

ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ «ШАГ В БУДУЩЕЕ, МОСКВА»

регистрационный номер

Инженерное дело

название факультета

ИУ7 - "Выставка-конкурс программных разработок"

название кафедры

Анализ тональности текста относительно аспектов

название работы

Автор: Храмченков Артур Владимирович
ГБОУ Инженерная школа №1581, 10 О

Научный руководитель: Смирнова Светлана Юрьевна
ГБОУ Инженерная школа №1581
Учитель информатики

Москва – 2022

**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ «ИНЖЕНЕРНАЯ ШКОЛА № 1581»**

СЕКЦИЯ: ИУ7 - "Выставка-конкурс программных разработок"

**АНАЛИЗ ТОНАЛЬНОСТИ ТЕКСТА ОТНОСИТЕЛЬНО
АСПЕКТОВ**

Автор: Храмченков Артур Владимирович
ГБОУ Инженерная школа №1581, 10 О

Научный руководитель: Смирнова Светлана Юрьевна
ГБОУ Инженерная школа №1581

Москва - 2022

Анализ тональности текста относительно аспектов.

Автор: Храмченков Артур Владимирович, ГБОУ 1581, Большой
Полуярославский переулок, дом 7 , 10 “О”.

Научный руководитель: Смирнова С.Ю., преподаватель информатики, ГБОУ
1581.

Аннотация

В настоящее время сильно увеличилось количество пользовательского контента на онлайн ресурсах(блогах, форумах, сайтах компаний и т.д.), в связи с этим увеличилось количество мнений о товарах или услугах. Например в бизнесе очень важно знать, что клиенты думают о каком-либо товаре или услуге. Часто количество таких мнений достигает больших чисел, в таких ситуациях помогает анализ тональности и анализ мнений. Такой анализ помогает получить общее мнение людей о каком-либо предмете.

Системы сентиментального анализа текста на основе аспектов на вход получают текст(например отзыв о продукте или сообщение из социальной сети) в котором выражается мнение о каком то предмете. Система определяет главные аспекты(предметы) и оценивает среднее настроение для каждого аспекта.

Цель работы: разработка и создание обучаемого алгоритма, для автоматического анализа тональности текста(позитивная, нейтральная, негативная) относительно аспектов(тема/предмета, о котором идёт речь в тексте).

Задачи:

1. Разработка и создание модели для извлечения аспектов, на основе АВАЕ модели.
2. Тренировка и оценки модели по извлечению аспектов.
3. Выбор модели для выявления тональности.
4. Внедрение обеих моделей в систему и создание пользовательского интерфейса.

Методы: Сбор и классификация данных(отзывов), Проектирование и разработка (модели аспектов), Анализ.

Материал: Python3, PyCharm(Среда разработки), Библиотеки Python(NumPy, TensorFlow и другие), научная работа An Unsupervised Neural Attention Model for Aspect Extraction. / Ruidan He

Результаты и вывод: Была разработана, натренирована и оценена модель для извлечения аспектов, создан пользовательский интерфейс для работы с системой, объединяющей обе модели.

СОДЕРЖАНИЕ:

1. ВВЕДЕНИЕ	5
2. ВЫБОР МОДЕЛЕЙ. ОПИСАНИЕ СУЩЕСТВУЮЩИХ СИСТЕМ	7
3. ОПИСАНИЕ СОЗДАНИЯ МОДЕЛЕЙ	8
3.1 Модель извлечения аспектов (АВАЕ).....	8
3.2 Модель выявления тональности	14
4. ОЦЕНКА МОДЕЛИ И СОЗДАНИЕ ИТОГОВОГО ПРОДУКТА.....	14
4.1 Тренировка и оценка АВАЕ модели	14
4.2 Создание пользовательского интерфейса.....	17
5. ЗАКЛЮЧЕНИЕ	18
6. СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ	18
ПРИЛОЖЕНИЕ	19

1. ВВЕДЕНИЕ

Анализ тональности текста - это класс методов, используемых в обработке естественного языка, анализе текста и компьютерной лингвистике. Этот класс методом предназначен для выявления в текстах эмоциональной окраски, эмоциональной оценки автора и окрашенной лексики. Данная задача может выполняться вручную (с помощью экспертов) и автоматически. Когда количество отзывов достигает тысячи и десятки тысяч это не эффективно для компаний, бизнесов или брэндов нанимать экспертов, анализирующих такое количество текста, поэтому создаются программы для автоматизированного анализа тональности текста, именно такая программа является конечным продуктом данной работы. Основные виды анализа тональности текста: классификация по бинарной шкале, классификация по многополосной шкале, системы шкалирования, субъективность/объективность.

Все виды, кроме последнего, подробно изучены и разобраны, существуют множество систем выполняющих анализ тональности текста данных типов. Но вид “субъективность/объективность” является менее изученным, более сложным и полезным. Главной задачей в данном типе является извлечение аспектов из отзывов. В настоящее время существует ограниченное количество программ, выполняющих данную задачу. Один из проектов, выполняющий задачу данного вида: (<https://pypi.org/project/aspect-based-sentiment-analysis/>). Этот API может выявлять тональность, относительно данных пользователем аспектов в данном тексте. Также API определяют тональность по бинарной шкале. Существуют модели основанные на этом API, выполняющие схожие задачи: (<https://arxiv.org/abs/1912.07976>), (<https://arxiv.org/abs/1908.11860>), (<https://arxiv.org/pdf/2001.11316.pdf>). Все выше перечисленные модели определяют термины аспектов.

В данном проекте будет создана модель, классифицирующая отзывы на категории аспектов. Учитывая заранее определённый набор категорий аспектов (например, цена, продукты питания), определяются категории аспектов, обсуждаемые в данном отзыве. Категории аспектов, как правило, более грубые, чем термины аспектов, и они не обязательно встречаются в качестве терминов в данном предложении. Например, в одном из набора данных даны следующие категории аспектов: {Еда, Атмосфера, Персонал, Цены, Разное} “Ресторан был слишком дорогим” → {Цены}. Далее определяется настроение отзыва, в данном проекте использована модель, определяющая тональность по 5-ти бальной шкале, где 1-максимально негативная тональность, а 5-максимально позитивная. После выявления тональности → {Цены, 2}. Для тренировки модели извлечения аспектов были взяты 2 набора данных: (<https://www.cs.cmu.edu/~mehrbor/RR/>), (<https://data.world/socialmediadata/beeradvocate>), также в перспективе, модель будет натренирована на наборе данных в сфере ноутбуков.

Итоговым продуктом является программа, содержащая графический пользовательский интерфейс для работы с обеими моделями (модель аспектов, модель тональности). Программа может принимать два вида входных данных:

отзыв и набор отзывов в формате .txt. В первом случае программа выводит аспект и тональность отзыва, а во втором, программа создаёт файл, с анализом предоставленного набора данных, расширения .xlsx, который можно скачать. Выходной файл содержит таблицу, в которой строка это аспект, а столбец это тональность (1-5), пересечение столбцов и строк показывает количество отзывов, относящихся к конкретному аспекту и тональности.

Преимущество данной работы заключается в том, что тональность определяется по 5-ти бальной шкале, в отличие от выше перечисленных работ. Также в выше перечисленных работах проводится выявление терминов аспектов, а данной работе проводится классификация по категории аспектов. В перспективе, в функционал программы будет добавлен метод выявления терминов аспектов. Также итоговая программа содержит удобный графический пользовательский интерфейс.

Итоговая программа будет полезна для компаний, бизнесов или брендов, для анализа отзывов их клиентов или пользователей. Большинство существующих инструментов анализа отзывов, выявляют только тональность отзывов, а итоговый продукт этой работы выявляет тональность отзыва приписывая её к аспекту. В данный момент, программа может анализировать отзывы из двух сфер: ресторан, пиво. В перспективе будут добавлены новые сферы, и разработан алгоритм, выполняющий анализ в независимости от сферы. Использование данной программы, поможет бизнесам получить преимущество над конкурентами. Например, улучшая аспекты, которые содержат большое количество отрицательных отзывов.

Цель работы - разработка и создание обучаемого алгоритма, для автоматического анализа тональности текста относительно аспектов.

Задачи поставленные при выполнении проекта:

1. Выбор, разработка и создание модели для извлечения аспектов
2. Тренировка и оценка модели на двух наборах данных
3. Выбор и использование модели выявления тональности
4. Создание графического пользовательского интерфейса (GUI)

Среда разработки и используемые библиотеки python:

Использовалась среда разработки PyCharm, так как является одной из самых популярных и удобных в работе среде разработок. Поддерживается работа с GitHub, куда по итогу был загружен проект (<https://github.com/xAxRxTxUxRx/aspect-based-sentiment-analysis>). Использовался Python 3.9.6 и библиотеки : tensorflow, numpy, gensim, tkinter, nltk, sklearn, keras (на основе tensorflow), pandas и другие.

2. ВЫБОР МОДЕЛЕЙ. ОПИСАНИЕ СУЩЕСТВУЮЩИХ СИСТЕМ

Две основные задачи поставленные в данной работе: выявления тональности и извлечение аспектов.

В сети существует огромное количество различных моделей, выполняющих выявление тональности текста (по бинарной шкале, по многополосной шкале, по системе шкалирования). Так что нужно выбрать самую оптимальную и подходящую из них. Была выбрана натренированная модель (bert-base-multilingual-uncased-sentiment (<https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment>), её преимущества заключаются в том, что тональность определяется по 5-ти бальной шкале. Точность модели 67% и 95% на английском языке, 67% где модель определяет точное количество звёзд, помеченное человеком, 95% +- 1 звезда. Также модель обучена на 6 языках: английский, итальянский, французский, датский, испанский, немецкий, что может быть очень полезно для возможного будущего расширение программы на разные языки.

В качестве модели, выполняющей задачу извлечения аспектов выбрана модель АВАЕ (attention based aspect extraction) представленная в работе [3]. Описание: модель машинного обучения “без учителя”, основанная на механизме внимания, является под видом модели encoder-decoder. Эта модель является основополагающей и единственной из моделей выполняющих классификацию отзывов по категориям аспектов. Официальный код этой работы, по большей части является устаревшим и нерабочим: (<https://github.com/ruidan/Unsupervised-Aspect-Extraction>). Другие имплементации данной модели является либо также устаревшими, либо неполными. Так что, полностью готовой и работающей имплементации данной модели в интернете нет. Ссылки на репозитории на данную тему: ([alexeyev/abae-pytorch](https://github.com/alexeyev/abae-pytorch)), (<https://github.com/onetree1994/Modified-and-Annotated-Code-of-An-Unsupervised-Neural-Attention-Model-for-Aspect-Extraction>). Так что, задача состоит в создании модифицированного кода, имплементирующего АВАЕ модель.

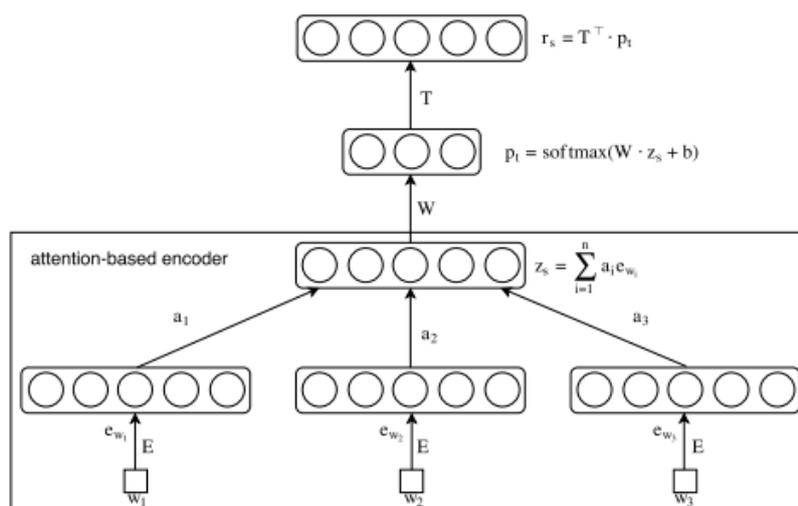


Рисунок 2.1. Схема АВАЕ модели, предоставленная в научной работе [9].

В интернете существуют инструменты для семантического анализа (<https://appfollow.io/ru/reviews-management-academy/automation-of-collecting-insights-through-reviews-semantic-analysis-and-auto-tagging>), (<https://www.reviewpro.com/blog/new-product-release-semantic-3-0/>) и другие, один из них основан на сфере отелей, другой на мобильных приложения. Оба продукта являются платными. Первый проект, из выше перечисленных, имеет полезный пользовательский интерфейс с очень подробным анализом, это будет взято на заметку и использовано в перспективе в этой работе.

3. ОПИСАНИЕ СОЗДАНИЯ МОДЕЛЕЙ

3.1 Модель извлечения аспектов (АВАЕ)

Параметры создания модели:

vocab - словарь

vocab_size - размер словаря

neg_samples - количество негативных примеров

emb_dim - размерность embedding пространства

aspect_size - количество аспектов, которое должна определить модель

dim = maxlen - размерность, длина самого длинного отзыва

batch_size - количество батчей

Разработка и описание структуры.

Подробно рассмотрим структуру данной модели (*В этом разделе часто используется слово `embedding` в связки со словами “слово”, “предложение”. `Embedding` - векторное представление слова. Преимущества `embedding` в том, что слова со схожим значением имеют схожий вектор.):

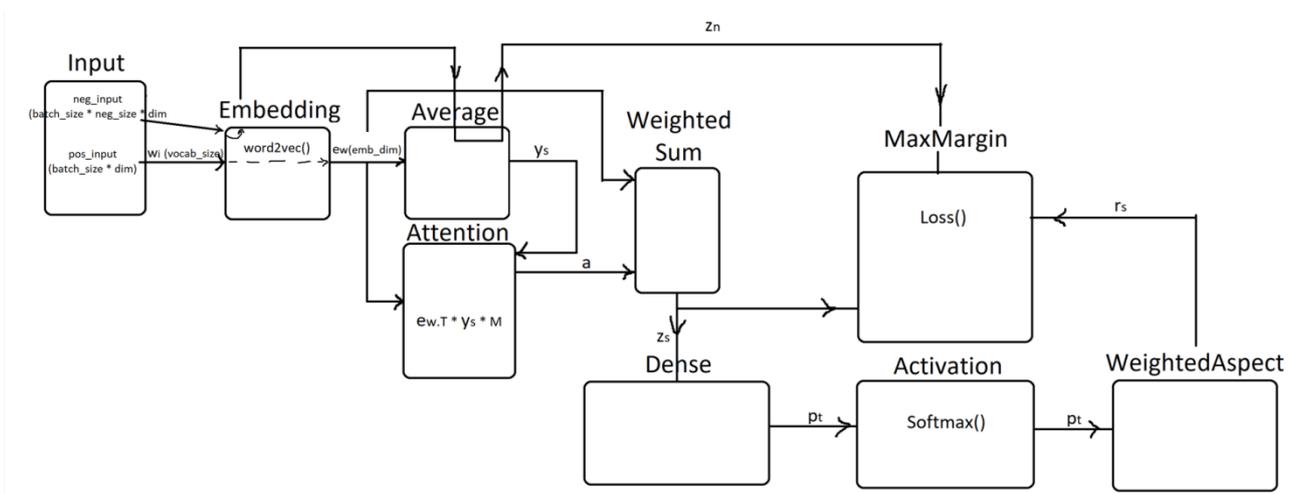


Рисунок 3.1. Послойная схема АВАЕ модели. Каждый прямоугольник обозначает слой, стрелки показывают как `Input` проходит через модель.

По ходу этого проекта была разработана послойная схема модели АВАЕ. На схеме мы можем наблюдать 9 слоёв, входящих в состав модели. Опишем принцип работы каждого из них, также опишем принцип работы слоёв с точки зрения линейной алгебры:

`Input(tensorflow.keras.Input)` слой, на входе получает два тензора: `pos_input` и `neg_input`. Оба тензора содержат целые числа, индексы слов в словаре `vocab`. Тензоры заданы на следующих векторных пространствах `pos_input` - (`batch_size` x `dim`), содержит обычные примеры отзывов, на которых будет тренироваться модель, `neg_input` - (`batch_size` x `neg_samples` x `dim`), содержит примеры которые будут использованы для “плохого пример” для модели, чтобы модель понимала как не должно быть, далее будет понятнее, зачем нужны эти примеры.

Embedding(tensorflow.keras.layers.Embedding) слой, переводящий индексы слов в их вектора (embeddings). Параметры для создания модели: input_dim = vocab_size (размер словаря), output_dim = emb_dim (размерность embedding пространства). В этот слой подаются pos_input и neg_input. Превращает e_w и e_w_n соответственно. Математическое описание: у тензоров pos_input (batch_size x dim) и neg_input (batch_size x neg_samples x dim) добавляется размерность emb_dim по оси -1.

Average(custom_layers.Average) слой, получающий на входе тензор и возвращающий средний тензор. В этот слой подаются e_w и e_w_n, превращаются в y_s и z_n, соответственно. z_n это конструкция поданного предложение используя neg_input, то есть неправильная конструкция. Математическое описание: Этот слой подсчитывает значение по формуле:

$\text{sum}(x, \text{axis} = -2) / \text{len}(x[-2])$ для каждого батча. На выходе тензоры y_s (batch_size x emb_dim) и z_n (batch_size x neg_samples x emb_dim) соответственно.

Attention(custom_layers.Attention) слой. Attention mechanism, используемый этим слоем, основой модели. Он подсчитывает на какие слова нужно обратить внимание (Attention), чтобы определить аспекты. То есть он подсчитывает веса для слов, слова не являющиеся аспектами имеют меньший вес, меньший Attention, являющиеся имеют больший вес. Attention слой содержит тензор M размером (emb_dim x emb_dim), он является частью обучения модели, то есть он изменяется в процессе обучение для понижения неточности модели. Подсчитывается attention_weights по формуле $\text{softmax}(e_w.T * M * y_s)$. Математическое описание: этот слой подсчитывает значение следующим образом: $y_s = M (\text{emb_dim} \times \text{emb_dim}) * y_s.T (\text{emb_dim} \times \text{batch_size})$, далее расширяем размерность y_s с 2 до 3, добавляя измерению по оси -2 -> $\text{expand_dims}(y_s, \text{axis}=-2)$, сейчас размерность y_s равна (emb_dim x 1 x batch_size), чтобы y_s можно было умножить на e_w нужно повторить ось 1 dim раз -> $y_s = \text{repeat}(y_s, \text{dim}, \text{axis}=1)$. Теперь финальный шаг:
 $a = \text{softmax}(e_w (\text{batch_size} \times \text{dim} \times \text{emb_dim})) * y_s (\text{emb_dim} \times \text{dim} \times \text{batch_size})$,

squeeze (a, axis=-1), сокращая размерность dim, но сохраняя все данные. Attention слой возвращает a (batch_size x dim)

WeightedSum(custom_layer.WeightedSum) слой, получает на вход тензор и весовую матрицу (матрица может являться частным случаем тензора) для неё. Подсчитывается сумма элементов тензора, получившейся в результате умножения двух поданных тензоров. На вход: e_w, attention_weights, на выходе: z_s (конструкция предложения на основе pos_input). Математическое описание: Этот слой подсчитывает значение по формуле: $\text{sum}(e_w (\text{batch_size} \times \text{dim} \times \text{emb_dim}) * a(\text{batch_size} \times \text{dim}), \text{axis}=1)$ и возвращает z_s (batch_size x emb_dim)

Dense(tensorflow.keras.layers.Dense) стандартный слой, через который проходит z_s с изменением размерности, и соединяется с Activation слоем.

Activation(tensorflow.keras.layers.Activation) слой, применяют функцию активации к входным данным, в данном случае функция активации - softmax() возвращает p_t. Математическое описание: Далее z_s проходит через Dense слой, и становится размера (batch_size x aspect_size), потом через Activation слой, где применяется softmax(z_s), по итогу двух этих слоёв мы получаем p_t (batch_size x aspect_size)

WeightedAspectEmbedding (custom_layers.WeightedAspectEmbedding) слой, подсчитывает r_s (реконструкцию предложения), которая будет сравниваться с z_n и z_s, если модель функционирует правильно, то r_s будет приближен к z_s и отдалён от z_n, с помощью этого подсчитывается loss модели. Для подсчёта используется матрица W, размера (aspect_size x emb_dim), как и матрица M, является процессом обучения матрицы. На вход подаётся p_t и возвращается тензор, полученный в процессе умножения p_t на W. (Матрицы W и M созданы случайным образом в начале и изменяются в процессе обучения повышая точность модели) Математическое описание: этот слой подсчитывает значение по формуле: $p_t (\text{batch_size} \times \text{aspect_size}) * W (\text{aspect_size} \times \text{emb_dim})$ и возвращается реконструкция предложения r_s (batch_size x emb_dim).

MaxMargin (custom_layers.MaxMargin) слой, подсчитывающий loss для модели. На вход подаются z_s, z_n и r_s. Тензоры нормализуются и считается

loss по формуле MaxMargin, более подробно описанной в пункте “Математическое описание модели”. Возвращается loss для каждого batch. Математическое описание: Этот слой подсчитывает loss, который стремится уменьшить модель. Для начала, нормализуем матрицы, для все тензоров применим функцию нормализации (tensorflow.math.l2_normalize): $\sqrt{\max(\text{sum}(\text{square}(x)), \text{epsilon}))} * x$. Далее посчитаем $\text{pos}(\text{batch_size} \times 1) = \text{sum}(z_s(\text{batch_size} \times \text{emb_dim}) * r_s(\text{batch_size} \times \text{emb_dim}), \text{axis}=-1)$, далее повторяем batch_size размерность neg_samples раз и получаем $\text{pos}(\text{batch_size} \times \text{neg_samples})$, далее расширяем мерность тензора r_s с 2 до 3 -> $\text{expand_dims}(r_s, -2)$ и получаем r_s (batch_size x 1 x emb_dim), далее повторяем размерность 1 neg_samples раз -> $\text{repeat}(r_s, \text{neg_samples}, 1)$ и получаем r_s (batch_size x neg_samples x emb_dim). Далее посчитаем $\text{neg}(\text{batch_size} \times \text{neg_samples}) = \text{sum}(r_s * z_n, \text{axis}=-1)$ Теперь воспользуемся формулой для подсчёта loss, формула MaxMargin: $\text{loss}(\text{batch_size} \times 1) = \text{sum}(\max(0, (1 - \text{pos} + \text{neg})), \text{axis} - 1)$, следовательно, задача модели - уменьшить neg и увеличить pos.

Создание программы, воплощающей АВАЕ.

Файл с кодом программы можно найти в репозитории GitHub (<https://github.com/xAxRxTxUxRx/aspect-based-sentiment-analysis>). Данный репозиторий не содержит наборы данных и преобразованные данные, так как они слишком большого размера.

Этапы работы программы модели можно поделить на следующие:

1. Обработка и считывание наборов данных. Создание словаря.
2. Работа с word embeddings

3. Создание самой модели

4. Тренировка и оценка (будут разобраны в 4 разделе)

Создадим директорию `datasets/{domain}` в каждый из доменов загрузим наши наборы данных и получим файлы `train.txt`, `test.txt`, `test_label.txt`. Имея данные нам нужно их обработать.

Для обработки текстовых данных создадим файл `preprocess.py`, выполняющий следующие преобразования текста: токенизация текста (превращение текста в массив слов), удаление стоп слов (часто используемые слова, используемые для построения предложений), лемматизация слов (преобразование слов в начальную форму, пример: `words -> word`, `goes -> go`)
Создаётся папка `preprocessed_data/{domain}` в которую записываются обработанные файлы.

Создание словаря. Считываем каждое слово с наборов данных и подсчитываем, сколько раз оно встречается в `preprocessed_data/{domain}/train.txt`. Создаём файл `vocab.txt` и записываем в него уникальные слова (одно и то же слово не будет записано несколько раз) и его индекс (количество встреч этого слова в наборе данных). Записываем этот файл в `preprocessed_data/{domain}`.
Файл выполняющий эти задачи `reader.py`:

АВАЕ модель основана на `word embedding`. `Word embedding` это алгоритм обученный представлять слова в виде векторов в `emb_dim`-мерном пространстве, где `emb-dim` - мерность `embedding` пространства. Сила `word embedding` заключается в том, что векторы слов близких по значению или контексту находятся рядом в `embedding` пространстве. Для создание модели `word2vec` была использована модель `word2vec` из библиотеки `gensim` (`gensim.models.word2vec`) Она была тренирована на созданном словаре `vocab.txt`. Задачу создания модели выполнил файл `embedding.py`.

Для работы с `word embeddings` был создан файл `Word2VecReader.py`, выполняющий задачи, предоставления вектора данного слова, создание `word`

embedding для словаря, создание матрицы aspect embedding, на основе алгоритма KMeans, содержащей embeddings для данного числа аспектов.

Файл model.py отображают полную модель. В нем находится функция build_model(), которая создаёт модель соединяя слои (tensorflow.keras.layers) в модель tensorflow.keras.Model. Модель содержит 9 слоёв, 5 из которых имплементированы библиотекой keras, 4 слоя кастомных(пользовательских) слоя, которые наследуют класс tensorflow.keras.layers.Layers. Документация о слоях, выполненных keras (Input, Embedding, Attention, Dense, Activation) находится по ссылке: https://www.tensorflow.org/api_docs/python/tf/keras/layers/. Кастомные слои (Average, WeightedSum, WeightedAspectEmbedding, MaxMargin) в файле выполнены custom_layers.py. Они основаны на keras.layers.Layer, подсчёты ведутся с помощью библиотеки tensorflow (.math, .keras.backend). В файле build_model() создаётся модель tensorflow.keras.Model, состоящая из всех слоёв. После создания, embedding матрицы слоёв Embedding и WeightedAspectEmb назначаются, используя функции файла w2vEmbReader.

3.2 Модель выявления тональности

Для выявления тональности текста выбрана модель (<https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment>). Она была интегрирована в программу с помощью библиотеки transformers и классов AutoTokenizer, AutoModelForSequenceClassification. С помощью функции .from_pretrained('nlptown/bert-base-multilingual-uncased-sentiment') мы загружаем токенайзер и саму модель по ссылке модели. Теперь, чтобы пользоваться моделью, мы преобразуем отзыв с помощью функции токенайзера .encode() и вызываем модель от преобразованного отзыва.

4. ОЦЕНКА МОДЕЛИ И СОЗДАНИЕ ИТОГОВОГО ПРОДУКТА

4.1 Тренировка и оценка АВАЕ модели

Для тренировки модели создан train.py. В нём определяются параметры создания модели. Для обучения модели были заданы следующие параметры:

```
emb_dim = 200
```

aspect_size = 14

neg_samples = 20

batch_size = 64

epochs = 15

domain = 'restaurant' и 'beer'

Algorithm = RMSprop (learning_rate = 0.1)

vocab, vocab_size и maxlen вычислены по ходу программы.

С помощью функций pos_batch_generate() и neg_input_generate() созданы pos и neg inputs. Были проведены тренировки модели на различных оптимизаторах (adam, sgd, adagrad, adamax, rmsprop) и было выявлено что rmsprop является лучшим алгоритмом оптимизации для данной модели. Был создан optimizer с помощью класса tensorflow.keras.optimizers.RMSprop, параметры: learning_rate = 0.1, epsilon=1e-06, clipnorm=10. Далее компилируется модель Model.compile(). model.compile(optimizer=optimizer, loss=max_margin_loss, metrics=[max_margin_loss]). Для каждой эпохи создаём pos и neg input и тренируем модель с помощью метода train_on_batch(), для каждого батча получаем loss и max_margin_loss (возвращается функцией train_on_batch()) если loss меньше минимального, то веса модели и текущий loss сохраняются, и выводятся 50 приближённых слов к словам полученным из весовой матрицы слоя WeightedAspectEmb. В конце эпохи выводится loss и max margin loss.

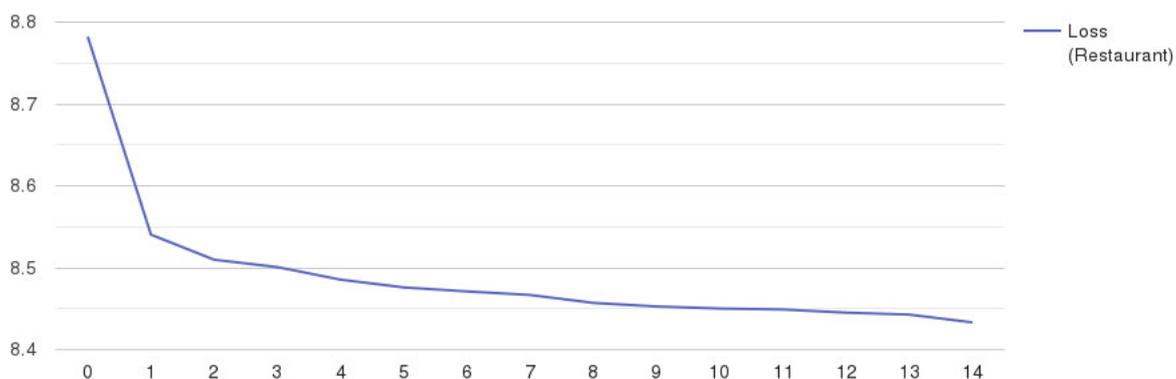


Рисунок 4.1. График изменения значения функции loss с каждой эпохой для домена “ресторан”.

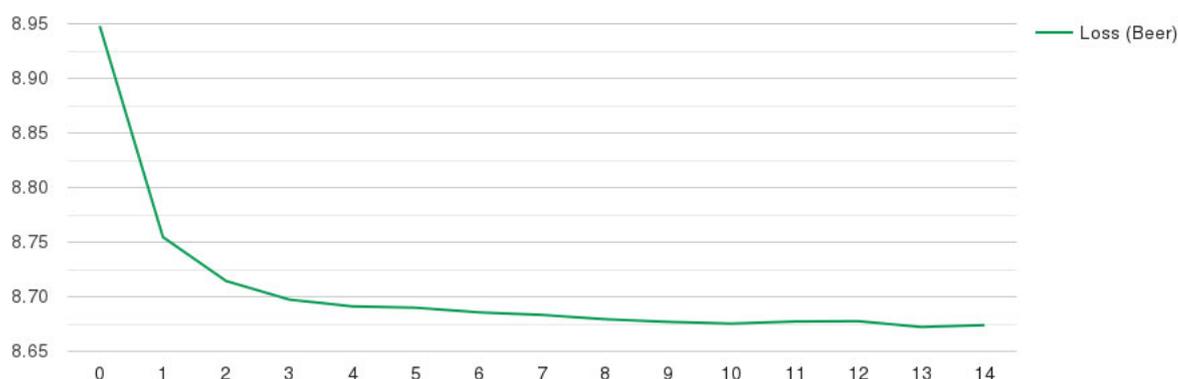


Рисунок 4.2. График изменения значения функции loss с каждой эпохой для домена “пиво”.

По итогам тренировки на последней эпохе, модель достигла следующих результатов: Total loss: 8.433, max_margin_loss: 8.344, ortho_reg: 0.088, для домена “ресторан” и Total loss: 8.67, max_margin_loss: 8.55, ortho_reg: 0.122 для домена “пиво”. Также были определены 14 аспектов, и 50 слов представляющих эти аспекты (50 для каждого аспекта). Далее вручную создан cluster_map, который к каждой группе аспектов, приписывает золотой аспект (Food, Staff, Ambience, Price, Miscellaneous) и (feel, taste+smell, look, overall, None). Теперь, при подачи в модель отзыва, мы получаем значения pt (значения выходных данных слоя Activation). На выходе получается вероятность того, на сколько данный отзыв относится к каждому из аспектов.

Cluster_map для домена ресторан cluster_map_restaurant = {0: 'Food', 1: 'Food', 2: 'Ambience', 3: 'Ambience', 4: 'Miscellaneous', 5: 'Food', 6: 'Miscellaneous', 7: 'Miscellaneous', 8: 'Staff', 9: 'Price', 10: 'Food', 11: 'Staff', 12: 'Miscellaneous', 13: 'Ambience'}

Cluster_map для домена пиво cluster_map_beer = {0: 'overall', 1: 'taste+smell', 2: 'overall', 3: 'taste+smell', 4: 'None', 5: 'taste+smell', 6: 'feel', 7: 'None', 8: 'taste+smell', 9: 'None', 10: 'look', 11: 'taste+smell', 12: 'None', 13: 'None'}

Весовые матрицы модели (M(из слоя attention) и W(из слоя WeightedAspectEmb)) были сохранены после тренировки с помощью функции save_weights().

Для оценки натренированной модели был создан файл evaluate.py, в котором была воссоздана модель с такими же параметрами и были загружены весовые матрицы с помощью функции load_weights(). Далее мы используем файлы test.txt и test_labels.txt. test_labels.txt - файл аспектов для отзывов из test.txt.

И на вход подаются данные из файла test.txt и сравниваются с test_labels.txt, в которых вручную помечены аспекты. В этом этапе были определены следующие результаты:

Домен: ресторан

Точность: 0.5865771812080537

	precision	recall	f1-score	support
Food	0.910	0.623	0.740	887
Staff	0.833	0.511	0.634	352
Ambience	0.671	0.562	0.612	251
Price	0.000	0.000	0.000	0
Miscellaneous	0.000	0.000	0.000	0
accuracy			0.587	1490
macro avg	0.483	0.339	0.397	1490
weighted avg	0.851	0.587	0.693	1490

Домен: пиво

Точность: 0.7199480181936322

	precision	recall	f1-score	support
feel	0.845	0.737	0.787	1022
taste+smell	0.900	0.821	0.859	3672
look	0.941	0.747	0.833	1607
overall	0.621	0.505	0.557	2090
None	0.310	0.741	0.437	843
accuracy			0.720	9234
macro avg	0.723	0.710	0.695	9234
weighted avg	0.784	0.720	0.739	9234

4.2 Создание пользовательского интерфейса

Файлы `final_models.py`, `preprocess_input.py`, `interface.py` были разработаны для создания графического пользовательского интерфейса. Файл `final_models.py` содержит два класса `ABAE_model` и `Sentiment_model` которые возвращают готовые к использованию модели и предсказывают аспекты и тональность соответственно.

Файл `interface.py` является пользовательским интерфейсом. В нем загружаются модели с помощью `final_models.py` и обрабатываются входные данные с помощью `preprocess_input.py`. На вход может подаваться либо единичный отзыв, который подаётся в обе модели, и на выходе выводится аспект к которому относится отзыв и его тональность, либо через файл с расширением `.txt` подаётся набор отзывов, содержащий отзыв на каждой строчке и выдаётся файл `.xlsx`, который можно скачать. Выходной файл содержит таблицу, в которой строка это аспект, а столбец это тональность (1-5), пересечение столбцов и строк показывает количество отзывов, относящихся к конкретному аспекту и тональности.

Графический интерфейс выполнен с помощью библиотеки `tkinter`. Были использованы следующие виджеты: `Canvas`, `Label`, `CheckButton`, `Text`, `Button`. Итоговый интерфейс выглядит следующим образом:



Рисунок 4.3. Графический пользовательский интерфейс(выполнен на библиотеке tkinter).

5. ЗАКЛЮЧЕНИЕ

Итоговым продуктом является программа, содержащая GUI, выполняющая задачу анализа тональности текста на основе аспектов. По ходу работы было подмечено много идей для будущей реализации:

1. Повышение функционала программы: тренировка модели АВАЕ на других сферах, имплементация метода извлечения терминов аспектов.
2. Поддержка анализа отзывов на разных языках (по ходу работы было выявлено, что (<https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment>) может определять тональность русскоязычных текстов, осталось придумать как извлекать аспекты из русскоязычных отзывов)
3. Улучшение GUI. На данном этапе пользовательский интерфейс выглядит просто. Нужно добавить визуализацию анализированных данных и более подробный анализ в целом.
4. Добавление подаспектов. Данная программа классифицирует отзывы только на основные аспекты, которые были помечены в наборах данных, но было бы полезно разбить эти аспекты на группы (пример: персонал: повар, официант, уборщик и т.д.).

6. СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Письменный, Д. Т. П35 Конспект лекций по высшей математике: полный курс/ Д. Т. Письменный. - 10-е изд., испр. - М.: Айрис-пресс, 2011. - 608 с.: ил. - (Высшее образование). ISBN 978-5-8112-4351-8.
2. Bing Liu. Sentiment Analysis and Opinion Mining second edition / Morgan & Claypool Publishers - 2020 - Morgan & Claypool
3. An Unsupervised Neural Attention Model for Aspect Extraction / Ruidan He, Wee Sun Lee, Hwee Tou Ng, Daniel Dahlmeier - P17-1036 - 2017 Vancouver, Canada - Association for Computational Linguistics.
4. Distributed Representations of Words and Phrases and their Compositionality / Tomas Mikolov, Ilya Sutskever, Kai Chen, [и др.] - 2013. - Curran Associates Inc. Red Hook, NY, United States.

ПРИЛОЖЕНИЕ

Презентация:



Защите_ABSA.ppt

Код программы можно найти в репозитории:

<https://github.com/xAxRxTxUxRx/aspect-based-sentiment-analysis>

Итоговый продукт можно скачать по ссылке:

(будет готов к защите)