ВИКОРИСТАННЯ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ СЕНТИМЕНТ-АНАЛІЗУ

Харченко С. Д.
Харківський національний автомобільно-дорожній університет

В аналізі текстів, зокрема в визначенні тональності, постійно розвиваються та вдосконалюються методи та підходи. Запитання про те, як автоматизовано розпізнавати емоційний відтінок висловлених думок та висловлювань, є важливим завданням у сучасній обробці природної мови. В даному розділі ми ретельно розглянемо сучасні методи визначення тональності, починаючи від традиційних правилових підходів та евристик, переходячи до методів машинного навчання та фокусуючись на застосуванні нейронних мереж.

Правила та евристичні методи визначення тональності використовують заздалегідь визначені правила та словники для ідентифікації позитивних чи негативних емоцій у текстах. З одного боку, це ефективні методи для простих сценаріїв, але вони обмежені у врахуванні контексту та адаптації до різних стилів мови.

Методи машинного навчання дозволяють автоматично визначати емоційний тон текстів, використовуючи моделі, навчені на великих наборах позначених даних. Ці підходи виявилися ефективними у вирішенні завдань аналізу тональності, але вони також мають свої обмеження, зокрема, у врахуванні складних залежностей в текстах та управлінні суб'єктивністю.

Останнім часом нейронні мережі завоювали популярність завдяки своїй здатності автоматично вивчати складні залежності та пристосовуватися до різних умов. У цьому контексті вони стають потужним інструментом для аналізу тональності, дозволяючи враховувати контекст, адаптуватися до різних стилів мови та працювати з великими обсягами даних.

Визначення методу для перевірки тональності тексту

МЛ-тональність - механізм визначення тональності тексту стосовно об'єкта моніторингу за допомогою AI-моделі, натренованої на корпусі текстів. Ці тексти спочатку були розмічені вручну за тональністю по відношенню до об'єктів, що зустрічалися в текстах. Для цього використовувались правила оцінки тональності, які використовуються під час роботи медіа-аналітиків.

Для МЛ-тональності використовується так званий об’єктно-орієнтований алгоритм визначення тональності (Entity related sentiment analysis), тобто емоційне забарвлення повідомлень визначається не просто для публікації як такої, а стосовно об’єкту моніторингу у темі. Об’єкт моніторингу – це ті ключові слова, які у вашій темі/темах визначені як "виділені ключові слова".

Під час створення теми менеджером системи або користувачем необхідно в окреме поле виділених ключових слів ввести ті слова та словосполучення, відносно яких буде визначатись тональність.

Так як для роботи були обрані згорткові нейронні мережі, нижче наведемо алгоритм їх дії. Алгоритм роботи з загортковими нейронними мережами:

1. Підготовка даних:

Перетворення тексту в числовий вектор за допомогою токенізації та векторизації слів (наприклад, за допомогою векторізаторів, як TF-IDF чи Word2Vec). Поділ даних на тренувальний та тестовий набори.

2. Створення моделі згорткової нейронної мережі:

Створення шарів згортки та пулінгу для виділення важливих ознак у тексті. Використання рекурентних шарів або шарів LSTM (Long Short-Term Memory) для збереження контексту між словами. Використання повнозв'язаних шарів для класифікації тексту на позитивний, негативний чи нейтральний.

3. Навчання та оптимізація:

Використання тренувального набору для навчання моделі. Вибір функції втрат та оптимізатора для мінімізації втрат під час навчання. Налаштування гіперпараметрів, таких як розмір фільтрів та кількість шарів.

4. Тестування та оцінка:

Використання тестового набору для оцінки продуктивності моделі. Визначення метрик ефективності, таких як точність, відновлення, та F-мера.

5. Удосконалення моделі:

Вирішення проблем перенавчання шляхом використання методів регуляризації та викидів. Оптимізація гіперпараметрів для покращення результатів.

Метод Adam

Вибір алгоритму для оптимізації системи є дуже важливим питанням, адже це може певним чином покращити результати навчання, та час навчання мережі.

Метод Адама є досить новим, та отримав досить велику популярність при навчанні глибоких мереж у сферах комп’ютерного зору та роспізнавання мови та тексту. Також варто зазначити що цей метод є модифікацією методу стахестичного градієнтного спуску, тож часто використовується замість нього для ітеративного корегування ваг мережі на основі вхідних даних.

Розглянемо чому саме обирають цей алгоритм, для цього наведемо його переваги котрі надали автори методу:

- Метод добре працює із задачами, котрі вимагають великих об’ємів даних;
- Є досить простим та не привередливим до сервера на котрому відбувається навчання, хоча б тому що не потребує багато пам’яті;
- Гіперпараметри методу є досить простими для розуміння та часто не потребують важкого та довгого налаштування;
- Відносно інших модифікацій є досить простим для реалізації та не потребує довгого написання коду;
- Має велику обчислювальну ефективність;

Порівнюючи метод Adam з його початковою версією – методом стохастичного градієнтного спуску, то перший підтримує постійну швидкість

навчання для всіх регуляцій ваг, в другому випадку було поєднано найкращі сторони двох інших модифікацій першого і метод адаптує швидкість навчання для кожного з параметрів використовуючи першого та другого моментів градієнту. Виходить так, що Adam на відміну від методу RMSProP використовує не тільки середнє першого моменту, але й середнє другого моменту градієнта – нецентрована дисперсія.

Для демонстрації можливостей цього методу було проведено навчання багатошарового персептрона і порівняно результати(training cost) з іншими методами. Далі розглянемо параметри котрі має даний метод, вони мають певні стандартні значення у популярних бібліотеках наприклад мови Python, таких як Tensorflow, Keras(надбудова Tensorflow), Blocks.

- Параметр a - це значення задає швидкість навчання, або розмір кроку з котрим оновлюються ваги.

- Параметр b1 – це показник експоненційного затухання першого моменту

- Параметр b2 – цей параметр має такий самий сенс як і попередній але для другого моменту, це значення може бути близьким до 1 або навпаки дуже малим значенням. Всі ці параметри мають певні стандартні та рекомендовані значення у вище наведених бібліотеках для спрощення роботи з ними.

Опис обраної нейронної мережі

Нейромережа, написана на Python, буде відрізнятися швидкістю, безпекою та стабільністю. Сама мова програмування передбачає такі переваги: функціональність, поділ підсумкової кодифікації на блоки, що дозволяє значно підвищити її читання, підтримка довгої арифметики, кросплатформність, безліч бібліотек, які зможуть отримати в будь-який час, зрозумілий синтаксис. Це ідеальний варіант для веб-розробки, додатків для месенджерів та дрібних проектів.

Sequential (tensorflow.keras.models.Sequential) – Ця бібліотека надає можливість створення лінійних моделей шляхом послідовного додавання шарів. Використовується для зручного визначення та навчання нейронних мереж.

Dense (tensorflow.keras.layers.Dense) – Шар, що представляє повнозв'язаний шар у нейронній мережі. Кожен нейрон у цьому шарі пов'язаний з кожним нейроном попереднього та наступного шарів.

Embedding (tensorflow.keras.layers.Embedding) – Шар вбудування для конвертації текстових даних в числовий вектор, де кожне слово представлено унікальним числовим значенням.

Conv1D (tensorflow.keras.layers.Conv1D) – Шар одновимірної згорткової нейронної мережі, який використовується для виявлення локальних залежностей у текстових даних.

GlobalMaxPooling1D (tensorflow.keras.layers.GlobalMaxPooling1D) – Шар для вибору максимального значення з кожного згорткового фільтра, що допомагає визначити найважливіші ознаки.

Dropout (tensorflow.keras.layers.Dropout) – Шар випадкового відключення, який допомагає уникнути перенавчання за допомогою випадкового вимкнення деяких нейронів під час навчання.

pad_sequences (tensorflow.keras.preprocessing.sequence.pad_sequences) – Функція для доповнення або обрізання послідовностей (текстових даних) до заданої довжини. Використовується для створення вхідних даних фіксованого розміру для моделі.

Tokenizer (tensorflow.keras.preprocessing.text.Tokenizer) – Клас для токенізації текстових даних, перетворення слів у числові індекси та створення словників для подальшого використання у вбудуванні.

Реалізація створеної нейронної мережі

Для реалізації нейронної мережі використовується бібліотека TensorFlow з використанням Keras API. Код розбитий на етапи, включаючи завантаження даних, побудову моделі, навчання та оцінку результатів. Застосовується звичайна процедура тренування за допомогою стохастичного градієнтного спуску та зворотнього поширення помилки.

Для практичної реалізації створеної нейронної мережі використовується потужна бібліотека TensorFlow, яка забезпечує ефективні інструменти для створення, навчання та експериментування з нейронними мережами. Реалізація проводиться через високорівневий інтерфейс Keras API, який дозволяє створювати та конфігурувати нейронні мережі з легкістю.

Основні етапи включають:

1. Завантаження даних: В цьому етапі використовуються інструменти TensorFlow для завантаження набору даних MNIST, які використовуються для тренування та тестування моделі.

2. Побудова моделі: Використовуючи Keras API, створюється архітектура нейронної мережі з вхідним, прихованим та вихідним шарами. Параметри моделі, такі як кількість шарів, кількість нейронів та функції активації, конфігуруються в цьому етапі.

3. Навчання моделі: Використовуючи стохастичний градієнтний спуск та зворотнє поширення помилки, модель навчається на тренувальному наборі даних. Ваги нейронів оновлюються так, щоб мінімізувати функцію втрат.

4. Оцінка результатів: Після завершення тренування проводиться оцінка результатів на тестовому наборі даних. Використовуються різні метрики, такі як точність класифікації та матриця плутанини, для об'єктивного визначення ефективності нейронної мережі.

Висновок

У даному проекті я використовувала бібліотеки Python, зокрема Pandas для обробки та аналізу табличних даних. Моя робота включала в себе етапи завантаження даних, їх підготовку та взаємодію з ними. Я вивчала та аналізу-

вала набір даних за допомогою Pandas, використовуючи методи для вибірки, фільтрації, та обчислення агрегованих статистик. Важливим етапом було використання бібліотеки Matplotlib для візуалізації результатів та отримання більш глибокого розуміння даних.

Під час проекту я також використовувала нейромережі, зокрема Tensor Flow та Keras, для навчання моделі на текстових даних. Я завантажувала та підготовлювала дані, використовуючи Tokenizer для векторизації тексту. Моя робота включала створення та навчання моделі, аналіз її результатів та тестування на нових даних.

Усе це спрямовано на досягнення кращого розуміння даних, виявлення закономірностей та навчання моделей для ефективного вирішення конкретних завдань у сферах обробки даних та машинного навчання.

У роботі з аналізом тональності тексту нейронні мережі виявляються потужним та ефективним інструментом. Застосування цих алгоритмів до визначення емоційного відтінку висловлень не лише дозволяє автоматизувати процес, але і забезпечує значно кращі результати в порівнянні з традиційними методами. Згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN), використовуючи свою здатність враховувати контекст та вивчати складні залежності в тексті, стали ключовими гравцями у вирішенні задачі визначення тональності. Використання шарів згортки та пулінгу дозволяє виділяти ключові ознаки, тоді як рекурентні шари враховують послідовність слів. Процес обробки текстових даних перед використанням в нейронних мережах також важливий. Використання Tokenizer для перетворення текстових відгуків у числові послідовності дозволяє нам подати інформацію так, що її можна використовувати в якості вхідних даних для моделей. Застосування нейронних мереж в аналізі тональності дозволяє не лише отримувати точні результати, але і адаптуватися до різних стилів мови, ураховувати контекст та працювати з великими обсягами даних. Це робить їх особливо корисними в сучасному середовищі, де обробка та розуміння великої кількості текстової інформації є ключовими завданнями.

Необхідно відзначити, що використання нейронних мереж вимагає належного підготовчого етапу, великої кількості даних для навчання та оптимізації гіперпараметрів. Проте, з усіма викликами, вони стають потужним засобом для вирішення завдань визначення тональності тексту в різних сферах, таких як відгуки користувачів, аналіз соціальних мереж, та багато інших.

Список джерел

1. Аналіз тональності текстів за допомогою згорткових нейронних мереж.: [Електронний ресурс]. – Режим доступу: https://habr.com/ru/companies/vk/articles/417767/

2. An easy tutorial about Sentiment Analysis with Deep Learning and Keras.: [Електронний ресурс]. – Режим доступу:https://towardsdatascience.com/an-easy-tutorial-about-sentiment-analysis-with-deep-learning-and-keras-2bf52b9cba91

3. How To Train a Neural Network for Sentiment Analysis.: [Електронний ресурс]. – Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-train-a-neural-network-for-sentiment-analysis