Московский государственный университет имени
М.В.Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра алгоритмических языков

Грибов Илья Юрьевич

Оптимизация порождения состязательных примеров в задаче классификации

Выпускная квалификационная работа

Научный руководитель:
д-р техн. наук
Лукашевич Наталья Валентиновна

Москва, 2024

Аннотация

Состязательные примеры - это данные, которые специально создаются или изменяются с целью нарушить работу алгоритма машинного обучения. Этот подход становится все более важным в контексте развития искусственного интеллекта, поскольку он позволяет проверять устойчивость и надежность алгоритмов в различных областях, таких как компьютерное зрение и обработка естественного языка.

В данной работе рассматриваются подходы, основанные на интерпретации прогнозов модели и анализе их внутренней структуры, к оптимизации процесса порождения состязательных примеров для задачи классификации текстов. В процессе работы было проведено сравнение различных подходов оптимизации и их влияния на итоговую генерацию. Промежуточные метрики и результаты сопровождаются иллюстрационными примерами. Результаты исследования могут быть применены для повышения устойчивости и надежности классификационных текстовых моделей и аугментации данных в области обработки текстов.

Содержание

1 Введение

Большие нейронные модели NLP, в первую очередь BERT/GPT-подобные модели [1], [2], получили широкое распространение как в исследованиях, так и в промышленности, ввиду установленной зависимости между размером модели и ее результативностью при решении задач. Однако как следствие, подобные модели обычно рассматриваются в виде некоторого **"чёрного ящика"**. Поэтому растёт озабоченность работой и надёжностью данных моделей, так как подобные системы часто принимают участие при принятии решений в различных сферах, где цена ошибки очень высока.

Зачастую невозможно учесть все виды отказов и проблем, которые могут возникнуть при эксплуатации данных моделей, поэтому оценка качества также должна проводиться с помощью модельных пояснений. Предоставление этих пояснений часто является основной мотивацией для понятия **“интерпретируемости”** моделей машинного и глубокого обучения. То есть, когда модели выдают неверный результат или их поведения начинает отклоняться от стандартного и первоначального задуманного, возникает необходимость в объяснениях, почему модель приняла то или иное решение на основе полученных входных данных. На сегодняшний день существует немало способов, которые могут позволить изучить поведение модели. Таковыми могут быть **состязательные примеры**.

Состязательные примеры - это специально подобранные входные данные, которые могут ввести в заблуждение или заставить ошибиться модель. В области обработки естественного языка (NLP) состязательные примеры могут быть созданы различными методами, такими как перефразирование текста, вставка опечаток, добавление отвлекающих элементов или замена слов синонимами. Цель состязательных примеров в NLP - проверить и улучшить устойчивость и надёжность моделей глубокого обучения. В данной работе в дальнейшем будут рассмотрены уже существующие на сегодняшний момент методы генерации состязательных примеров для текстов.

Также наряду с понятием состязательных примеров вводится понятие **состязательная устойчивость** — это мера восприимчивости модели к состязательным примерам. Её часто измеряют, используя процент попыток, которые приводят к успешным атакам. **Состязательным возмущением** же обычно называют то, насколько сильно состязательный пример отличается от исходного.

Хотя атаки в NLP не могут найти такое состязательное возмущение, которое бук-

вально неотличимо от исходного ввода, так как данные являются дискретными, а не непрерывными, они могут найти возмущение, которое очень похоже. Состязательные атаки NLP можно разделить на две группы, основанные на их понятиях «сходства».

Original Input	Connoisseurs of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: **Positive (77%)**
Adversarial example [Visually similar]	**Aonnoisseurs** of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: **Negative (52%)**
Adversarial example [Semantically similar]	Connoisseurs of Chinese **footage** will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: **Negative (54%)**

Рис. 1: Иллюстрация различных видов схожести текстов

Визуальное сходство: при котором исходный и состязательный текст едва отличимы друг от друга при зрительном обращении, но при этом изначальный смысл может быть искажен. На Рис. 1 видно, что слово Connoisseurs было подменено похожим на него с точки зрения написания словом Aonnoisseurs. То есть смысл текста был искажен, но заметить данную опечатку можно не сразу. **Семантическое сходство**: при котором два исходный и состязательный текст имеют схожий смысл, но при этом имеет явные отличия при зрительном обращении. На Рис. 1 видно, что слово film было заменено синонимом footage. То есть смысл текста был сохранен, однако видно, что сам текст был слегка видоизменен.

Однако вопрос оптимизации порождения состязательных примеров является не менее важным, чем их генерации. В дальнейшем будет показано, что сами по себе методы генерации не могут должным образом достичь наилучшего эффекта. В данной работе будут рассмотрены различные подходы, которые позволяют выделять слова в тексте, наибольшим образом влияющие на итоговое предсказание. Данные методы опираются на выходы модели или внутренную (латентную) структуру данных в ходе её работы. Тем самым за счет атаки наиболее важных слов в тексте для модели в совокупности с существующими методами порождения можно повысить эффективность состязательных примеров.

2 Постановка задачи

Целью данной работы является изучение и сравнение методов интерпретации нейросетевых моделей для оптимизации порождения состязательных примеров, то есть достижения наибольшего успеха атаки при минимальном числе вносимых изменений. Таким образом, выбранные методы генерации состязательных примеров должны удовлетворять критериям либо семантического сходства, либо визуального сходства. Для этого необходимо:

1. Подготовить датасеты для обучения BERT-подобной модели (вариация BERT с некоторыми модификациями), изучить их структуру и выявить особенности текстов, а также обучить выбранную модель;

2. Замерить качество baseline решения, суть которого заключается в порождении состязательных примеров без использования методов оптимизации, либо с использованием упрощенных способов оптимизации;

3. Провести сравнение выбранных методов оптимизации между собой в задаче выделения слов в тексте, наибольшим образом влияющих на прогноз обученной модели. Провести анализ словарей выделяемых слов в текстах каждым методом;

4. Сравнить по подобранным метрикам качество baseline решения и решения с использованием методов оптимизации;

5. Проанализировать полученные результаты и сделать выводы.

3 Методы интерпретации

На сегодняшний день существует немало подходов к интерпретации прогнозов модели. Таковым может быть **конструирование похожих примеров** [3], суть которого заключается в поиске схожих примеров с точки зрения модели из тренировочных данных. Таким образом, можно понять, на какие токены модель больше всего обращала внимание для осуществления поиска. Еще одним подходом к интепретации является **изучение концепций** [3], то есть попытка выявить какие-то предрасположенности и предрассудки у модели ко входным данным. Однако в данной работе будет рассмотрена группа методов, которые тем или иным способом пытаются оценить вклад каждого отдельного токена на итоговый прогноз модели. Таким образом, данную группу методов можно охарактеризовать как **извлечение важности входных признаков**.

3.1 LIME

Метод **LIME** (сокращение от **Local Interpretable Model-agnostic Explanations)** представляет собой метод машинного обучения, который помогает объяснить принятие решений моделями. Он работает, создавая интерпретируемую локальную модель вокруг конкретного предсказания модели машинного обучения, и пытается оценить вклад каждого отдельного признака в итоговый прогноз [4]. В качестве модели, которую можно легко интерпретировать, берется **Logistic Regression**, и действительно, при нормализации признаков веса перед признаками тем больше по модулю, чем важнее сам признак. В качестве признаков LIME использует непосредственно сами слова предложения, важность которых надо извлечь (то есть для векторного представления используется модель **BOW**), а в качестве обучающей выборки LIME некоторым некоторым образом удаляет или оставляет слова в предложении, таким образом объем выборки можно довести при необходимости до 2^n. LIME также учитывает не только саму структуру выборки, но и расстояние до сгенерированных объектов по некоторой метрике. Таким образом, оптимизационная задача, которую решает LIME для извлечения важности слов выглядит следующим образом:

$$\epsilon(x) = \min_{g \in G} \left\{ L(f, g, \pi_x) + \Omega(g) \right\}$$

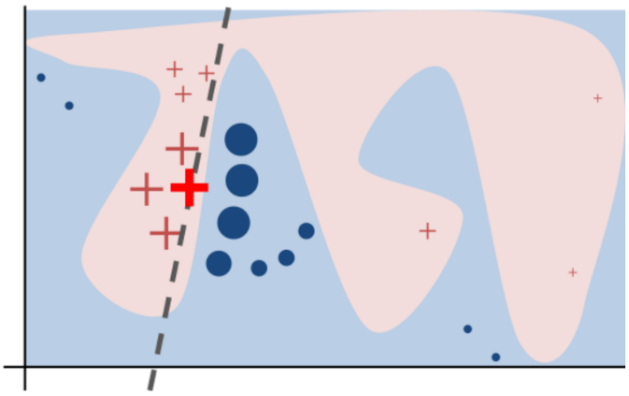

Рис. 2: Иллюстрация работы метода LIME

$$L(f, g, \pi_x) = \sum_{z', z \in Z'} \pi_x(z)(f(z) - g(z'))^2$$

$$\Omega(x) = \infty \mathbb{1}\{\|w_g\| > K\}$$

$$\pi_x(z) = \exp(-D(x, z)^2/\sigma^2)$$

где ищется Logistic Regression наилучшим образом приближающая выходы модели на конкретном тексте (L - слагаемое) и при этом не столь сложная (Ω - слагаемое), в качестве ядра π_x используется экспоненциальное ядро.

Главная особенность lime заключается в том, что он использует исходную модель лишь как black-box, извлекая из нее лишь выходы на конкретных текстах.

3.2 SHAP

Метод **SHAP** (расшифровывается как **SHapley Additive exPlanations**) - это способ объяснения прогнохов моделей машинного обучения, основанный на теории кооперативных игр и концепции значимости признаков. Основная идея метода SHAP заключается в том, чтобы оценить вклад каждого признака в предсказания модели, учитывая все возможные комбинации признаков [5]. То есть оценивается маргинальный вклад каждого отдельного признака по следующей формуле:

$$\Phi_i = \frac{1}{N} \sum_{s \in S} \frac{|S|!(K - |S| - 1)!}{K!} \{f(s \cup i) - f(s)\}$$

При этом так как коалиции для оценки могут образовываться различными перестановками, то надо нормировать оценку на число всех возможных коалиций. Как и в

методе LIME в качестве признаков выступают отдельные слова текста, которым необходимо придать определенный вес значимости слова.

Ниже проиллюстрировано схематическое устройство работы SHAP, где маргинальный вклад каждого признака “толкает” предсказание в определенную сторону, пока не получится просто предсказание модели на исходном тексте.

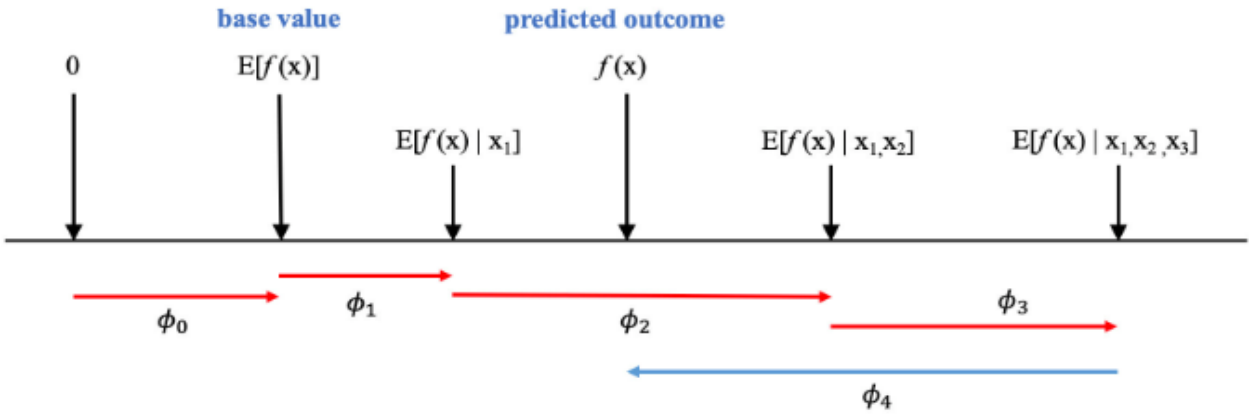

Рис. 3: Иллюстрация работы метода SHAP

Как и для LIME, SHAP не требует знание о внутреннем устройстве модели и работает с ней как с black-box, извлекая лишь выходны модели на конкретном тексте. Кроме того, одной из основных особенностей метода SHAP является то, что за счет оценки важности признаков, модели можно сравнивать между собой (Если мы знаем какие признаки важны для предсказания, то та модель будет лучше, на которой SHAP придал больший вес для данного признака)

3.3 Attention

Основная суть метода **Attention** заключается в использовании весов механизма self-attention для извлечения вклада каждого токена на каждый на k-ом слое transformer [6]. Таким образом, после прохода через модель поданных данных на выходе получается l матриц весов внимания размера sequence_len $\times$ sequence_len, где l - это количество encoder в используемой архитектуре, а sequence_len - длина подаваемой последовательности в модель. Каждый элемент a_{ij}^{k} такой матрицы показывает влияние j-ого токена на i-токен на k-ом по счету encoder. Далее данные l матриц с весами внимания усредняются поэлементно и получается следующая матрица:

$$\begin{pmatrix} a_{11}^{res} & a_{12}^{res} & ... & a_{1seq\_len}^{res} \\ a_{21}^{res} & a_{22}^{res} & ... & a_{2seq\_len}^{res} \\ ... & ... & ... & ... \\ a_{seq\_len1}^{res} & a_{seq\_len2}^{res} & ... & a_{seq\_lenseq\_len}^{res} \end{pmatrix}$$

где a_{ij}^{res} находится по следующей формуле:

$$a_{ij}^{res} = \frac{\sum_{k=1}^{l} a_{ij}^{k}}{l}$$

Таким образом, получается итоговая оценка вклада каждого токена на каждый. Для извлечения оценки важности слов на итоговый прогноз необходимо в данной полученной матрице взять 1 строку, так как в ней содержательно содержится итоговый вклад каждого токена на [CLS] токен, который используется для классификации текстов.

3.4 ALTI

Основная идея метода **ALTI** заключается в переосмыслении значений матрицы внимания в механизме **Self-Attention** архитектуры transformer. Соответственно данный метод интерпретации работает только с **BERT**/**GPT**-подобными архитектурами. Суть данного метода заключается в декомпозиции работы механизма self-attention, в линеаризации **Layer Normalization** и перевзвешивании вклада каждого скрытого вектора [7].

Как уже известно: в различных архитектурах используется нормализации данных для ускорения обучения, в частности в трансформерах используется нормализация вдоль скрытого состояния каждого слова по следующей формуле:

$$LN(u) = \frac{u - \mu(u)}{\sigma(u)} \cdot \gamma + \beta$$

Однако можно показать, что данное преобразование можно представить в матричном виде по следующей формуле:

$$LN(u) = \frac{1}{\sigma(u)} Lu + \beta$$

где матрица L представляется в следующем виде:

$$L = \begin{pmatrix} \gamma_1 & 0 & ... & 0 \\ 0 & \gamma_2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & \gamma_n \end{pmatrix} \quad \begin{pmatrix} -\frac{n-1}{n} & -\frac{1}{n} & ... & -\frac{1}{n} \\ -\frac{1}{n} & -\frac{n-1}{n} & ... & -\frac{1}{n} \\ ... & ... & ... & ... \\ -\frac{1}{n} & -\frac{1}{n} & ... & -\frac{n-1}{n} \end{pmatrix}$$

Таким образом, multi-head self-attention для поданных векторов x_i можно переписать следующим образом:

$$\widehat{y_i} = LN(\widehat{x_i} + x_i)$$

где

$$\widehat{x_i} = W_o Concat(z_i^1, ..., z_i^H), z_i^h = \sum_j^J A_{i,j}^h W_V^h x_j$$

то есть

$$\widehat{x_i} = \sum_h^H W_o^h z_i^h + b_o$$

тогда получается

$$y_i = LN(\sum_j^J \sum_h^H W_o^h A_{i,j}^h W_V^h x_j + b_o + x_i) = \sum_j^J T_i(x_j) + \frac{1}{\sigma(\widehat{x_i} + x_i)} L b_o + \beta$$

при учете, что

$$T_i(x_j) = \begin{cases} \frac{1}{\sigma(\widehat{x_i}+x_i)} L \sum_h^H W_o^h A_{i,j}^h W_V^h x_j & \text{если } i = j \\ \frac{1}{\sigma(\widehat{x_i}+x_i)} L(\sum_h^H W_o^h A_{i,j}^h W_V^h x_j + x_i) & \text{если } i \neq j \end{cases}$$

То есть показано, что влияние j-го признака на i-ый выражается через вектор $T_i(x_j)$ на некотором l-ом по счету encoder, однако простая $\| * \|_2$ норма от $T_i(x_j)$ приведет лишь к тому, что:

1. произойдет потеря информации о расположении в пространстве данного вектора
2. если в векторе $T_i(x_j)$ присутствуют слишком большие значения, то их влияние будет доминирующим при финальном учете важности, что негативно сказывается на итоговой оценке весов слов.

Для борьбы с данными проблемами был предложен немного иной пересчет важности вклада каждого отдельного слова по следующим формулам:

$$d_{i,j} = \|y_i - T_i(x_j)\|_1$$

$$c_{i,j} = \frac{\max(0, -d_{i,j} + \|y_i\|_1)}{\sum_k \max(0, -d_{i,j} + \|y_i\|_1)}$$

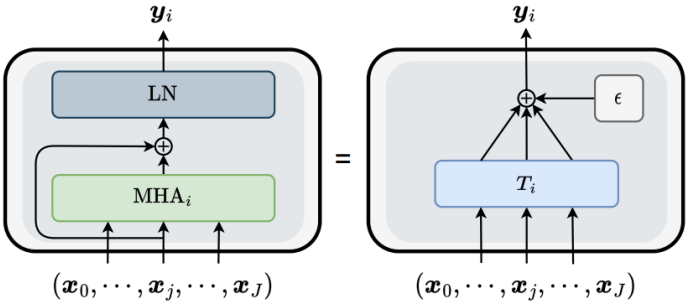

Рис. 4: Декомпозиция ALTI

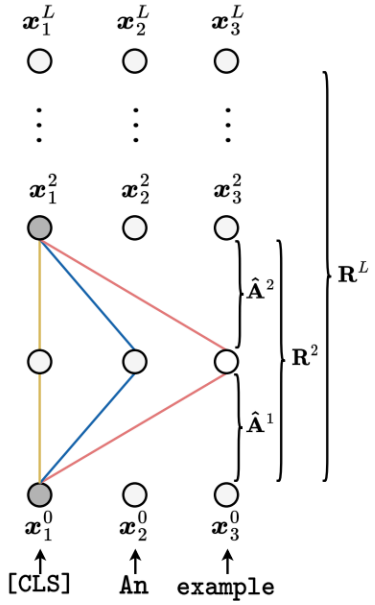

Рис. 5: Rollout

Таким образом, учитывается пространственная информация, и уходит вырожденность важности вектора в важность одного большого значения в векторе. Последняя модификация в методе ALTI - это метод **Rollout**, который позволяет отследить влияние исходного токена при подачи в трансформер на скрытое представление токена после механизма self-attention на некотором l-ом по счету encoder. Пусть A - некоторая матрица весов важности, тогда с учетом **residual connection** вклад токена на другой токен через один encoder можно выразить как:

$$\widehat{A}^l = 0.5A^l + 0.5I$$

Тогда для учета важности влияния исходных токенов на скрытые представления токенов после механизма self-attention на l-ом по счету encoder выразиться как:

$$R = \widehat{A}^l * \widehat{A^{l-1}} * ... * \widehat{A^1}$$

То есть матрица R будет содержать те самые обновленные веса важности каждого токена на каждый. Объединив все преобразования, в итоге получим:

$$R = \widehat{C^l} * \widehat{C^{l-1}} * ... * \widehat{C^1}$$

3.5 Gradient

Основная суть метода **Gradient** $\times$ **Input** заключается в том, что выход модели аппроксимируется по первому члену из разложения в **ряд Тейлора** [8]:

$$m(X^o) \approx grad(m(X^o)) * X^o$$

Таким образом выход приблизительно равен градиенту в точке на значение самой точки и автоматически получается оценка важности каждого отдельного признака. Но не всегда такую оценку возможно получить, так как на некоторых участках функция может быть постоянной, поэтому градиенты дадут 0 значение. Модификацией простого метода Gradient $\times$ Input является **Integrated Gradients** [8], которые учитывают изменение функции не только в конкретное точки а вдоль прямой линии между данной точкой и некоторым baseline x^o по следующей формуле:

в непрерывном случае

$$IntegratedGrads_i(x) = (x_i - x_i^o) \int_{\alpha=0}^{1} \frac{\partial F(x^o + \alpha * (x - x^o))}{\partial x_i} \, d\alpha$$

или в дискретном случае

$$IntegratedGrads_i(x) \approx (x_i - x_i^o) * \sum_{k=1}^{m} \frac{\partial F(x^0 + \frac{k}{m} * (x - x^0))}{\partial x_i} * \frac{1}{m}$$

Одно из главных свойств Integrated gradients заключается в том, что отслеживается изменение не только в конкретной точке, а на пути к этой точке от некоторой другой, что делает невозможным ситуацию, когда Integrated gradients были бы равны 0. Кроме того Integrated Gradients обладают важным свойством в случае когда целевая функция F (нейронная сеть) почти везде дифференцируема (что истинно почти всегда):

$$\sum_{i=1}^{m} IntegratedGrads_i(x) = F(x) - F(x^0)$$

То есть если взять baseline, при котором $F(x^o) \approx 0$, то получится разложение значения функции F в точке на признаки с весами, равными их важности.

4 Методы генерации состязательных примеров

Основная идея генерации состязательных примеров для текстов состоит в искажениях символов, слов или целых предложений так, чтобы при этом допускалось либо визуальное сходство, либо семантическое сходство между состязательным примеров и исходным текстом.

4.1 Символьное искажение

Подход на уровне символов основан на идее добавления естественных опечаток в слово текста, то есть **удаление**/**del**, **замена**/**sub**, **вставка**/**ins**, согласно его **WordPiece токенизации** [9], [10]. Таким образом, данный метод оперирует **визуальным сходством** между состязательным примером и исходным текстом. Подход пытается сделать опечатки так, чтобы лексемы искаженного слова, полученные после токенизации, имели минимальное пересечение с лексемами исходного слова. Для подсчета данного пересечения используются метрики расстояния между лексемами слов, в случае если было сгенерировано несколько слов-кандидатов на замену исходному:

1. **Min Token Intersection (MTI)** - Кандидаты с минимальным пересечениям по лексемам остаются. Например, для слова “melodrama” → [’mel’, ’##od’, ’##rama’] мы оставил “melodarma” → [’mel’, ’##oda’, ’##rma’], вместо “melodramas” → [’mel’, ’##od’, ’##rama’, ’##s’];

2. **Max Token Count Distance (MTCD)** - Вычисляется разность между количеством лексем до и после Word Piece токенизации. Например, в этом случае для слово «melodrama» → [’mel’, ’##od’, ’##rama’] оставляется «nelodrama» →[’ne’, ’##lo’, ’##dra’, ’##ma’], вместо "melodarma"→ [’mel’, ’##oda’, ’##rma’];

3. **Min Token Intersection** + **Max Token Count Distance** (**MTI** + **MTCD**) - Если по какой-то из двух выше метрик нельзя выбрать лучшего кандидата, то просматривается ещё другая метрик и уже по ней происходит отбор.

4.2 Искажение слов

Основная идея генерация состязательных примеров на уровне слов заключается в подмене наиболее важного слова некоторым синонимом, чтобы изменения были как

можно более естественными, то есть сохранялись исходные морфологические особенности (например, падеж, род и т.д.), тогда в данном случае имеет место **семантическое сходство**. Таким образом, для достижения данной цели можно применить следующий подход [10]:

1. Первым делом необходимо выделить наиболее важное слово в тексте и лемматизировать его;
2. Затем для данного лемматизированного слова надо извлечь его синоним;
3. Далее необходимо уточнить морфологические признаки полученного синонима;
4. В самом конце надо встроить полученный синоним с его морфологическими признаками в текст вместо исходного слова.

Наиболее сложным этапом в данной схеме является извлечение морфологических признаков для полученного синонима. Сделать это можно с помощью предобученного **BERT** на задаче **MLM** по следующей схеме[10]:

1. Оригинальное слово заменяется полученным синонимом (в своем изначальном виде);
2. Синоним обрабатывается WordPiece токенизатором;
3. В зависимости от того, на сколько частей был разбит синоним токенизатором, производится маскирование следующим образом:
    - Если синоним был разбит на более чем 1 часть, то тогда последняя из них заменяется токеном **[MASK]** и предсказывается с помощью BERT. Полученное предсказание заменяет последнюю часть токенизированного синонима и содержит в себе морфологические признаки для встраивания в контекст.
    - Если синоним не был разбит, то тогда окончание для встраивания в контекст предсказывается итеративно. На первой итерации к синониму добавляется токен **[MASK]** и предсказывается. Далее на каждой последующей итерации удаляется по одному символу с конца из исходного синонима (2 итерация - 1 символ, 3 итерация - 2 символа и так далее) и аналогично добавляется и предсказывается токен **[MASK]**. Всего таких итераций будет 5. Таким образом,

получается массив окончаний синонима, из которого выбирается наиболее вероятное. Данное наиболее вероятное предсказание замещает удаленные к тому моменту символы. Полученный синоним встраивается в исходный текст.

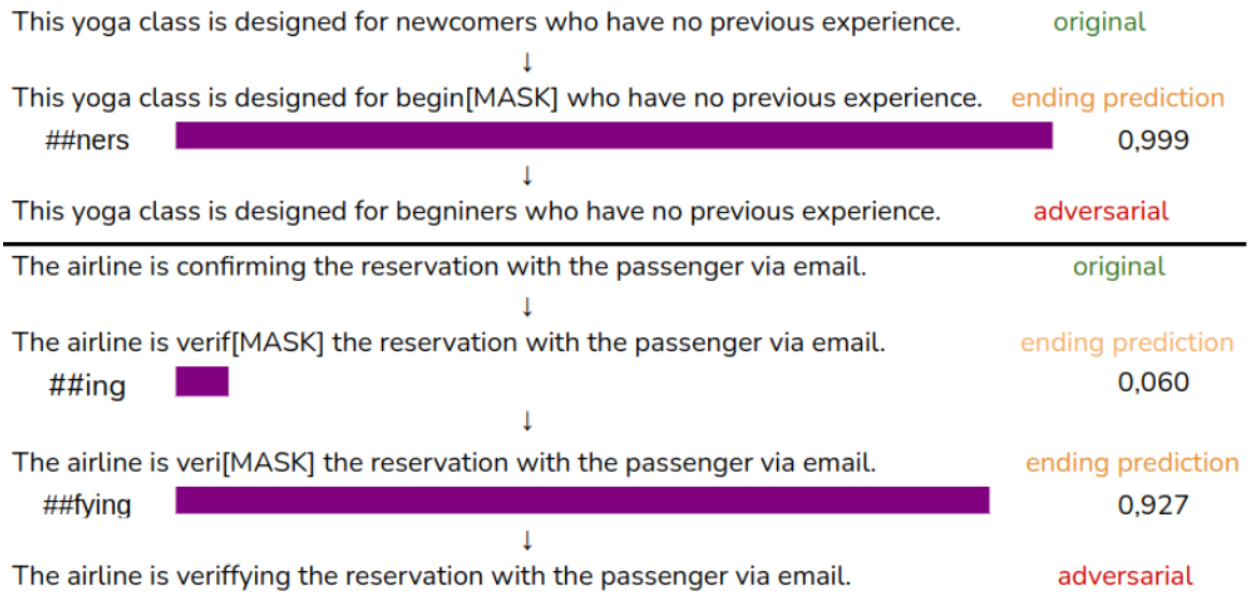

Рис. 6: Иллюстрация встраивания синонима в контекст

ну

5 Постановка экспериментов

В качестве методов для составления состязательных примеров использовались следующие операции:

1. Символьные опечатки наиболее важных слов заменой, вставкой или замещением соседних на клавиатуре символов;

2. Замены наиболее важных слов синонимами с дальнейшим встраиванием (прогнозирование морфологических признаков) их в контекст предложения. Итоговое слово конструируется с помощью предобученного MLM BERT, который предсказывает морфологическое окончание для лемматизированного подобранного синонима 4.2. Синоним для исходного лемматизированного слова искался путем минимизации косинусного расстояния;

Так как на поставленную задачу были поставлены ограничения в виде близкого **семантического** или **визуального** сходства, то для генерации состязательных примеров допускается 1 символьное искажение (удаление, замена или вставка соседнего на клавиатуре символа) 1 или 2 слов в тексте, либо замена 1 или 2 слов синонимами.

Однако сначала во всех методах порождения состязательных примеров надо определить последовательность слов, подлежащих изменению, путем вычисления их важности. Сделать это можно двумя способами:

- Рассматривается предложение $S = \{w_1, w_2, ..., w_n\}$, где w_i представляет собой токен текста. После удаления слова w_i результирующий пример обозначается $S \backslash w_i = S \backslash \{w_i\} = \{w_1, ..., w_{i-1}, w_{i+1}, ..., w_n\}$. Модель, подлежащая атаке, M_y(-), используется для представления оценки прогнозирования для метки y. Оценка важности I_{w_i} рассчитывается как разница между оценкой предсказания до и после удаления слова w_i из предложения, которая формально определяется следующим образом:

  $$I_{w_i} = My(S) - My(S \backslash w_i)$$

  Таким образом, получены оценки важности для каждого слова в предложении. Затем они используются для выбора наиболее влиятельных слов в контексте атакуемой модели. К данной группе можно отнести, например, метод **SHAP**.

- Извлечение важности каждого отдельного токена текста за счет интерпретации прогноза модели на входных данных. К данной группе можно отнести такие методы как: **Attention**, **ALTI**, **LIME**, **Gradient**.

Основной интерес вызывает вопрос: какой из методов извлечения важности токенов при заданных ограничениях на задачу приведет к лучшему качеству сгенерированных состязательных примеров.

5.1 baseline решение

В качестве baseline решения рассматриваются следующие методы выделения наиболее важных слов для прогноза: **loss** и **random**.

5.1.1 Loss

Данный способ выделения важности токенов относится к первой группе, то есть пусть $F(x)$ - некоторая целевая оценочная функция, которая принимает на вход метку класса для текста и сам текст и затем в качестве результата выдает скалярное значение. Чем выше/ниже значение этого результата, тем менее подходящим оказался текст для данной метки. Таким образом, выкидывая различные слова из текста, можно оценивать их важность для предсказания модели. В задаче классификации в качестве функции F можно взять **CrossEntropyLoss**. Плюсом данного подхода также является то, что эксплуатация исходной модели происходит в режиме "чёрного ящика"и не зависит от ее внутреннего устройства. С другой стороны, в отличии от других рассматриваемых методов Loss работает только с размеченными объектами из выборки.

5.1.2 Random

Название метода говорит само за себя, вместо целенаправленной оценки важности слов для предсказания модели используется **стохастическое сэмплирование** порядка слов из некоторого распределения. Данный метод может сэмплировать как хорошие, так и плохие слова для генерации состязательных примеров, но в среднем на передний план будут выходить слова наименьшим образом влияющие на предсказания модели.

5.2 Параметры исследований

В качестве атакуемой модели был выбран **DistilBERT** [11], состоящий из 6 кодировщиков, предобученный на большом корпусе текста **RuCorpora**, включающий в себя словарь из более чем 500 000 слов из разных источников и дообученный на датасетах по классификации тональности.

Для лемматизации слов использовался **Pymorphy2**, для поиска синонима из некоторой базы использовались предобученные по методу **Word2Vec** [12] вектора **RusVectors** из **Gensim** (как уже упоминалось выше, путем минимизации косинусного расстояния) и граф знаний русских слов **RuWordNet** (если граф знаний выдавал несколько синонимов, то среди них выбиралось единственное с минимальным косинусным расстоянием по отношению к исходному слову). Для обучения моделей использовался **PyTorch** и **Transformers**, для выделения отдельных слов из текста использовалась библиотека **Razdel**. Для методов интерпретации использовались готовые библиотеки: **Lime**, **Shap** и **Captum**.

6 Датасеты

В качестве данных использовался датасет **SentiRuEval-2016-banks** [13], содержащий 9392 отзывов на банки за 2016 год и датасет **SentiRuEval-2016-telecoms** [13], содержащий 8643 отзывов на телекоммуникационные компании за 2016 год. Изначальная задача, поставленная перед атакуемой моделью, для обоих датасетов заключалась в классификации тональности отзывов.

6.1 SentiRuEval-2016-banks

Особенность данного датасета заключается в том, что:

1. присутствовал явный дисбаланс классов в пользу нейтральных отзывов (примерно 55%), затем в пользу отрицательных отзывов (примерно 30%), и только потом - положительных отзывов (примерно 15%);
2. Для многих отзывов не всегда сразу возможно понять тональность текста
3. Присутствие во многих отзывах знаков препинания характерных той или иной тональности, которые могут повлиять на прогноз модели

Примеры отзывов из данного датасета:

1. Автокредит в россельхозбанк 2012
2. Sberbank CIB: Цены на нефть по-прежнему остаются на очень низких уровнях РБК НПЗ стремятся отложить импортные...
3. RT В Костроме прямой саботаж со стороны - потеряли три дня от сбора подписей

6.2 SentiRuEval-2016-telecoms

Особенность данного датасета заключается в том, что:

- присутствовал явный дисбаланс классов в пользу нейтральных отзывов (примерно 65%), затем в пользу отрицательных отзывов (примерно 25%), и только потом - положительных отзывов (примерно 10%);

- Многие отзывы читаются и воспринимаются с трудом, не всегда сразу возможно понять тональность текста
- Присутствие во многих отзывах знаков препинания характерных той или иной тональности, которые могут повлиять на прогноз модели

Примеры отзывов из данного датасета:

- RT Мтс банк кредитная карта проценты
- Арбитражный суд Тамбовской области подтвердил МТС своей рекламой вводило потребителей в заблуждение
- Мне нравится Волейбол. Мировая лига. Матч недели. Прямая трансляция. Австралия Сербия на канале Спорт Интерактивное ТВ, Ростелеком

7 Метрики

7.1 ASR

Метрика **ASR (Attack Success Rate)** характериузет процент успешных атак на модель. Под атакой подразумевается подача в модель состязательного и исходного текстов и дальнейшее исследование выходов. Атака считается успешной, если прогноз модели изменился (в случае классификации текстов: произошла смена предсказанного класса), то есть ASR метрика показывает, на каком проценте текстов исходная модель изменила свой прогноз. Таким образом, чем выше ASR, тем лучше оказался метод генерации состязательного примера. ASR можно записать следующей формулой:

$$ASR = \left(\frac{\text{Число успешных атак}}{\text{Общее число атак}} \right) \times 100$$

7.2 USE

На высоком уровне идея метрики **USE (Universal sentence encoder)** состоит в том, чтобы разработать кодировщик, который для любого заданного предложения извлекает N-мерное векторное представление (для BERT-подобных моделей N чаще всего равно 512). Данный кодировщик представляет из себя суммаризатор, который на любую ограниченную по длине последовательность токенов выдаёт числовой вектор данной последовательности. Используя данные вектора и косинусную метрику, можно оценивать схожесть двух и более текстов между собой: 1-значит они семантически индентичны, -1-значит тексты совсем не похожи друг на друга. Аналогично ASR чем выше метрика USE, тем лучше, так как важно наравне с успешностью атаки сохранить исходный семантический смысл текста. Существует несколько вариантов кодировщика для вычисления вектора предложения:

7.2.1 Кодировщик трансформера (BERT)

В этом варианте используется энкодерная часть оригинальной архитектуры трансформера. Архитектура состоит из 2-24 многоуровневых слоев. Каждый слой включает в себя **механизм внимания**, за которым следует последовательность **полносвязных**

слоёв и **нормализация** по каждому отдельному латентному вектору. На выходе получается 768-мерный вектор в качестве векторного представления входного предложения.

7.2.2 Deep Averaging Network (DAN-кодировщик)

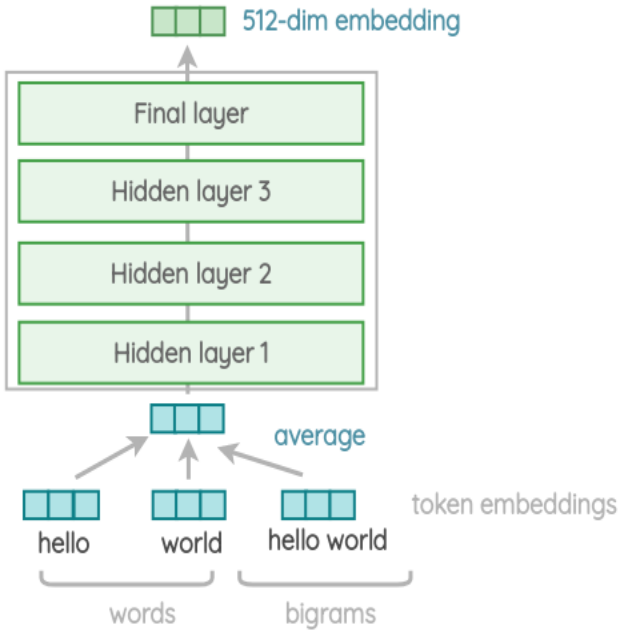

Рис. 7: Архитектура DAN

В этом варианте кодировщик представляет из себя последовательность подряд идущих **fully-connected слоёв** нейронной сети. Векторное представление слов и биграмм, присутствующие в предложении, усредняются вместе. Затем они проходят через n-слойный глубокий DNN с прямой связью, чтобы получить 512-мерный вектор предложения в качестве выходных данных.

8 Результаты

8.1 Словари

В первую очередь интересно рассмотреть распределение слов наиболее часто подверженных атаке для обоих датасетов и для всех методов извлечения важности. По вертикали расположены слова, по гризонтали - частота:

- Для датасета с отзывами на банки **SentiRuEval-2016-banks**

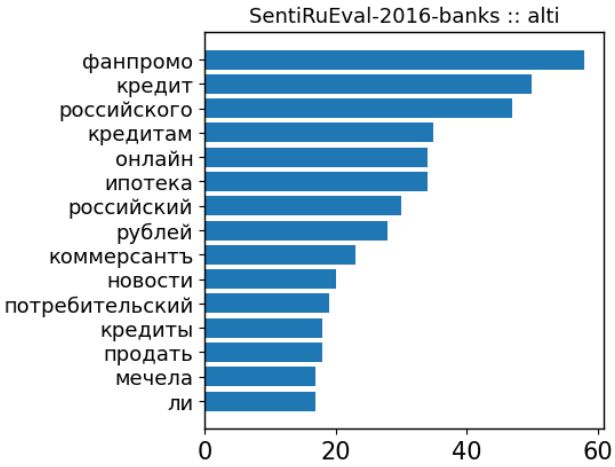

Рис. 8: ALTI

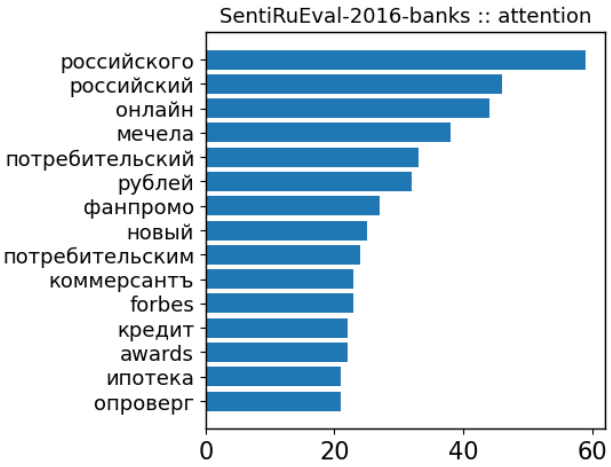

Рис. 9: Attention

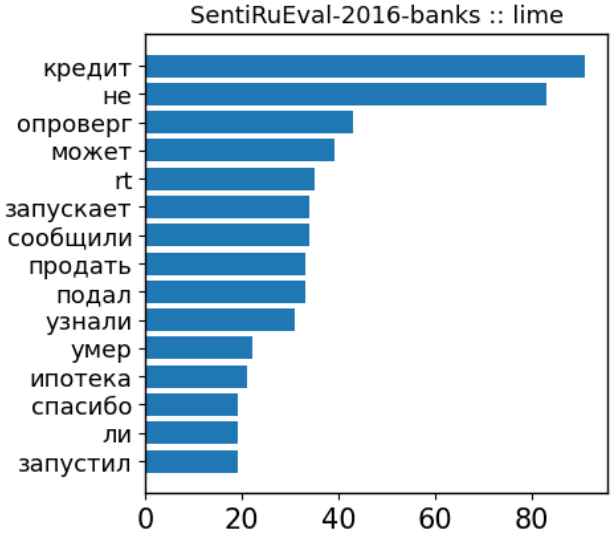

Рис. 10: LIME

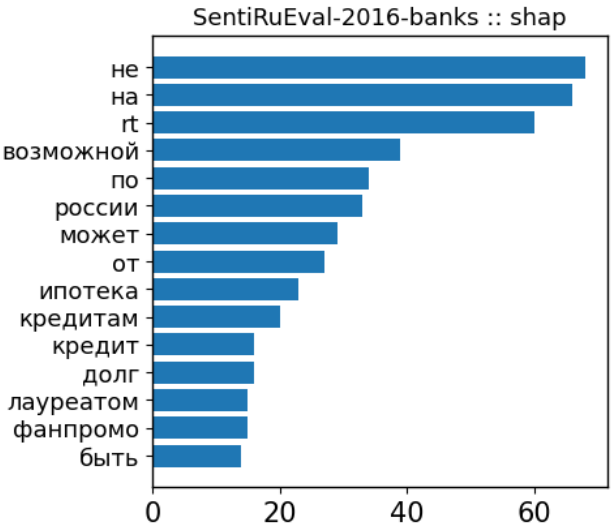

Рис. 11: SHAP

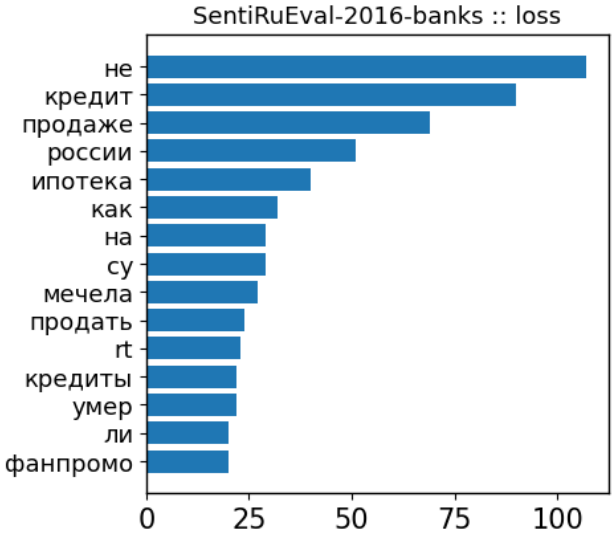

Рис. 12: Loss

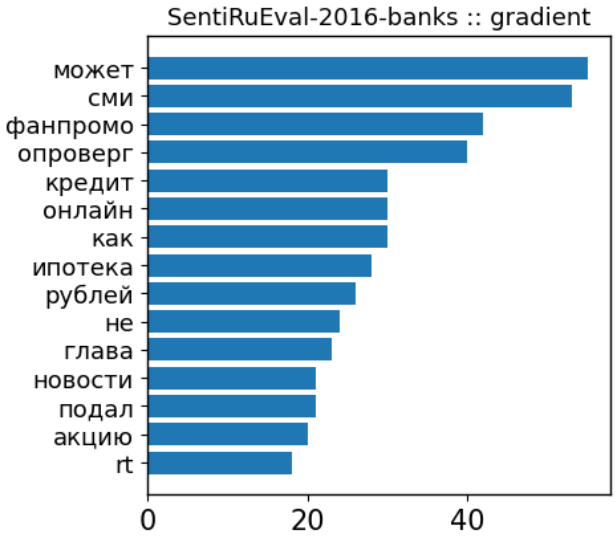

Рис. 13: Gradient

Исходя из вида полученных словарей наиболее часто выделяемых слов различных методами интерпретации можно сделать вывод, что почти все подходы в основном выделяют слова, связанные с банковской тематикой. Тем не менее некоторые методы, например SHAP, очень часто выделяют слова слабо относящиеся к данной тематике (как видно из рисунка, например, это частица не и прилагательное возможной). Данная корреляция может быть связана с тем, что атакуемая модель была дообучена не на самое высокое качество на используемом датасете.

- Для датасета с отзывами на компании сотовой связи **SentiRuEval-2016-telecoms**

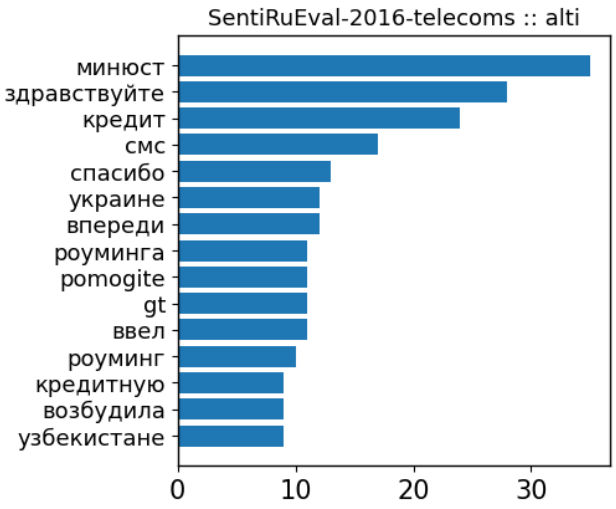

Рис. 14: ALTI

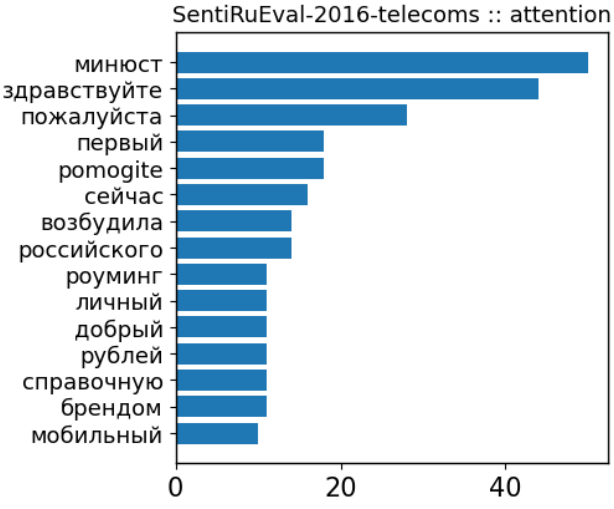

Рис. 15: Attention

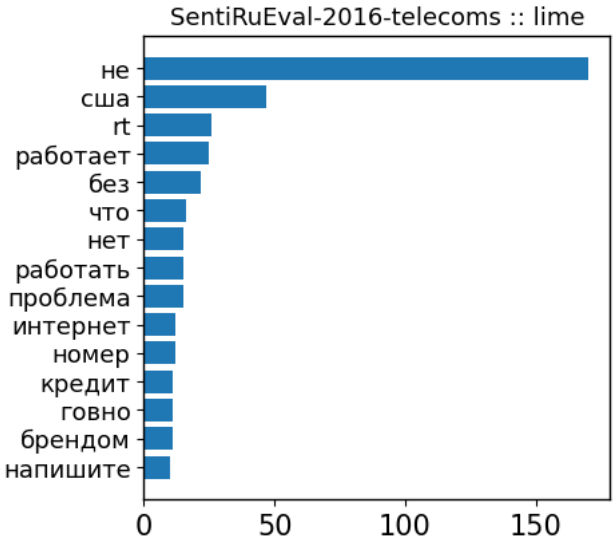

Рис. 16: LIME

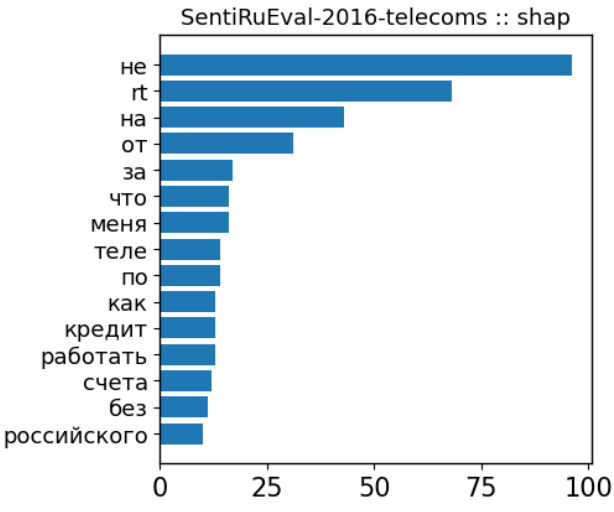

Рис. 17: SHAP

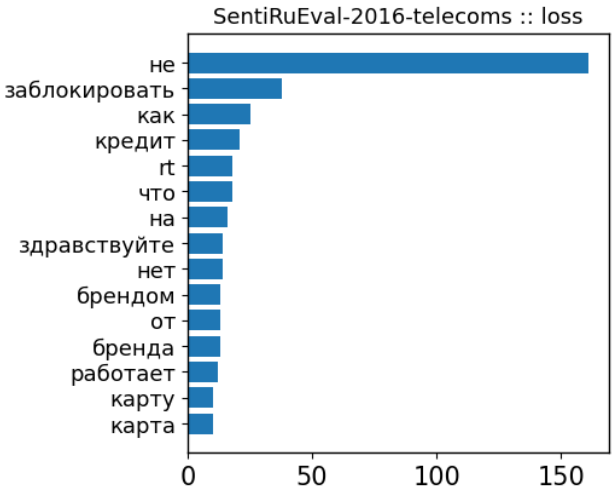

Рис. 18: Loss

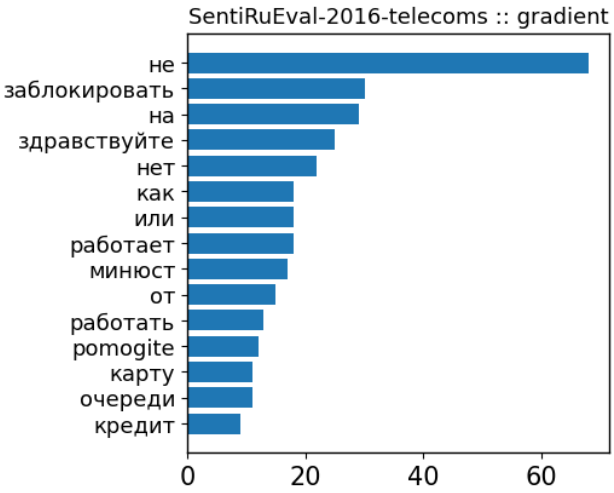

Рис. 19: Gradient

Исходя из вида полученных словарей наиболее часто выделяемых слов различных методами интерпретации можно сделать вывод, что большинство подходов в основном выделяют слова, которые мало связаны с тематикой телекомпаний. Тем не менее некоторые методы, например ALTI, иногда выделяют слова относящиеся к данной тематике (например, слова роуминг или смс, как видно из рисунка). Данная корреляция как и в случае с предыдущим датасетом может быть связана с тем, что атакуемая модель была дообучена не на самое высокое качество на используемых данных.

8.2 Время работы

Ниже указано время работы каждого метода в секундах. Стоит отметить, что некоторые методы можно оптимизировать по скорости работы, например, **SHAP** и **LIME** за счет подбора гиперпараметров: для SHAP - количество различных маргинальных контрибуций, для LIME - объем генерируемой выборки и параметры обучаемой Logistic Regression, - однако в таком случае это может сказаться на итоговом качестве сгененрированных состязательных примеров

Таблица 1: Среднее время работы методов оптимизации

	lime	shap	loss	gradient	alti	attention
время работы	0.23	0.08	0.04	0.046	0.11	0.007

8.3 Итоговые показатели

Ниже представленые таблицы для каждого рассматривамого датасета и для 1-2 искаженных слов, где в строках расположены исследуемые методы оптимизации, а в столбцах - способы генерации состязательных примеров. На пересечении находятся 3 значения в следующем формате **ASR/BERT_USE/DAN_USE**. Жирным шрифтом выделены лучшие показатели **ASR** для каждого столбца. Под **ins_char** подразумевается вставка символа, **del_char** - удаление символа, **sub_char** - замена символа случайным ближайшим на клавиатуре, **sub_word** - замена слова синонимом, **del_word** - удаление наиболее важного слова из текста. В каждой колонке для удобства восприятия добавлена приписка **AR/BU/DU** (сокращение от **ASR/BERT_USE/DAN_USE**). Также в самой последней строчке у всех таблиц приведены усредненные значения по каждому столбцу.

Таблица 2: SentiRuEval-2016-banks :: искажение 1 слова

method	del_word	ins_char	del_char	sub_char	sub_word
	AR/BU/DU	AR/BU/DU	AR/BU/DU	AR/BU/DU	AR/BU/DU
alti	15/0.93/0.85	12/0.95/0.84	10/0.96/0.84	12/0.95/0.84	15/0.9/0.85
attention	12/0.95/0.85	9/0.97/0.84	8/0.97/0.84	8/0.97/0.84	12/0.94/0.85
gradient	16/0.95/0.85	11/0.96/0.84	11/0.97/0.84	12/0.96/0.84	15/0.94/0.83
lime	**22**/0.95/0.84	13/0.96/0.84	**13**/0.96/0.84	**15**/0.96/0.84	15/0.94/0.83
shap	11/0.96/0.84	11/0.96/0.83	9/0.97/0.83	11/0.96/0.83	12/0.95/0.84
loss	20/0.95/0.84	**14**/0.96/0.84	**13**/0.96/0.84	14/0.96/0.84	**17**/0.94/0.84
random	8/0.97/0.86	7/0.96/0.87	7/0.97/0.86	6/0.97/0.87	9/0.95/0.86
avg_value		11/0.96/0.843	10.1/0.966/0.841	11.1/0.961/0.843	**13.6**/0.937/0.843

Таблица 3: SentiRuEval-2016-telecoms :: искажение 1 слова

method	del_word	ins_char	del_char	sub_char	sub_word
	AR/BU/DU	AR/BU/DU	AR/BU/DU	AR/BU/DU	AR/BU/DU
alti	13/0.94/0.9	10/0.96/0.89	10/0.96/0.89	11/0.95/0.89	15/0.91/0.89
attention	11/0.95/0.9	8/0.97/0.89	8/0.97/0.89	8/0.97/0.89	11/0.94/0.89
gradient	14/0.96/0.89	11/0.97/0.89	10/0.97/0.89	11/0.97/0.89	14/0.95/0.89
lime	**21**/0.96/0.88	**13**/0.97/0.87	**13**/0.97/0.87	**14**/0.96/0.87	15/0.95/0.89
shap	10/0.97/0.89	8/0.97/0.88	7/0.97/0.88	9/0.97/0.88	10/0.95/0.87
loss	**21**/0.96/0.89	**13**/0.96/0.88	11/0.97/0.88	**14**/0.96/0.88	**16**/0.95/0.88
random	8/0.97/0.91	6/0.98/0.9	5/0.97/0.91	5/0.96/0.91	8/0.96/0.9
avg_value		9.9/0.969/0.886	9.1/0.969/0.887	10.3/0.963/0.887	**12.7**/0.944/0.887

Таблица 4: SentiRuEval-2016-banks :: искажение 2 слов

method	del_word	ins_char	del_char	sub_char	sub_word
	AR/BU/DU	AR/BU/DU	AR/BU/DU	AR/BU/DU	AR/BU/DU
alti	19/0.88/0.8	18/0.92/0.81	14/0.92/0.81	17/0.91/0.81	20/0.85/0.81
attention	18/0.9/0.8	14/0.93/0.8	12/0.94/0.8	16/0.93/0.8	18/0.88/0.81
gradient	22/0.91/0.8	18/0.93/0.8	16/0.94/0.8	18/0.93/0.8	20/0.9/0.8
lime	**30**/0.91/0.78	**20**/0.92/0.78	**19**/0.93/0.78	**21**/0.92/0.78	**21**/0.89/0.79
shap	17/0.93/0.79	15/0.93/0.79	14/0.94/0.79	16/0.93/0.79	17/0.9/0.8
loss	27/0.92/0.79	19/0.93/0.79	18/0.93/0.79	20/0.92/0.79	**21**/0.89/0.8
random	14/0.93/0.82	12/0.94/0.82	12/0.94/0.83	12/0.94/0.83	15/0.9/0.83
avg_value		16.6/0.929/0.798	15/0.934/0.8	17.1/0.926/0.8	**18.8**/0.887/0.806

Таблица 5: SentiRuEval-2016-telecoms :: искажение 2 слов

method	del_word	ins_char	del_char	sub_char	sub_word
	AR/BU/DU	AR/BU/DU	AR/BU/DU	AR/BU/DU	AR/BU/DU
alti	19/0.89/0.87	15/0.92/0.86	15/0.93/0.86	16/0.92/0.86	19/0.86/0.86
attention	17/0.91/0.86	13/0.94/0.86	12/0.94/0.86	13/0.94/0.86	16/0.89/0.86
gradient	20/0.92/0.86	15/0.94/0.85	15/0.94/0.85	16/0.94/0.85	**20**/0.91/0.84
lime	**29**/0.92/0.83	**18**/0.93/0.83	**18**/0.94/0.83	19/0.93/0.84	**20**/0.91/0.84
shap	14/0.94/0.85	13/0.94/0.84	12/0.95/0.84	14/0.94/0.84	16/0.91/0.85
loss	**29**/0.92/0.85	**18**/0.93/0.84	17/0.94/0.84	**20**/0.93/0.85	19/0.91/0.85
random	11/0.94/0.88	10/0.95/0.88	9/0.95/0.87	9/0.94/0.88	13/0.92/0.87
avg_value		14.6/0.936/0.851	14/0.941/0.85	15.3/0.934/0.854	**17.6**/0.901/0.853

Исходя из полученных выше таблиц, можно сделать вывод, что, с одной стороны, в большинстве своем именно оптимизация генерации методом **LIME** приводила к наилучшим показателям метрики **ASR**. С другой стороны, почти все методы оптимизации имеют схожее значение метрик BERT_USE и DAN_USE, посчитанные для контроля качества в условиях заданных изначальных ограничений. Также видно, что в среднем метод генерации, суть которого заключается в замене наиболее важных слов си-

нонимами, дает наилучший показатель метрики **ASR**, когда как для других способов (ins_char, del_char, sub_char) показатель метрики ASR ниже при одинаковых значениях DAN_USE.

Также аналогично для датасета **SentiRuEval-2016-banks** было рассмотрено влияние выбора части слова, где портить символы, на итоговое качество состязательных примеров. Под **fst** подразумевается первая половина слова, **snd** - вторая половина слова, **ins** - вставка символа, **del** - удаление символа, **sub** - замена символа случайным ближайшим на клавиатуре. Результаты представлены ниже:

Таблица 6: Вставка символа в 1 слово

method	ins_fst	ins_snd
	AR/BU/DU	AR/BU/DU
alti	12/0.95/0.84	12/0.95/0.84
attention	9/0.97/0.84	9/0.97/0.84
gradient	11/0.96/0.84	11/0.96/0.84
lime	13/0.96/0.84	13/0.96/0.84
loss	14/0.96/0.84	14/0.96/0.84
shap	11/0.96/0.83	11/0.96/0.83
avg_value	11.7/0.96/0.838	11.7/0.96/0.838

Таблица 7: Вставка символа в 2 слова

method	ins_fst	ins_snd
	AR/BU/DU	AR/BU/DU
alti	19/0.92/0.81	18/0.92/0.81
attention	14/0.93/0.8	14/0.93/0.8
gradient	18/0.93/0.8	19/0.93/0.8
lime	20/0.92/0.78	20/0.92/0.78
loss	19/0.93/0.79	19/0.93/0.79
shap	15/0.92/0.79	15/0.93/0.79
avg_value	17.5/**0.925**/0.795	17.5/**0.927**/0.795

Таблица 8: Удалание символа в 1 слове

method	del_fst	del_snd
	AR/BU/DU	AR/BU/DU
alti	10/0.96/0.84	10/0.95/0.84
attention	8/0.97/0.84	8/0.97/0.84
gradient	11/0.97/0.84	11/0.97/0.85
lime	13/0.96/0.84	14/0.96/0.84
loss	13/0.96/0.84	13/0.96/0.84
shap	9/0.97/0.83	9/0.97/0.83
avg_value	**10.7**/**0.965**/**0.838**	**10.8**/**0.963**/**0.84**

Таблица 9: Удаление символа в 2 словах

method	del_fst	del_snd
	AR/BU/DU	AR/BU/DU
alti	14/0.92/0.81	14/0.92/0.81
attention	12/0.94/0.8	12/0.94/0.8
gradient	16/0.94/0.8	16/0.94/0.8
lime	19/0.93/0.78	19/0.93/0.78
loss	18/0.93/0.8	18/0.93/0.79
shap	15/0.94/0.79	14/0.94/0.79
avg_value	**15.7**/0.933/**0.797**	**15.5**/0.933/**0.795**

Таблица 10: Замена символа в 1 слове

method	sub_fst	sub_snd
	AR/BU/DU	AR/BU/DU
alti	12/0.95/0.84	12/0.95/0.84
attention	9/0.97/0.84	8/0.97/0.84
gradient	12/0.96/0.84	13/0.96/0.84
lime	15/0.96/0.84	15/0.96/0.84
loss	14/0.96/0.85	14/0.96/0.84
shap	11/0.96/0.83	11/0.96/0.83
avg_value	12.2/0.96/**0.84**	12.2/0.96/**0.838**

Таблица 11: Замена символа в 2 словах

method	sub_fst	sub_snd
	AR/BU/DU	AR/BU/DU
alti	17/0.91/0.81	17/0.91/0.81
attention	16/0.93/0.8	16/0.93/0.8
gradient	18/0.93/0.8	18/0.92/0.8
lime	21/0.92/0.78	21/0.92/0.78
loss	20/0.92/0.79	20/0.92/0.79
shap	16/0.93/0.79	16/0.93/0.79
avg_value	15.5/**0.923**/0.795	15.5/**0.922**/0.795

Исходя из полученных выше таблиц, можно сделать вывод, что искажение символов только в конкретной части слова почти что не сказывается на качестве итоговой генерации состязательных примеров.

8.4 Примеры работы методов интерпретации

Рассмотрим некоторые примеры выделения методами интерпретации наиболее важных слов (Красным цветом выделены те слова, которые тот или иной метод посчитал наиболее важным, но при этом их удаление не привело к смене прогноза модели. Зеленым же выделены слова, удаление которых изменило прогноз модели):

1. Все методы указали на одно и то же слово:
    - в мире о финансах втб сократил убыток за третий квартал на 6,2 млрд руб. рбк
    - rt минюст сша хочет заблокировать счета мтс и вымпелкома штаты хорошо так взялись за российские комп...
2. Только один метод указал на правильное слово и прогноз изменился:
    - я хорошо помню обман сбербанка в 1962 и 1961 когда сгорели все мои денюжки :: **LIME** дал правильный прогноз

- rt минюст сша хочет заблокировать счета мтс и вымпелкома штаты хорошо так взялись за российские комп... :: **Gradient** дал правильный прогноз

8.5 Примеры итоговой генерации

Рассмотрим некоторые примеры сгенерированных состязательных примеров в следующих случаях (Красным цветом выделены искажения, которые были внесены в текст. Зеленым же выделены слова, которые подверглись искажению):

1. Символьное искажение

   - (a) Было: ростелеком обеспечил бесперебойную видеотрансляцию егэ2015 из более 93 тысяч видеокамер по всей стране
     (b) Стало: ростелеком обеспкчил бесперебойную видеотрансляцпю егэ2015 из более 93 тысяч видеокамер по всей стране
   - (a) Было: rt выгодно ложить деньги на кредитную карту в россельхозбанке
     (b) Стало: rt выгодгно ложрить деньги на кредитную карту в россельхозбанке

2. Искажение слов

   - (a) Было: rt сбербанк замедлил отставание показатели банка начинают восстанавливаться
     (b) Стало: rt сбербанк задержат отставание показатели банка начинают восстанавливаться
   - (a) Было: купила новый телефон от мтс безумно довольна ценой и качеством!!! smartstart мтс
     (b) Стало: купила новый телефон от мтс безумно удовлетворена ценой и качеством!!! smartstart мтс

9 Заключение

Таким образом, в ходе работы были исследованы различные методы интерпретации для определения важных слов в задаче классификации текстов и их влияние на итоговое качество состязательных примеров. Исходя их полученных результатов можно сделать следующие выводы:

1. Улучшенный метод извлечения важности токенов из механизма self-attention для трансформеров **ALTI** во многих случаях смог повысить показатель метрики **ASR** по сравнению с методом интерпретации **Attention**, но в то же время имел меньший показатель метрики **BERT_USE**.

2. Качество **random baseline** было превышено всеми расмотренными методы интерпретации прогнозов по показателю метрики **ASR** при схожих значениях метрики **USE**.

3. В большинстве случаев **LIME** приводил к наилучшему итоговому качеству состязательных примеров для рассматриваемых методов генерации.

4. В среднем **искажение слов** приводило к лучшему значению ASR при одинаковом показателе DAN_USE по сравнению с **искажением символов**.

5. Искажение символов только **первой** или только **второй половины** слова не дает практически никакого эффекта.

Список литературы

[1] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova // *arXiv*. — 2018.

[2] *Zong, Mingyu*. a survey on GPT-3 / Mingyu Zong, Bhaskar Krishnamachari // *arXiv*. — 2022.

[3] *Madsen, A*. Post-hoc Interpretability for Neural NLP: A Survey. / A. Madsen, S. Reddy, S. Chandar // *ACM*. — 2023.

[4] *Tulio Ribeiro, Marco*. Why Should I Trust You? Explaining the Predictions of Any Classifier. / Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin // *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. — 2016.

[5] SHAP values for Explaining CNN-based Text Classification. / Wei Zhao, Tarun Joshi, N. Vijayan Nair, A. Sudjianto // *arXiv*. — 2020.

[6] Why Attentions May Not Be Interpretable? / Bing Bai, Jian Liang, Guanhua Zhang et al. // *27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. — 2021. — Pp. 25–34.

[7] Measuring the Mixing of Contextual Information in the Transformer. / Javier Ferrando, I. Gerard Gallego, Ioannis Tsiamas1, R. Marta // *EMNLP*. — 2022.

[8] *Sundararajan, Mukund*. Axiomatic Attribution for Deep Networks. / Mukund Sundararajan, Ankur Taly, Qiqi Yan // *34th International Conference on Machine Learning*. — 2017.

[9] TextBugger: Generating Adversarial Text Against Real-world Applications. / Li Jinfeng, Ji Shouling, Du Tianyu, Li Bo // *NDSS*. — 2019.

[10] *Ter-Hovhannisyan, T.* Adversarial Attacks on Language Models: WordPiece Filtration and ChatGPT Synonyms / T. Ter-Hovhannisyan, H. Aleksanyan, K. Avetisyan // *Investigations on applied mathematics and informatics*. — 2023. — Pp. 80–95.

[11] DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter / Sanh6 Victor, Lysandre Debut, Julien Chaumond, Thomas Wolf // *NeurIPS*. — 2019.

[12] *Rong, Xin.* word2vec Parameter Learning Explained / Xin Rong // *arXiv.* — 2014.

[13] *Лукашевич, Наталья.* SentiRuEval-2016: преодоление временных различий и разреженности данных для задачи анализа репутации по сообщениям Твиттера / Наталья Лукашевич, Юлия Рубцова // *International Conference "Dialogue 2016".* — 2016.

[14] Attention is all you need. / A. Vaswani, N.M. Shazeer, N. Parmar et al. // *NIPS.* — 2017.

[15] ONION: A simple and effective defense against textual backdoor attacks / F. Qi, Y. Chen, M. Li et al. // *EMNLP.* — 2021.

[16] RAP: Robustness-aware perturbations for defending against backdoor attacks on NLP models / W. Yang, Y. Lin, P. Li et al. // *EMNLP.* — 2021.

[17] Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. / D. Jin, Z. Jin, J.T. Zhou, P. Szolovits // *Proceedings of the AAAI Conference on Artificial Intelligence.* — 2020.

[18] Generating Natural Language Adversarial Examples. / M. Alzantot, Y. Sharma, A. Elgohary, B.J. Ho // *EMNLP.* — 2018.

[19] *Scott, M.* A Unified Approach to Interpreting Model Predictions. / M. Scott, Lee Su-In // *NIPS.* — 2017.

[20] Explaining How Transformers Use Context to Build Predictions. / Javier Ferrando, I. Gerard Gallego, Ioannis Tsiamas1, R. Marta // *ACL.* — 2023.