:

$$\min \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$

при условии: $y^T\alpha = 0, 0 \leq \alpha_i \leq C$, где e является единичным вектором, а $Q^-n \times n$ положительная полуопределенная матрица: $Q_{ij} \equiv y_i y_j \cdot K(x_i, x_j)$, где $K(x_i, x_j) = \varphi(x_i)^T\varphi(x_j)$ это ядро. α_i называются двойственными коэффициентами, и они ограничены сверху . Это двойственное представление подчеркивает тот факт, что тренировочные вектора неявно отображаются в более высокое (возможно бесконечное) пространство функцией φ.

После того как проблема оптимизации решена, выходные данные для данного объекта будут выглядеть так:

$$\sum_{i \in SV} y_i\alpha_i K(x_i, x) + b$$

и предсказанный класс соответствует знаку. Суммирование проводится только по опорным векторам, потому что двойственные коэффициенты α_i равны нулю для других объектов.

Преимущества опорных векторных машин:

- Эффективен в больших пространствах.
- По-прежнему эффективен в тех случаях, когда количество измерений превышает количество образцов.
- Использует подмножество обучающих точек в функции принятия решений (так называемые опорные векторы), поэтому она эффективно использует память.
- Универсальность: различные функции ядра могут быть указаны для решающей функции. Предоставляются общие ядра, но также можно указывать собственные ядра.

К недостаткам опорных векторов машин можно отнести:

- Если число функций намного больше, чем количество выборок, избегайте чрезмерной подгонки при выборе функций ядра, и термин регуляризации имеет решающее значение.

- *SVM* напрямую не предоставляют оценки вероятности, они рассчитываются с использованием дорогостоящей k-кратной перекрестной проверки.

Применение метода SVC в задаче детектирования искусственных новостей:

В моей задаче, для получения хороших результатов, нужно было определить оптимальный C, а также оптимальное ядро. Для этого я использовал кросс-валидацию по K блокам с $K = 3$. Был выполнен сеточный поиск типов ядра и C, чтобы дать наиболее точную модель *SVM*. Параметры, которые использовались для каждого ядра, были линейными и *rbf* (радиальными), в то время как значения, которые использовались для C, составляли 0.25, 0.5 и 0.75. После того как поиск по сетке для этих гиперпараметров был завершен, модель была оценена с использованием наиболее оптимальных гиперпараметров.

2.5.2 Random Forest

Случайные леса — это контролируемый алгоритм машинного обучения. Может использоваться как для классификации, так и для регрессии. Это также самый гибкий и простой в использовании алгоритм. Случайные леса создают деревья решений для случайно выбранных подвыборок данных, получают прогнозы по каждому дереву и выбирают лучшее решение с помощью голосования. Такой алгоритм обеспечивает довольно хороший показатель важности функции.

Алгоритм:

1. Выбирается случайная подвыборка из обучающего набора данных.

2. Создается дерево решений для каждой подвыборки, затем из каждого дерева решений получаем результат прогноза.

3. Проводится голосование за каждый прогнозируемый результат.

4. Выбирается результат прогноза с наибольшим количеством голосов в качестве окончательного прогноза.

Преимущества случайных лесов:

- Случайные леса рассматриваются как высокоточный и надежный метод из-за количества деревьев решений, участвующих в процессе.
- Алгоритм не страдает от проблемы переобучения, потому что он принимает среднее значение по всем прогнозам, что устраняет отклонения.
- Возможность получить относительную важность функции, которая помогает при выборе наиболее полезных функций для классификатора.
- Нечувствительность к любым монотонным преобразованиям значений признаков.

Недостатки случайных лесов:

- Случайные леса медленно генерируют прогнозы, так как состоят из множества деревьев решений. Всякий раз, когда случайный лес делает прогноз, все деревья в лесу должны сделать прогноз для одного и того же заданного ввода, а затем выполнить голосование по нему. Весь этот процесс занимает много времени.
- Большой размер получающихся моделей.

Применение метода Random forest в задаче детектирования искусственных новостей:

Алгоритм случайного леса требует 4 гиперпараметра для настройки, таких как количество деревьев в лесу $(200, 400, 800)$; максимальная глубина дерева $(1, 5, 9)$; минимальное количество выборок, которое должно быть в ведущем узле $(2, 4)$; минимальное количество выборок, которое должно быть в листовом узле $(5, 10)$. Минимальное количество выборок в каждом листовом узле имеет эффект сглаживания модели. Все параметры применяются для поиска в сетке, и, в конце концов, лучший набор параметров может быть определен, так как использовалась кросс-валидация по K блокам с $K = 3$.

2.6 Deep learning based methods

2.6.1 LSTMs (Long Short Term Memory networks)

LSTM — особый тип рекуррентных нейронных сетей (*RNN*), который способен изучать долгосрочные зависимости. В отличии от *RNN*, сети с долговременной краткосрочной памятью не пытаются обучиться запоминать информацию на долгий промежуток времени, так как для них это обычное поведение.

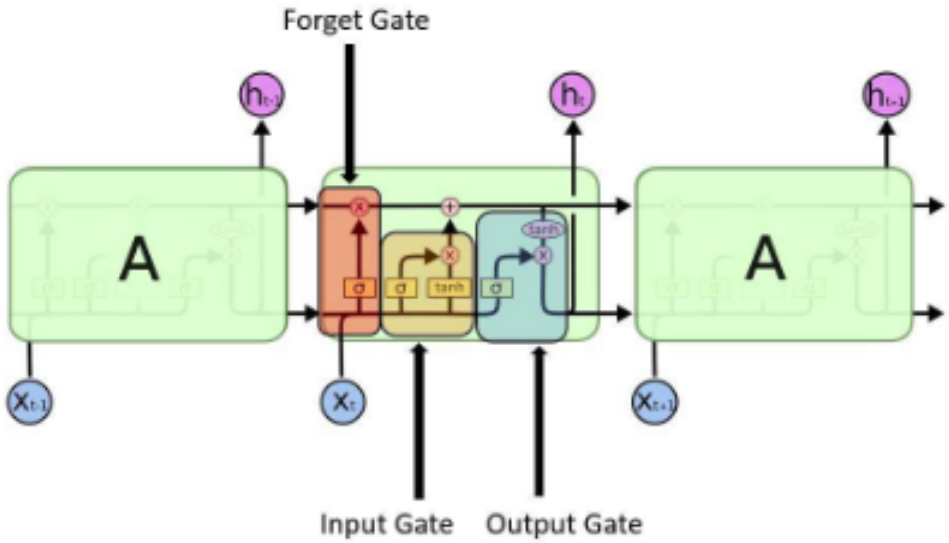
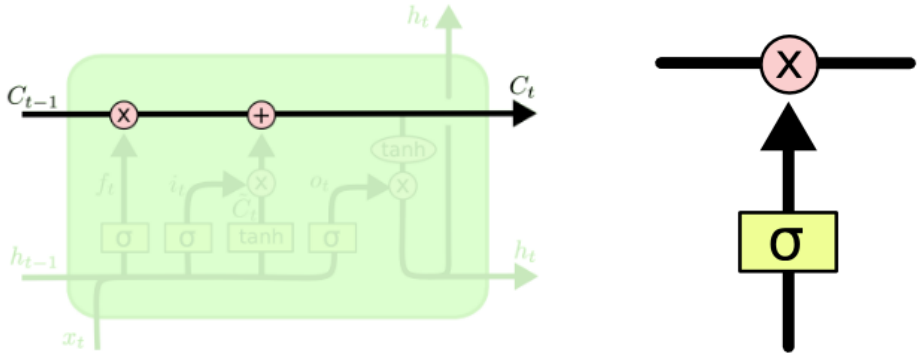


Рис. 6: Структура *LSTM* сети.

Основная идея LSTM сети:

Главным компонентом *LSTM* сети является состояние ячейки (a), которое проходит через всю цепочку подвергаясь лишь нескольким линейным преобразованиям. Удаление информации из состояния ячейки регулируется фильтрами (b), которые представляют собой слой сигмоидальной нейронной сети и поточечного умножения.



(a) Состояние ячейки **(b)** Фильтр

Принцип работы:

1. *LSTM* определяет информацию которую можно исключить из состояния ячейки.
2. Определяет новую информацию которую стоит добавить в состояние ячейки.
3. Производит замену старого состояния на новое.
4. Решает вопрос, с тем какую информацию мы хотим получить на выходе.

Преимущества:

- Отсутствие проблемы с обработкой долговременных зависимостей (по сравнению с *RNN*).

Недостатки:

- Высокие затраты на вычисления, то есть более долгий инференс.

Применение LSTM сети в задаче детектирования искусственных новостей:

На уровне *LSTM* для предотвращения переобучения используется коэффициент отсева 0.2, который также часто используется на практике. Для поиска в сетке применяются следующие параметры: оптимизатор $(Adam, SGD)$, функция активации $(RELU, linear, sigmoid)$, относительно небольшое количество скрытых слоев $(0, 1, 2)$ и узлов $(200, 400, 600)$ выбрано в качестве

основы для поиска в сетке, поскольку это простая задача двоичной классификации и слишком большое количество может привести к переобучению. Из-за ограниченности вычислительных ресурсов поиск в сетке для *LSTM* делится на четыре последовательных этапа. Вместо того, чтобы выполнять поиск по сетке по всем гиперпараметрам одновременно, поиск по сетке сначала выполняется по количеству скрытых слоев, а все остальные гиперпараметры выбираются случайным образом. Затем выполняется поиск в сетке по количеству узлов в скрытом слое(ях), используя наилучшее количество скрытых слоев, найденных на шаге 1. Поиск по сетке завершается, когда все четыре шага завершены. На каждом этапе использовалась K-кратная перекрестная проверка с $K = 3$.

2.6.2 Трансформеры

Этот тип архитектуры нейронной сети был впервые предложен в [11]. Его главная цель, учитывая входную последовательность, преобразовать ее в другую. В архитектуре используются «механизмы внимания», которые отвечают за определение наиболее важных частей входной последовательности. Таким образом, создаются лучшие языковые представления, потому что можно захватывать более длинные отношения в последовательности, обычно более длинные, чем с нейронами *LSTM*, несмотря на то, что они более эффективны в вычислительном отношении, поскольку операции, применяемые к входу, проще.

Трансформатор основан на идее наличия двух частей: кодера и декодера. Кодер создает представление данного входа в другом размерном пространстве, а затем декодер принимает это представление и генерирует другую последовательность. Эта стратегия называется «кодер-декодер». Схема архитектуры трансформатора показана на рисунке 7, где левая часть соответствует блоку кодера, а правая часть - декодеру.

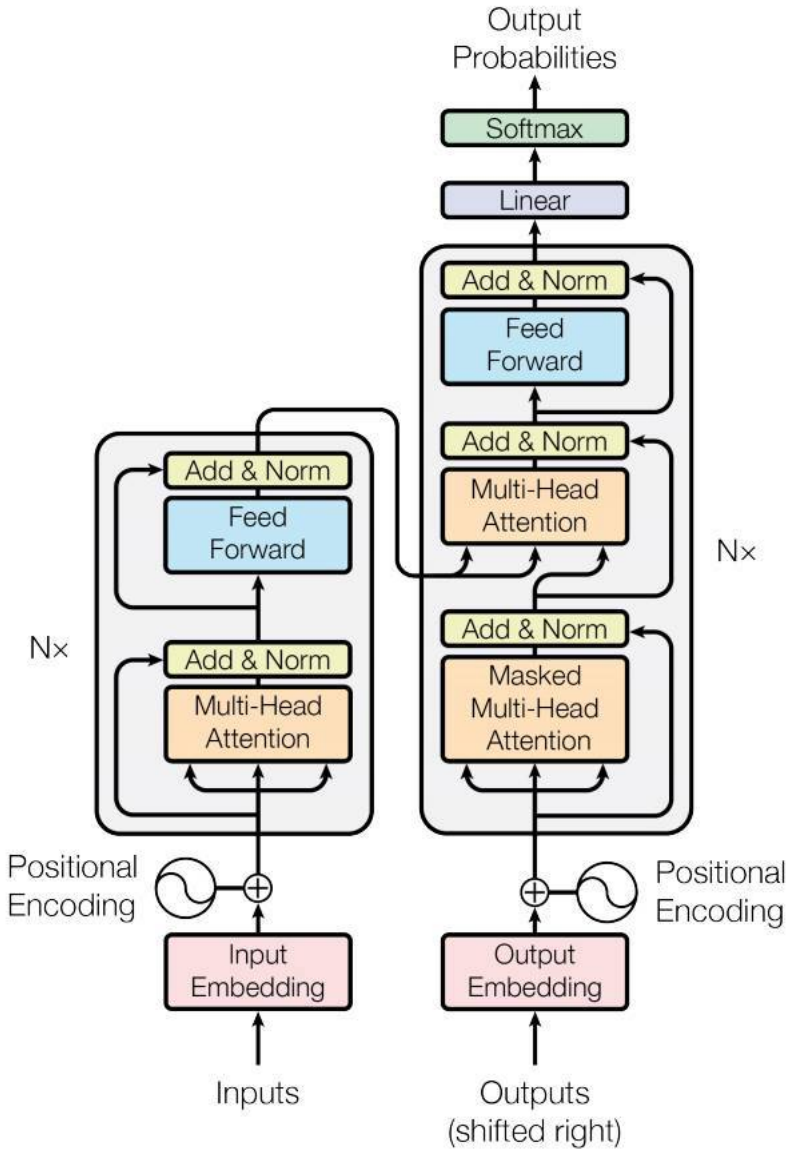


Рис. 7: Блок-схема трансформера.

В каждом блоке используется система «самообслуживания», которая отвечает за выбор наиболее важных частей входной последовательности. Эта система работает, оперируя тремя матрицами: Q, K и V(Query, Key и Value), которые представляют абстракцию для вычисления матрицы внимания Z:

$$Z = softmax\left(\frac{QK^T}{\sqrt{d_k}}V\right)$$

Эти три матрицы изучаются на этапе обучения сети. После получения этих

матриц Z может быть рассчитан, как показано в уравнении выше, где d_k - выбранная размерность каждого ключевого вектора (то есть количество столбцов в K). Также на рисунке 7 видно несколько блоков «Multi-Head Attention ». Они просто повторяют операцию внимания, объясненную выше, n раз, получая n матриц внимания, которые объединяются и умножаются на другую матрицу, W_O, чтобы получить вывод, который подается в блок нормализации («Add & Norm»).

Помимо многоголовой системы, исследователи также предложили использовать пропускаемые соединения, чтобы сигнал идентификации мог передаваться на более глубокие уровни, улучшая процесс обучения. Однако трансформеры имеют существенный недостаток по сравнению с *RNN*: они имеют ограниченный контекст, потому что трансформеры принимают последовательности фиксированной длины. Все его вычисления не имеют состояния, поэтому существует верхний предел на расстояние отношений, которое может моделировать трансформер.

Transformer XL - это простое расширение *Transformer*, которое стремится решить эту проблему. Идея проста: добавить повторение на уровне «сегмента». *Transformer XL* выполняет это путем кэширования скрытых состояний предыдущей последовательности и передачи их в качестве значений при обработке текущей последовательности. Также в *Transformer XL* вводится понятие относительного позиционного вложения. Вместо того, чтобы вложение представляло абсолютную позицию слова (как это было в *Transformer*), *Transformer XL* использует вложение для кодирования относительного расстояния между словами. Это вложение используется при вычислении оценки внимания между любыми двумя словами.

2.6.3 BERT

Традиционные языковые модели обучаются слева направо, чтобы предсказать следующее слово, учитывая последовательность слов. Такие модели имеют ограничение, не требующее, чтобы модель моделировала двунаправленный контекст. Так как для некоторых слов их значение может стать очевидным только тогда, когда вы смотрите одновременно на левый и правый

контекст. Важна одновременная часть: модели, подобные ELMo, обучают две отдельные модели, каждая из которых учитывает левый и правый контекст, но не обучает модель, которая использует обе одновременно.

BERT(Bidirectional Encoder Representations from Transformers) — это языковая модель, созданная исследователями в *Google* основанная на трансформерах, которая решает эту проблему, вводя новую задачу в форме моделирования языка масок. Идея проста: вместо предсказания следующего токена в последовательности, *BERT* заменяет случайные слова во входном предложении специальным токеном *[MASK]* и пытается предсказать, каким был исходный токен. В дополнение к этому, *BERT* использовал мощную архитектуру *Transformer*, чтобы включить информацию из всего входного предложения.

Использование BERT в задаче детектирования фейковых новостей:

Для реализации данного метода использовалась библиотека *Simple Transformers* которая позволяет быстро обучать и оценивать модели. Из-за ограничений вычислительной мощности, обучение проходило в течение трёх эпох с гиперпараметрами, рекомендованными *Google* ($learningrate = 3 \cdot 10^{-5}$, а также $batch\_size = 32$). Использовался *bert-base-cased*.

Преимущества:

- Учитывает одновременно левый и правый контекст, что позволяет обрабатывать большее количество зависимостей между словами.
- Использует мощную архитектуру Transformers позволяющую включать информацию из всего входного предложения.

Недостатки:

- Маркер *[MASK]*, используемый в обучении, не появляется во время тонкой настройки.
- *BERT* предсказывает все маскированные позиции параллельно, а это означает, что во время обучения он не учится обрабатывать зависимости между предсказанием одновременно маскируемых токенов. Другими словами, он не изучает зависимости между своими собственными предсказаниями, что уменьшает количество зависимостей, которые

BERT изучает одновременно, делая сигнал обучения слабее, чем могло бы быть.

2.6.4 XLNet

XLNet является обобщенной моделью авторегрессии перед тренировкой. Вместо того чтобы полагаться на маскирование входных данных для обучения объединенного представления для левого и правого контекстов, как в *BERT*, *XLNet* изучает двунаправленный контекст путем максимизации ожидаемой логарифмической вероятности по всем перестановкам порядка факторизации. Поскольку перестановки могут сделать маркеры из левого и правого контекстов доступными с одной стороны позиций, *XLNet* преодолевает ограничения традиционных моделей авторегрессии, которые можно только обучить однонаправленно представлять контекст, а затем объединить их. Это также позволяет избежать потенциального расхождения в предтренировочных настройках, вызванного повреждением данных из-за маскирования токенов в BERT. Несмотря на перестановку, *XLNet* кодирует позиции, чтобы сохранить информацию о местоположении. *XLNet* использует механизм рекуррентности и схему относительного позиционного кодирования *Transformer-XL* для изучения зависимости сверх фиксированной длины и репараметризирует ее, чтобы устранить неоднозначность порядка факторизации.

Использование XLNet в задаче детектирования фейковых новостей:

Реализовано с помощью библиотеки Simple Transformers [9] с использованием рекомендованных гиперпараметров ($learningrate = 2{\cdot}10^{-5}, batch\_size = 32$) в течении 3 эпох. Использовался *xlnet-base-cased*.

Преимущества:

- Отсутствие скрытых токенов при использовании предобученной модели.
- Формирует контекст из разных мест исходной последовательностии предсказывает каждый токен в последовательности по предыдущим.

Недостатки:

- Более высокие временные затраты на обучение по сравнению с моделью BERT.

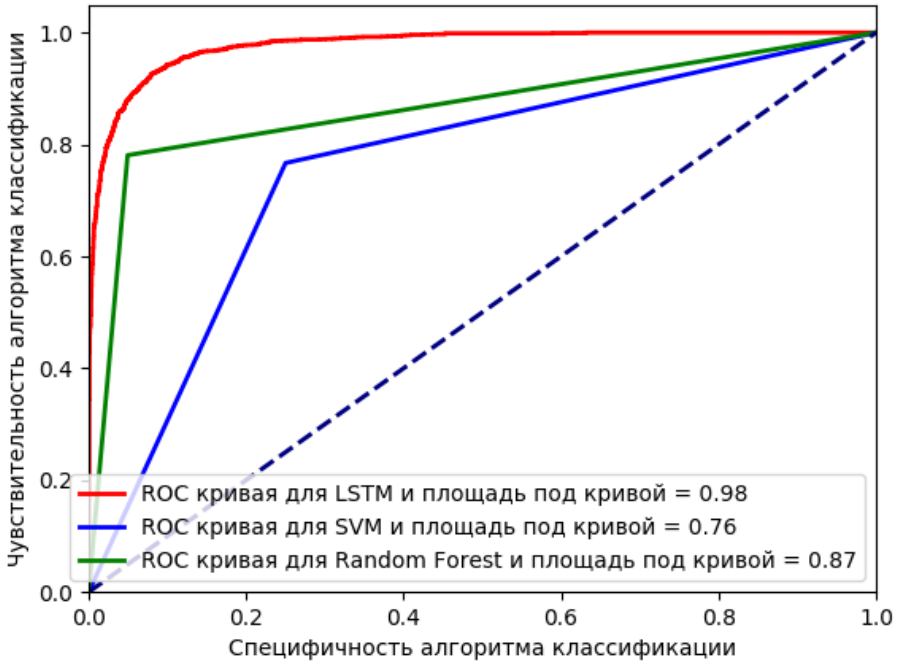
Глава 3. Анализ результатов

Таблица 3: Результаты поиска лучших гиперпараметров по сетке

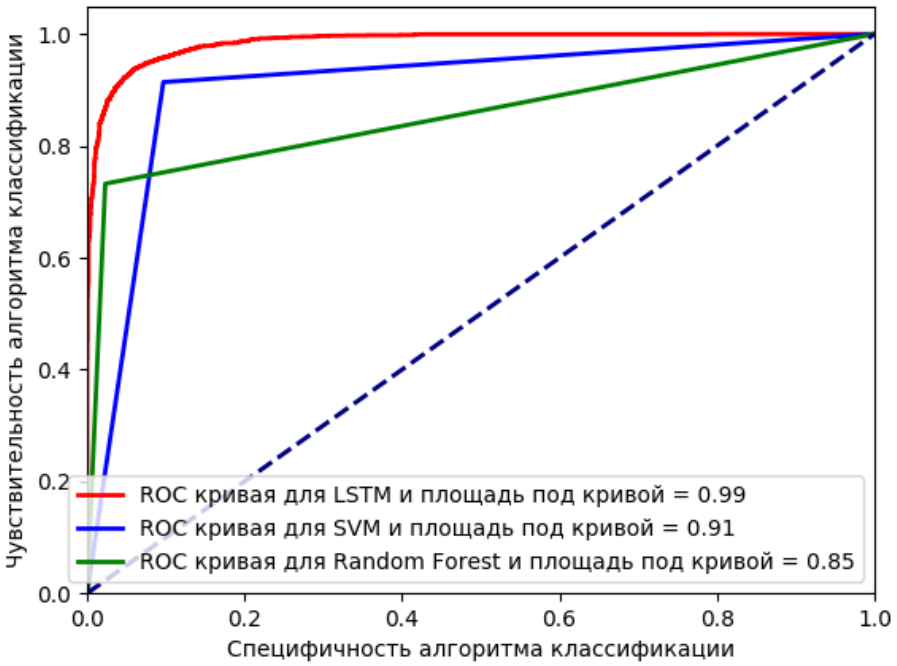
	SVM	Random Forest	LSTM
W2V	Kernel = Linear C = 0.75	Max Depth = 9 Min_samples_leaf = 2 Min_samples_split = 10 N_estimators = 400	Activation = relu Optimizer = Adam Hidden_layers = 2 Memcells = 200 Num_Neurons = 600
TF_IDF	Kernel = Linear C = 0.75	Max Depth = 9 Min_samples_leaf = 4 Min_samples_split = 5 N_estimators = 400	Activation = sigmoid Optimizer = Adam Hidden_layers = 2 Memcells = 200 Num_Neurons = 600

Таблица 4: Точность предсказания(%)

	SVM	Random Forest	LSTM
W2V	91.71	88.6	92.29
TF_IDF	94.58	87.64	93.89



(a) W2V



(b) TF-IDF

Рис. 8: ROC кривые

Среди двух моделей предварительного обучения TF-IDF достигает в целом наилучшей производительности, а Word2Vec работает относительно плохо. Изначально я предполагал, что обучение W2V модели на рассматриваемом датасете сделает модель специфичной для этой задачи, однако ре-

зультаты показывают, что Word2Vec, возможно, придется обучать на большем наборе данных.

Для того, чтобы убедится в том что предобученная модель W2V сможет дать лучшие результаты (относительно моей), я скачал одну из них и использовал в данной задаче. Результаты оказались следующие: SVM - $91.91\%(+0.74\%)$, Random Forest - $89.24\%(+0.64\%)$, LSTM - $93.02(+0.73\%)$. Да, результаты улучшились но не значительно. Также можно заметить, что в целом нейронная сеть LSTM работает стабильнее, поскольку нейронные сети служат мощным универсальным аппроксиматором. Хотя нейронные сети могут очень хорошо вписаться в данные, но они рискуют перегрузить сами данные. В результате LSTM не так хороша, как SVM для TF-IDF. Эта комбинация является лучшей среди всех моделей.

Таблица 5: Точность предсказания(%)

BERT	XLNet
'acc': 0.974,'eval_loss': 0.101, 'fn': 39, 'fp': 101, 'mcc': 0.947,'tn': 2974, 'tp': 2278	'acc': 0.977, 'eval_loss': 0.100, 'fn': 43, 'fp': 81,'mcc': 0.953, 'tn': 2994, 'tp': 2274

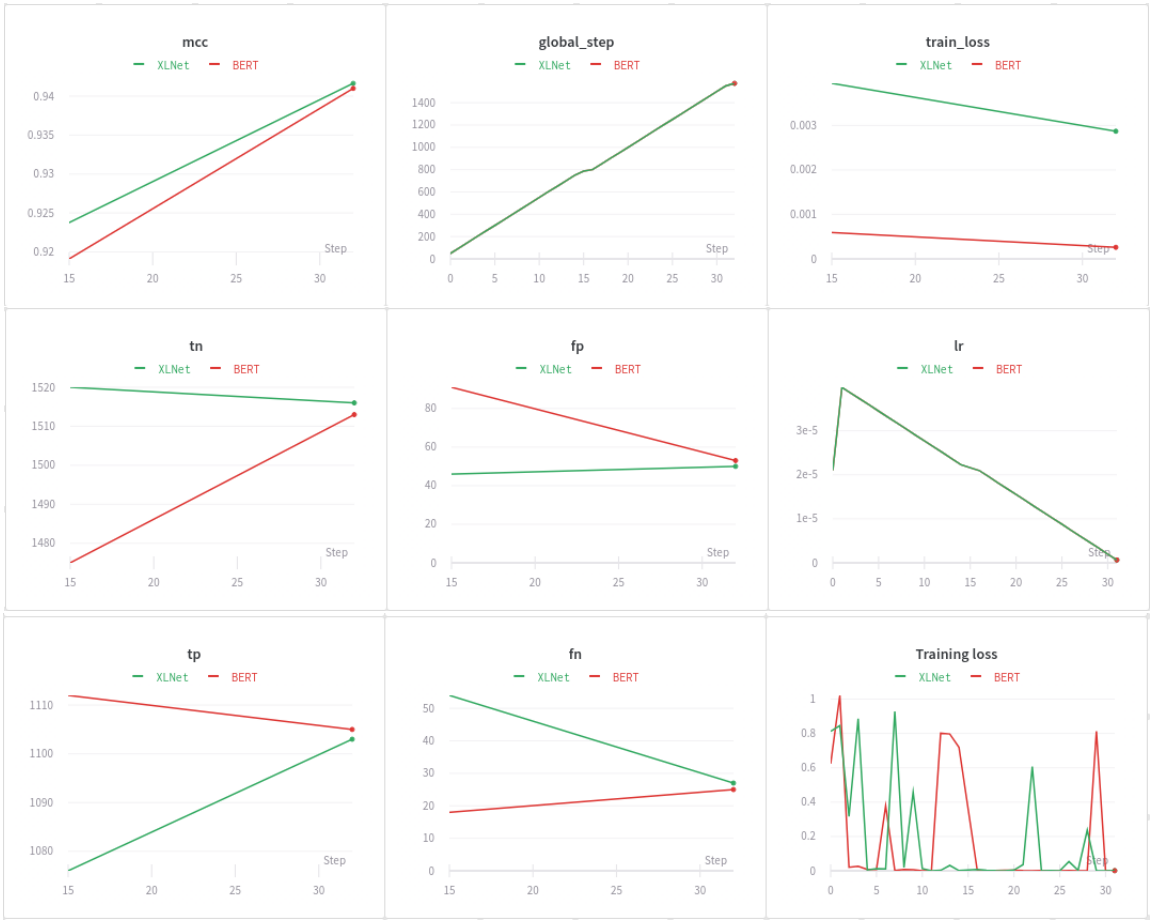


Рис. 9: Визуализаация BERT и XLNet

Рисунок 9 были получены с помощью библиотеки wandb, которая предоставляет возможность визуализировать обучение моделей. Как и ожидалось языковые модели BERT и XLNet показали более высокую точность предсказания. С одной стороны, лучшие результаты показала модель XLNet $(+0.3\%)$, с другой, время обучения модели BERT было значительно меньше чем у XLNet. Так как в процессе обучения были выбраны рекомендованные гиперпараметры, для архитектур BERT и XLNet. Можно предположить, что более интенсивная подстройка этих архитектур может привести к дальнейшим лучшим результатам.

Вывод

В ходе выполнения данной работы были решены следующие задачи:

- Проведен обзор существующих решений
- Реализованы методы основанные на классических алгоритмах машинного обучения (SVM, Random Forest)
- Реализованы методы основанные на глубоком обучении (LSTM, BERT, XLNet)
- Проанализировать полученные результаты

Наилучшая точность, которой удалось добиться - 97.7%. Лучше всего себя показала модель XLNet.

Список литературы

[1] Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro. «Some like it hoax: Automated fake news detection in social networks.». 2017

[2] Nguyen Vo and Kyumin Lee. «The rise of guardians: Fact-checking url recommendation to combat fake news.». 2018

[3] Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, Niloy Ganguly. «Stop clickbait: Detecting and preventing clickbaits in online news media.». 2016

[4] Kai Shu, H. Russell Bernard, Huan Liu. «Studying fake news via network analysis: Detection and mitigation.». 2018

[5] Kyle Shaffer, Svitlana Volkova. «Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter.». 2018

[6] Rashki Hannah, Choi Eunsol «Truth of varying shades: Analyzing language in fake news and political fact-checking.». 2017

[7] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, Michael M. Bronstein. «Fake news detection on social media using geometric deep learning.». 2019

[8] Junaed Younus Khan, Md. Tawkat Islam Khondaker, Anindya Iqbal, Sadia Afroz. «A benchmark study on machine learning methods for fake news detection.». 2019

[9] Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Zhoujun Li, Philip S. Yu. «Ti-cnn: Convolutional neural networks for fake news detection.». 2018

[10] `https://www.kaggle.com/c/fake-news`

[11] N. Shazeer et al A. Vaswani «Attention is all you need.». 2017