

Нижегородский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет информатики, математики и компьютерных наук

Miakov Timofey Ilch

DYNAMIC TOPIC MODELING FOR VOICE DIALOGUES

Курсовая работа

студента образовательной программы «Прикладная математика и
информатика»

Руководитель
Приглашенный преподаватель
НИУ ВШЭ-Нижний Новгород

А.А. Шимко

Нижний Новгород, 2023 год

Оглавление

Введение

Юмор - это уникальное и творческое коммуникативное поведение, проявляемое во время социальных взаимодействий. Он создается мультимодальным способом, с использованием слов (текст), жестов (зрение) и просодических сигналов (акустика). Понимание юмора с помощью этих трех модальностей укладывается в рамки мультимодального языка - недавнего направления исследований в области обработки естественного языка (NLP). Хотя распознавание юмора является признанной областью исследований в НЛП, подход к этой проблеме со стороны мультимодального глубокого обучения недостаточно изученная область.

В данной работе, будет проведено исследование по классификации видео на наличие юмора с использованием датасета UrFunny[urfunny:2019], который был создан специально для детектирования юмора с помощью разных модальностей. Кроме того, исследование будет сосредоточено на использовании архитектуры Трансформер и его модификациях, которые будут использованы для извлечения признаков из разных модальностей и предсказании классов.

Глава 1. UrFunny датасет

Авторы используемого мной датасета, собрали 1866 видеороликов, а также их стенограммы с портала TED.

TED talks - один из самых разнообразных каналов обмена идеями, как по выступающим, так и по темам. Спикеры из разных слоев общества, этнических групп и культур излагают свои мысли на широко популярном канале. Темы этих презентаций разнообразны: от научных открытий до повседневных событий. Благодаря разнообразию докладчиков и тем выступления на TED охватывают широкий спектр юмора. Таким образом, эта платформа представляет собой уникальный ресурс для изучения динамики юмора в мультимодальной среде.

Видеоролики TED включают в себя ручные расшифровки и маркеры аудитории. Транскрипции отличаются высокой надежностью, что, в свою очередь, позволяет выравнивать текст и аудио. Транскрипции также включают надежно аннотированные маркеры поведения аудитории. В частности, маркер “смех” использовался в исследованиях НЛП в качестве показателя юмора. Предыдущие исследования выявили важность, как кульминационного момента, так и контекста для понимания и моделирования юмор. В юмористическом сценарии контекст - это постепенное развитие сюжета, а кульминационный момент - внезапный поворот сюжета, вызывающий смех.

В собранных данных, с помощью маркера смеха были размечены 8257 отрывков, где присутствует юмор. из расшифровок. Контекст для этих видео извлекался из предыдущих предложений до кульминационной фразы. Используя аналогичный подход, 8257 отрывков, в которых не присутствует юмор, отбираются через случайные промежутки времени, где за последним предложением не сразу следует маркер смеха. В наборе данных наблюдается однородное 50%-ное разделение между положительными и отрицательными примерами.

Приведем немного статистических данных о датасете.

Общая продолжительность всего набора данных составляет 90,23 часа. В UR-FUNNY насчитывается в общей сложности 1741 отдельный рассказчик и 417 различных тем. В пятерку наиболее частовстречающихся тем входят технологии, наука, культура, глобальные проблемы и дизайн. Средняя продолжительность каждого экземпляра данных составляет 19,67 секунды, при этом

контекст в среднем из 14,7 сек. и кульминационный момент в среднем за 4,97 сек. Среднее количество слов в заголовке составляет 16,14, а среднее количество слов в контекстных предложениях - 14,80.

Глава 2. Теоретическая часть

Мультимодальное глубокое обучение — это сфера машинного обучения, целью которого является обучение моделей искусственного интеллекта обрабатывать и находить взаимосвязи между различными типами данных (модальностями) - как правило, изображениями или видео, аудио и текстом. Комбинируя различные модальности, модель глубокого обучения может более универсально воспринимать свою среду, поскольку некоторые сигналы существуют только в определенных модальностях.

Данный раздел представит архитектуру Трансформера, который изначально придуман для задач в области НЛП. А также её модификации для обработки текста (RoBERTa), видео (Vision Transformer) и звука (Wave2Vec), которые будут использоваться в практической части работы.

2.1. NLP до Трансформера

Для решения задач в сфере обработки естественного языка до определенного момента широко применялись рекуррентные нейронные сети (RNN). Основной архитектурой в таких задачах была модель encoder-decoder[seq2seq:2014] (рис А), в которой каждый из блоков представлял из себя RNN. На вход энкодера подавались последовательно токены, после чего на выход получался эмбединг, который в свою очередь подавался декодеру. Декодер, используя токены изначального предложения, а также переданное представление от кодировщика, генерирует целевую последовательность.

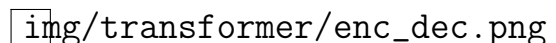
img/transformer/enc_dec.png

Рис. 2.1. Архитектура encoder-decoder

Но у такой архитектуры были некоторые проблемы. Во-первых, представление текста сжималось в один вектор, чаще всего это из-за многообразия возможных входных данных и ограниченности вектора терялась важная информация. Во-вторых, декодеру могла понадобиться разная информация от энкодера на разных этапах генерации, но декодер видит только одно представление источника.

Чтобы исправить эти проблемы, представили механизм внимания (attention mechanism[attention:2014]).

2.2. Механизм внимания - первый шаг к Трансформеру

Механизм внимания является частью нейронной сети. На каждом этапе декодирования он решает, какие части входных данных важнее всего. При этом кодировщику не нужно сжимать весь исходный код в один вектор - он создает представления для всех токенов источника.

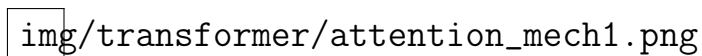
img/transformer/attention_mech1.png

Рис. 2.2. Схема работы механизма внимания

Алгоритм работы механизма внимания в encoder-decoder модели приведен ниже.

- На каждом этапе декодирования механизм внимания получает на вход: состояние декодера h_t и все состояния кодировщика s_1, s_2, \dots, s_m
- Вычисляет коэффициенты внимания (attention scores) для каждого состояния кодировщика s_k механизм внимания вычисляет свою "релевантность" для этого состояния декодера h_t .

Для вычисления можно применить любую функцию, однако, есть несколько популярных и простых вариантов, которые работают довольно хорошо.

- $scores(h_t, s_k) = h_t^T \cdot s_k$
- $scores(h_t, s_k) = h_t^T \cdot W \cdot s_k$, где W - матрица (обучаемый параметр)
- $scores(h_t, s_k) = w_2^T \cdot \tanh(W \cdot [h_t^T, s_k])$, где W, w_2 - матрицы (обучаемые параметры), а $[h_t^T, s_k]$ означает конкатенацию векторов

- Вычисляет веса внимания (attention weights): распределение вероятностей - softmax, применяемое к показателям внимания;
- Вычисляет вывод внимания : взвешенная сумма состояний декодера с весами внимания.

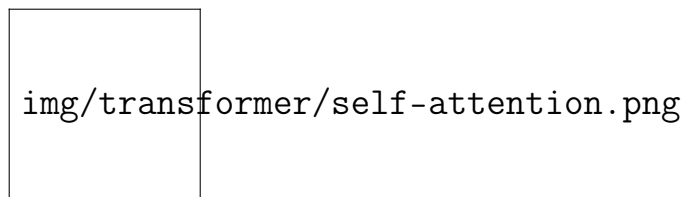


Рис. 2.3. Схема работы self-attention механизма

2.2.1. Self-attention

Self-attention - является одним из ключевых компонентов модели, в которой токены взаимодействуют друг с другом. На каждом шаге в энкодере токены обмениваются информацией между собой, собирает контекст и обновляет предыдущее представление себя (рис 2.3). Такой подход полностью исключает последовательную обработку, как в RNN, что ускоряет работу нейронной сети и дает возможность обрабатывать каждое слово параллельно.

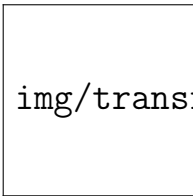
2.2.2. Query, Key, Value

Каждый входной токен в self-attention получает три представления, соответствующие ролям, которые он может играть:

- query - запрашивает информацию;
- key - показывает, какая информация есть у него;
- value - дает запрашиваемую информацию;

От слова, которое мы сейчас подаем на вход (центральное), требуется query вектор, этим вектором мы запрашиваем у других информацию о текущем слове в данном контексте. От каждого другого слова нам будет нужен вектор key, который показывает, какую информацию данное слово может сказать о смысле центрального слова, а также вектор value, из которого мы и получим нужную нам информацию.

Получив информацию от каждого вектора, мы подаем это на вход softmax функции и получаем итоговый эмбединг центрального слова. На рисунке ниже представлен полный алгоритм взаимодействия Q, K, V в self-attention механизме.



img/transformer/query-key-value.png

Рис. 2.4. Query, Key, Value в self-attention механизма

Формула для вычисления выходных данных внимания следующая:

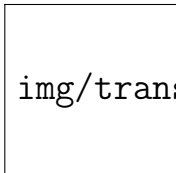
$$Attention(q, k, v) = softmax(\frac{q \cdot k^T}{\sqrt{d_k}}) \cdot v$$

2.2.3. Masked self-attention

Для энкодера и декодера используются разные способы получения токенов, если энкодер получает все токены сразу и каждый токен может смотреть на любой другой токен во входном предложении, в декодере, мы генерируем по одному токеноу за раз: во время генерации мы не знаем, какие токены мы будем генерировать в дальнейшем. Чтобы запретить декодеру заглядывать вперед, модель использует masked-self attention, который видит токены, только стоящие до него во входной последовательности.

Работа masked self-attention отличается во время обучения и инференса модели. Во время второго декодер не заглядывает вперед - так как, мы не знаем, что будет дальше. Но при обучении используются уже известные нам данные. Поэтому при обучении на вход декодеру подается все целевое предложение целиком - без масок.

На рисунке ниже приведен алгоритм работы masked self-attention механизма.



img/transformer/masked self-attention.png

Рис. 2.5. Схема работы masked Self-attention механизма

2.2.4. Multi-Head Attention

Обычно понимание роли слова в предложении требует понимания того, как оно связано с разными частями предложения. Это важно не только при обработке исходного предложения, но и при генерации целевого. Поэтому мы

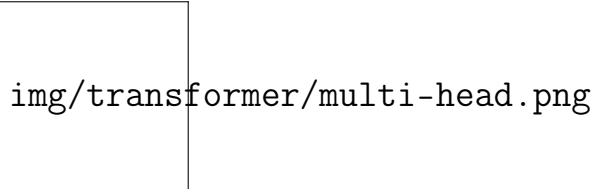


Рис. 2.6. Схема Multi-head attention

должны позволить модели сосредоточиться на разных вещах: для этого сделали multi-head attention. Вместо одного attention механизма multi-head attention имеет несколько «голов», которые работают независимо.

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, head_2, \dots, head_n) \\ head_i &= Attention(QW_Q^i, KW_K^i, VW_V^i) \end{aligned}$$

Формально это реализовано в виде нескольких механизмов внимания, результаты которых объединяются. (рис 2.6)

2.3. Трансформер: Архитектура модель

Трансформер - это модель, основанная только на механизме внимания и оказавшая огромное влияние на NLP в частности и машинном обучении в целом. Впервые представленная в статье Attention is All You Need[allyouneed:2017] в 2017 году. Модель сразу показала лучшие результаты в задаче перевода, в сравнении с encoder-decoder модели.

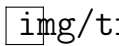
img/transformer/transformer.png

Рис. 2.7. Схема архитектуры Трансформер

Трансформер состоит из частей, которые ранее описывались. Encoder состоит из multi-head self-attention механизмов и генерирует представление слов, а decoder из masked self-attention механизма, который уже генерирует новую последовательность. Это происходит в несколько слоев, обычно 6.

Теперь рассмотрим другие составляющие архитектуры:

2.3.1. Feed forward and Add & Norm

Кроме блоков внимания, в Трансформере также присутствуют feed forward слои, которые состоят из линейного слоя и функции активации ReLU.

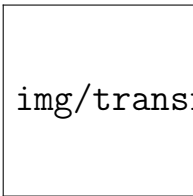
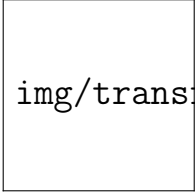
img/transformer/feed_forward.png

Рис. 2.8. Feed-forward слой

Вторым важным слоем является LayerNorm, который независимо нормализует векторное представление каждой последовательности в батче. В Трансформере нормализуются векторное представление каждого токена. Кроме того, LayerNorm имеет обучаемые параметры, *scale* и *bias*, которые используются после нормализации для масштабирования выходных данных слоя.

На рис 2.8 стоит обратить внимание на то, что μ_k и σ_k оцениваются для каждой последовательности, в отличие от *scale* и *bias*, которые являются параметрами слоя.

A diagram showing a square box with the text "img/transformer/layer_norm.png" inside it.

img/transformer/layer_norm.png

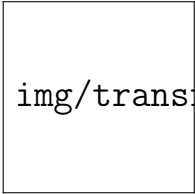
Рис. 2.9. LayerNorm слой

2.3.2. Positional encoding

Поскольку Transformer не знает порядок входных токенов, т.к. все токены обрабатываются одновременно, поэтому мы должны явно показать модели положения токенов. Для этого у нас есть два набора вложений: для самих токенов и для их позиций. Тогда входное представление токена является суммой двух вложений. Positional encodings можно обучить, но авторы оригинальной статьи обнаружили, что наличие исправленных вложений не ухудшает качество модели. Итоговая формула для позиционных вложений выглядит так:

$$PE_{pos,2i} = \sin(pos/10000^{2i/d_{model}}),$$
$$PE_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}}),$$

где pos это позиция и i является векторным измерением. Каждое измерение позиционного кодирования соответствует синусоиде, а длины волн образуют геометрическую прогрессию от 2π до $10000 \cdot 2\pi$.

A diagram showing a square box with the text "img/transformer/pos_encoding.png" inside it.

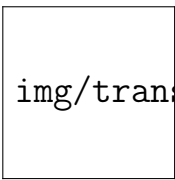
img/transformer/pos_encoding.png

Рис. 2.10. Positional encoding

2.3.3. Residual connection

Как и в сверточных сетях, в Трансформере применяется residual connections. Смысл residual connection очень прост, добавляет входные данные к выходным данным модели, но в то же время это очень полезная вещь: он борется с проблемой затухания градиента по сети и позволяют использовать много слоев.

В трансформаторе остаточные соединения используются после каждого блока внимания и полносвязного блока.



img/transformer/res_block.png

Рис. 2.11. Residual connection

2.4. Модель TimeSformer

2.4.1. Новый подход к решению задач компьютерного зрения

Сверточные нейронные сети (CNN) являлись базовой архитектурой для задач распознавания изображений и показывали лучшие результаты во всех бенчмарках. Базовым слоем таких сетей являлись свертки, которые обрабатывали все изображения сразу.

Однако в 2016 году Google опубликовали статью, в которой представили новую нейронную сеть для работы с изображениями - Vision Transformer[**vit:2016**] (далее ViT). Основным отличием от других нейронных сетей для обработки изображений стало использование архитектуры Transformer вместо сверток.

Ключевая идея, лежащая в основе ViT, заключается в обработке изображения как набор патчей (patches), а затем использовании модели на основе Transformer для обработки этих патчей. Участки сначала сглаживаются в набор векторов, которые затем обрабатываются. Это позволяет модели находить и запоминать долгосрочные зависимости между различными патчами, что потенциально приводит к повышению качества модели по сравнению с традиционными CNN.

Так как основной темой работы, являются видео, основной моделью стал не ViT, а еще более новая модель, разработанная Facebook, которая называется TimeSFormer. TimeSformer[**timesformer:2021**] адаптирует стандартную архитектуру Трансформера и приемы, используемые в ViT, к видео, позволяя изучать пространственно-временные характеристики непосредственно из последовательности исправлений на уровне кадров.

2.4.2. Обработка входного изображения

На вход в модель подается разбитое на N частей изображение. Т.к. Трансформер принимает на вход 1D-вектора, части изображения с помощью

линейно проекции ”вытягивают” из 2D в 1D. Также как и в задачах обработки текста, на вход в модель кроме патчей надо подать positional encoders. Итоговые преобразования изображения выглядят так:

$$z_0 = [x_{class}; x_p^1 \cdot E; x_p^2 \cdot E; \dots x_p^N \cdot E] + E_{pos},$$

$x \in R^{H \times W \times C}$, $E \in R^{N \times (P^2 \times C)}$, где C - кол-во каналов, (P, P) - размер каждого патча, $N = HW/P^2$

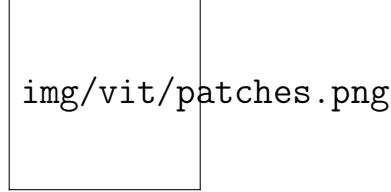


Рис. 2.12. Подготовка патчей

В качестве альтернативы патчам изображения, входная последовательность может быть сформирована из карт характеристик (feature map) CNN. В этой гибридной модели проекция E применяется к участкам, извлеченным из карты объектов CNN, как частный случай, патчи могут иметь пространственный размер 1×1 . Патч с классом изображения и positional embeddings добавляются, как описано выше.

2.4.3. Query-Key-Value вычисления

TimeSFormer состоит из L блоков энкодера. В каждом блоке l , как и в оригинальном трансформере вычисляются Query, Key, Value для каждого патча из его прошлого состояния $z_{(p,t)}^{(l-1)}$

$$\begin{aligned} q_{(p,t)}^{(l,a)} &= W_Q^{(l,a)} \cdot LN(z_{(p,t)}^{(l-1)}) \in R^{D_h}, \\ k_{(p,t)}^{(l,a)} &= W_K^{(l,a)} \cdot LN(z_{(p,t)}^{(l-1)}) \in R^{D_h}, \\ v_{(p,t)}^{(l,a)} &= W_V^{(l,a)} \cdot LN(z_{(p,t)}^{(l-1)}) \in R^{D_h}, \end{aligned}$$

где $LN()$ означате LayerNorm, $a = 1, \dots, A$ - индексы голов в Multi-head attention.

2.4.4. Attention weights

Вычисление self-attention коэффициентов для $a_{(p,t)}^{(l,a)}$ для патча (p, t) вычисляется как:

$$a_{(p,t)}^{(l,a)} = SoftMax\left(\frac{q_{(p,t)}^{(l,a)T}}{\sqrt{D_h}} \cdot \left[k_{(0,0)}^{(l,a)} k_{(p',t')}^{(l,a)}\right]_{p'=1,\dots,N,t'=1,\dots,F}\right)$$

2.4.5. Encoding

Закодированное представление $z_{(p,t)}^{(l)}$ в блоке l получается путем вычисления взвешенной суммы векторов значений с использованием коэффициентов self-attention от каждой головы multi-head attention:

$$s_{(p,t),(0,0)}^{(l,a)} = a_{(p,t),(0,0)}^{(l,a)} \cdot v_{(0,0)} + \sum_{p'=1}^N \sum_{t'=1}^F a_{(p,t),(p',t')}^{(l,a)} \cdot a_{(p',t')}^{(l,a)}$$

Затем векторы каждого головы конкатенируются, проецируются и передается через MLP, используя residual connections после каждой операции:

$$z'_{(p,t)}^{(l)} = W_O \cdot \begin{bmatrix} s_{(p,t)}^{(l,1)} \\ \vdots \\ s_{(p,t)}^{(l,A)} \end{bmatrix} + z_{(p,t)}^{(l-1)}$$

$$z_{(p,t)}^{(l)} = MLP(LN(z'_{(p,t)}^{(l)})) + z'_{(p,t)}^{(l)}$$

Для финальной классификации используется MLP с 1 скрытым слоем.

Ниже представлен блок энкодера в TimeSFormer:

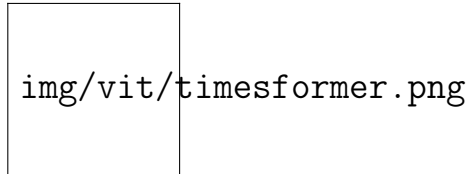


Рис. 2.13. Энкодер блок в TimeSFormer

2.5. Модель RoBERTa

2.5.1. BERT

BERT был представлен в статье BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[bert:2018]. Архитектура модели BERT - это энкодер Transformer. По новому подошли к процессу обучения и к тому, как BERT используется для последующих задач.

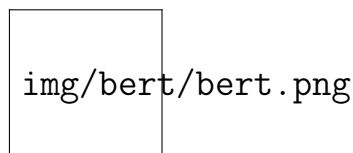


Рис. 2.14. Архитектура BERT

Новый способ обучения

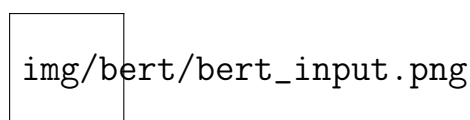


Рис. 2.15. Входные данные

Входные данные для обучения: пары предложений со разделенных специальным токеном-разделителем [SEP] (рис 2.15). Чтобы модель могла легко различать эти предложения, в дополнение к токеновым и позиционным вложениям она использует вложения сегментов. Также BERT имеет 2 цели обучения, для которых используются некоторые специальные токены([MASK], [CLS]).

Первая цель обучения: Masked Language Modeling (MLM)

Наиболее важной из двух целей обучения является цель моделирования замаскированного языка (MLM). На этапе обучения для MLM происходит следующее:

- Выбирается несколько токенов (каждый токен выбирается с вероятностью 15%)

- Выбранные токены заменяются на [MASK] с вероятности 80%, случайный токен - 15% и остается без изменений - 10%
- Модель должна предсказать исходный токен

На рисунке ниже показан пример шага обучения для одного предложения.

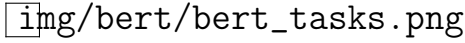
img/bert/bert_tasks.png

Рис. 2.16. Masked Language Modeling (MLM)

Вторая цель обучения: Next Sentence Prediction (NSP)

Задача «Предсказание следующего предложения» (NSP) представляет собой задачу бинарной классификации. На последнем уровне модели, исходя из токена [CLS], модель предсказывает, являются ли два предложения последовательными предложениями в некотором тексте или нет. В датасете для обучения, представленном в оригинальной статье сказано, что при обучении 50% примеров содержат последовательные предложения, извлеченные из обучающих текстов, а еще 50% — случайную пару предложений.

Это задание учит модель понимать отношения между предложениями и позволяет BERT решать более сложные задачи.

Пример из исходного датасета:

Ввод: [CLS] мужчина пошел в [MASK] магазин [SEP] он купил галлон [MASK] молока [SEP]

Метка: Последовательные

Ввод: [CLS] мужчина пошел в магазин [MASK] [SEP] пингвин [MASK] летают ##less birds [SEP]

Метка: Непоследовательные

Следует также упомянуть, что BERT подходит для решения разных задач, ниже приведены основные типы задач:

- **Классификация отдельных предложений** Для классификации отдельного предложения, вводятся токены предложения и [CLS] с классом предложения.
- **Классификация пар предложений** Для классификации пары предложений, вводятся токены предложений, как в обучении, и [CLS] с классом предложения.
- **Ответ на вопрос (Question answering)** В задачах вопросов на ответы, на вход дают текст и вопрос. Ответ на этот вопрос содержится в исходном тексте, и задача состоит в том, чтобы найти отрывок из текста с ответом.

- Тэгирование слов в предложении В задачах тегирования прогнозируются теги для каждого токена. Например, в распознавании именованных объектов (NER) вы должны предсказать, является ли слово именованным объектом и его типом (например, местоположение, человек и т. д.).

2.5.2. RoBERTa - улучшенная версия BERT

Модель RoBERTa была предложена в книге RoBERTa: A Robustly Optimized BERT Pretraining Approach[**roberta:2019**]. Он основан на модели Google BERT, выпущенной в 2018 году. Она основан на BERT и изменяет ключевые гиперпараметры, удаляя цель предварительного обучения в следующем предложении и обучая с гораздо большими мини-батчами и learning rate.

Список основных изменений представлен ниже:

- Обучение большем объем данных (в 10 раз больше) и дольше по времени.
- Входные данные представляют собой более длинные последовательности, при оставшемся ограничении в 512 токенов, как и у BERT
- Динамическое маскирование токенов: токены маскируются по-разному в каждую эпоху, тогда как BERT делает это раз и навсегда
- Использование большого батча (относительно исходного BERT с батчем 256) при обучении модели
- Использование Byte-Pair Encoding с байтами в качестве субъединицы, а не с символами (из-за символов юникода)

2.6. Модель Wave2vec 2.0

Фреймворк wave2vec 2.0[w2v:2020] разработан для self-supervised обучения эмбеддингов на основе необработанных аудиоданных. Модель кодирует речевой звук с помощью многослойной сверточной нейронной сети, а затем маскирует участки результирующих скрытых речевых представлений, аналогично моделированию замаскированного языка (MLM), который был разобран в разделе про BERT. Скрытые представления передаются в Трансформер для построения контекстуализированных эмбеддингов, и модель обучается с помощью contrastive learning.

Как показано на рисунке ниже, модель обучается в два этапа. Первый этап проходит в режиме самоконтроля, который выполняется с использованием немаркированных данных и направлен на достижение наилучшего возможного представления речи. Вы можете думать об этом так же, как вы думаете о встраивании слов. Эмбеддинги слов также направлено на достижение наилучшего представления естественного языка. Основное отличие заключается в том, что Wav2Vec 2.0 обрабатывает аудио вместо текста.

Вторая фаза обучения - это контролируемая тонкая настройка, в ходе которой помеченные данные используются для обучения модели предсказывать определенные слова или фонемы (наименьшая возможная звуковая единица в определенном языке).

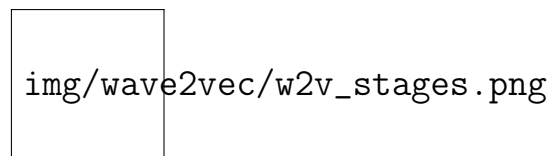


Рис. 2.17. Этапы обучения wave2vec

Архитектура окончательной модели, используемой для прогнозирования, состоит из трех основных частей:

- Сверточные слои, которые обрабатывают входные данные необработанной формы сигнала для получения скрытого представления - Z ,
- Трансформер из n слоев, создающий эмбеддинги с контекстом - C ,
- линейная проекция на выходные данные - Y

2.6.1. Контрастивное обучение

Контрастивное обучение - это концепция, в рамках которой входные данные преобразуются двумя различными способами. После этого модель обучается распознавать, являются ли два преобразования входных данных по-прежнему одним и тем же объектом. В Wav2Vec 2.0 трансформаторные слои являются первым способом преобразования, второй осуществляется путем квантования, что будет объяснено в дальнейшей части этой статьи. Более формально, для замаскированного скрытого представления z_t мы хотели бы получить такое контекстное представление s_t , чтобы иметь возможность угадать правильное квантованное представление q_t среди других квантованных представлений.

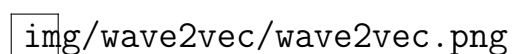


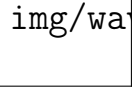
Рис. 2.18. Входные данные

2.6.2. Квантизация

Квантизация - это процесс преобразования значений из непрерывного пространства в конечный набор значений в дискретном пространстве.

У нас есть вектор представления скрытой речи z_t охватывает несколько фонем. Число фонем в языке конечно. Более того, число всех возможных пар фонем конечно. Это означает, что они могут быть идеально представлены одним и тем же скрытым речевым представлением. Кроме того, их число конечно, поэтому мы можем создать "кодovou книгу содержащий все возможные пары фонем. Затем квантование сводится к выбору правильного кодового слова из "кодovou книгу". Однако вы можете себе представить, что количество всевозможных звуков огромно. Чтобы упростить обучение и использование, авторы Wav2Vec 2.0 создали G кодовых книг, каждая из которых состоит из V кодовых слов. Чтобы создать квантизированное представление, следует выбрать лучшее слово из каждой кодовой книги. Затем выбранные векторы объединяются и обрабатываются линейным преобразованием для получения квантизированное представления.

Выбор правильного кодового представления выбирается с помощью Gumbel softmax:



img/wave2vec/w2v_books.png

Рис. 2.19. Выбор токена из кодовых книг

$$p_{g,v} = \frac{\exp \frac{\text{sim}(l_{g,v} + n_v)}{\tau}}{\sum_{k=1}^V \exp \frac{l_{g,k} + n_k}{\tau}}, \text{ где}$$

- sim — косинусное сходство,
- l — логиты, полученные свертками,
- $n_k = -\log(-\log(g_l))$,
- u_k - выборки из равномерного распределения $U(0, 1)$,
- τ - температура

Поскольку это задача классификации, функция softmax представляется естественным выбором для выбора наилучшего кодового слова в каждой кодовой книге. Gumbel softmax поставляется с двумя улучшениями: рандомизацией и температурой τ . Благодаря рандомизации модель более охотно выбирает разные кодовые слова во время обучения, а затем обновляет их веса. Важно, особенно в начале обучения, не допускать использования только подмножества кодовых книг.

2.6.3. Diversity loss (Потеря разнообразия)

Потеря разнообразия - это своего рода метод регуляризации. Авторы установили $G=2$ кодовых книги с $V=320$ кодовыми словами в каждой кодовой книге. Теоретически это дает $320 \cdot 320 = 102400$ возможных квантованных представлений. Но в процессе обучения можно столкнуться с проблемой того, что модель использовать все эти возможности. Она научится использовать, например, только 100 кодовых слов из каждой кодовой книги и растратит весь потенциал кодовой книги впустую. Вот почему потеря разнообразия может быть полезной. Он основан на энтропии, которая может быть вычислена по следующей формуле:

$$H(x) = - \sum_x P(x) \cdot \log(P(x)), \text{ где}$$

x — возможный результат действия дискретной случайной величины X ,
 $P(x)$ — вероятность события x .

Энтропия принимает максимальное значение, когда распределение данных является равномерным. В случае wave2vec это означает, что все кодовые слова используются с одинаковой частотой. С помощью этого производится вычисление энтропии каждой кодовой книги по всей серии обучающих примеров, чтобы проверить, используются ли кодовые слова с одинаковой частотой. Максимизация этой энтропии побудит модель использовать преимущества всех кодовых слов. Максимизация равна минимизации отрицательной энтропии, которая является потерей разнообразия.

$$L_d = \frac{1}{GV} \cdot (-H(\bar{p}_g)) = \frac{1}{GV} \sum_{g=1}^G \sum_{v=1}^V \bar{p}_{g,v} \cdot \log(\bar{p}_{g,v})$$

Глава 3. Практическая часть

Основной задачей в работе будет классификация видео на наличие юмора с использованием разных моделей для разных модальностей. Для работы с разными модальностями будут использованы 3 модели, которые были описаны в теоретической части:

- Wave2Vec - извлечение аудио признаков,
- BERT - извлечение текстовых признаков,
- TimeSFormer - извлечение признаков из видео фрагментов,

Сначала модели попробуют предсказать классы в своих модальностях, после чего их предсказания будут обработаны вместе для улучшения результатов.

3.1. Классификация по одной модальности

Для извлечения признаков, как уже было описано, будут использоваться 3 различные модели. В силу ограниченных вычислительных мощностей, будут взяты предобученные модели с Hugging Face и дообучены на наших данных.

Так как у нас равное кол-во наблюдений положительного и отрицательного классов, делим выборку 70% на дообучение и 30% валидацию с одинаковым кол-вом примеров разных классов в обеих подвыборках.

Для каждой модели начнем считать качество модели по 3 метрикам: Accuracy (из-за отличной балансировки, можем использовать обычную accuracy), Precision, Recall.

- Для работы с текстом будем использовать RoBERTa из библиотеки моделей HuggingFace: roberta-base.
- Для работы с аудио будем использовать Wave2vec2 из библиотеки моделей HuggingFace: wav2vec2-base.
- Для работы с видео будем использовать Wave2vec2 из библиотеки моделей HuggingFace: timesformer-base.

С помощью предлагаемых Hugging Face инструментов, сделаем дообучение моделей под нашу задачу.

3.1.1. Результаты

Далее в таблицах с результатами модальности будут называться в сокращенном виде: Текст - Т, Аудио - А, Видео - В.

Результаты классификации по одной модальности:

| Классификация по тексту | | | |
|-------------------------|----------|-----------|---------|
| Модальность | Accuracy | Precision | Recall |
| Т | 0.71377 | 0.70563 | 0.75064 |
| В | 0.47836 | 0.50600 | 0.59729 |
| А | 0.50885 | 0.57865 | 0.78420 |

TimeSFormer показывает достаточно низкие значения метрик, относительно других моделей. Такие результаты связаны со спецификой задачи, модель не фокусируется только на человеке, но на всем видеофрагменте. Возможным решением станет использование других моделей или инструментов, которые будут обрабатывать именно человека на видео.

Wave2vec показывает хороший recall (т.е модель хорошо угадывает, класс с юмором), но accuracy все еще плохой.

По результатам можем сделать вывод, что самым подходящей для детекции юмора оказалась текстовая модальность, с которой работала RoBERTa. Действительно по тексту легче всего понять, есть ли шутка, потому что не надо обрабатывать много второстепенной информации, как в видео, и "поведение" голоса разных людей во время шутки, как на аудио.

3.2. Классификация видео с использованием нескольких модальностей

Теперь попробуем собирать вместе голосование моделей в разных модальностях, сначала будем брать все комбинации 2 модальностей, а потом возьмем 3 модальности.

Подсчет результатов будем производить двумя разными способами:

- Посчитать среднюю вероятность
- Выбирать голосованием, каждой модели за класс

3.2.1. Результаты

Результаты экспериментов с подсчетом средней вероятности:

| Модальности | Accuracy | Precision | Recall |
|-------------|----------|-----------|---------|
| A + B | 0.49737 | 0.50512 | 0.60373 |
| A + T | 0.71377 | 0.70488 | 0.75257 |
| B + T | 0.63573 | 0.63569 | 0.66559 |
| A + B + T | 0.63081 | 0.62862 | 0.67074 |

В голосованиях в парах и тройке результаты лучше, чем у A и B по одиночке. A + T слегка улучшает результат RoBERTa в одиночку, а в остальных парах и тройке, результаты улучшаются, но не достигают RoBERTa.

При спорных ситуациях в голосовании по классам, решающий голос за модальностями с лучшим результатом в прошлой частию (т.е. TimeSFormer имеет самый слабый голос, затем Wave2Vec и самый сильный голос у RoBERTa).

Результаты экспериментов с голосованием по классам:

| Модальности | Accuracy | Precision | Recall |
|-------------|----------|-----------|---------|
| A + B | 0.49737 | 0.50512 | 0.60373 |
| A + T | 0.71377 | 0.70563 | 0.75064 |
| B + T | 0.71377 | 0.70563 | 0.75064 |
| A + B + T | 0.50885 | 0.50885 | 1.0 |

Здесь можно заметить, т.к. текст имеет решающий голос, то в парах с текстом результаты одинаковые. В A + B + T интересно заметить, что recall равен 1.0. То есть наша "тройка" относит все сэмплы к одному классу.

Глава 4. Заключение

В ходе проделанной работы были прочитаны статьи в разделе нейронных сетей, изучен механизм внимания и архитектура Трансформер с теоретической точки зрения. Также изучены модели, в основе которых лежит Трансформер: RoBerta, используемая для работы с текстовой модальностью, Wave2vec2, используемая для работы с звуковой модальностью, и TimeSFormer, используемая для работы с визуальной модальностью.

Далее были проведены эксперименты на датасете UrFunnny с каждой из моделей. Изучив результаты можно увидеть, что RoBERTa показывает самые хорошие результаты. Предположения, почему получились такие результаты было сделано в пункте 3.1.1 под таблицей с результатами.

Что касается совмещения результатов моделей (голосование по вероятностям и финальным лэйблам), аудио и текст вместе улучшают результат RoBERTa в одиночку.

Для работы с моделями, обучение и инференс, был изучен фреймворк HuggingFace, имеющий большую популярность в сфере глубокого обучения. Кроме того были использованы язык программирования Python, PyTorch - бек-энд для нейронных сетей, а также вспомогательные библиотеки librosa (работа со звуком) и Numpy.

Следующим шагом в исследовании детекции юмора может стать объединением моделей в одну, на примере CLIP от openAI или улучшение моделей в аудио и видео модальностях.

Библиографический список