**Лекция 11**

**Классификация текстов с помощью BERT**

Поскольку текст представляет собой неструктурированный формат, которому не нравится формат структурированных таблиц, NLP играет очень важную роль при интеллектуальном анализе данных. Это также часть контролируемого обучения, что означает, что он имеет предварительный ответ для данных обучения. После обучения модели мы можем использовать модель для классификации будущего входящего текста. Например, электронная почта может быть отнесена к категории спама или нет, текстовая обратная связь от клиентов может быть разделена на положительную или отрицательную, текст в социальных сетях может быть отнесен к категории «угощение» или «нет» и т. Д.

Еще один практический пример: у нас есть много данных о комментариях клиентов из социальных сетей, таких как Twitter, Facebook и т. Д. Мы хотим понимать, что эти тексты означают как положительные, так и отрицательные. На высоком уровне мы будем работать над следующим процессом.

1. Маркировка вводимого текста по пяти категориям: хорошо, частично хорошо, нейтрально, частично плохо, плохо.
2. Мы можем использовать этот обучающий набор данных для обучения нашей модели НЛП.
3. Мы можем скормить новый текст в модель НЛП, чтобы получить предсказание.

Загрузив приведенный ниже свободный текст, возможно, мы получим вывод ***частично неверно***, тогда менеджер по продукту сможет углубиться в этот текст со стороны пользовательского интерфейса или где-то еще, чтобы внести улучшения, такие как создание изменение на стороне пользовательского интерфейса.

«Мне нравится использовать эту новую функцию, но иногда она меня сильно сбивает с толку, потому что пользовательский интерфейс» - Берт Сноу



Включая приведенный выше пример, обычно мы можем разделить классификацию текста на следующие категории:

**Анализ настроений:** процесс понимания того, говорит ли данный текст положительно или отрицательно о данной теме (например, для целей мониторинга бренда). (Ссылка 1)

**Обнаружение темы:** задача определения темы или темы фрагмента текста (например, узнать, касается ли обзор продукта *Простота использования*, *Поддержка клиентов* или *Pricing* при анализе отзывов клиентов). (Ссылка 1)

**Определение языка:** процедура определения языка данного текста (например, определение того, написан ли входящий билет поддержки на английском или испанском языках, для автоматической маршрутизации билетов в соответствующую команду). (Ссылка 1)

**Модель классификации двоичного текста**

Мы представим три модели двоичной классификации текста с использованием CNN, LSTM и BERT.

**Предварительная обработка данных**

Поскольку мы получаем данные из социальных сетей, таких как Twitter или Facebook, в исходном наборе данных содержится много бесполезных или зашумленных данных. Прежде чем вводить данные в модель НЛП для обучения, нам нужно сначала очистить наши текстовые данные. Я перечисляю некоторые шаги, которые мы выполнили ниже, вы можете изменить любые правила здесь для очистки.

* *удалить пустое правило*
* *правило ретвита*
* *правило хэштега*
* *правило разметки*
* *правило URL*
* *правило электронной почты*
* *числовое правило*
* *удалить знаки препинания*
* *удалить непечатаемые*
* *удалить подмножество ascii*
* *правило лемматизации*
* токенизировать правило
* векторизирующее правило
* *и т. д.*

**Эксперимент CNN и LSTM**

Для модели CNN, которая является кратким названием сверточной нейронной сети, она обычно используется для части изображения, но мы просто используем ее для нашей базовой линии здесь, чтобы увидеть, что является наихудшим случаем. Мы можем найти определение модели ниже.

|  |
| --- |
|  |
| nlp\_cnn\_model = tf.keras.Sequential([ |
|  |

|  |
| --- |
| tf.keras.layers.Embedding(vocab\_size, embedding\_dim, input\_length=max\_length), |
|  |

|  |
| --- |
| tf.keras.layers.Conv1D(128, 5, activation='relu'), |
|  |

|  |
| --- |
| tf.keras.layers.GlobalMaxPooling1D(), |
|  |

|  |
| --- |
| tf.keras.layers.Dense(24, activation='relu'), |
|  |

|  |
| --- |
| tf.keras.layers.Dropout(rate=0.5), |
|  |

|  |
| --- |
| tf.keras.layers.Dense(1, activation='sigmoid') |
|  |

|  |
| --- |
| ]) |

Для модели LSTM, которая представляет собой долгую краткосрочную память, в ней есть все, от рекуррентной нейронной сети (RNN) до соединений обратной связи. Для NLP это имеет больше смысла при использовании LSTM или RNN, потому что некоторые более поздние слова будут влиять на более ранние слова в одном предложении. Мы можем найти определение модели LSTM ниже.

|  |
| --- |
|  |
| nlp\_lstm\_model = tf.keras.Sequential([ |
|  |

|  |
| --- |
| tf.keras.layers.Embedding(vocab\_size, embedding\_dim, input\_length=max\_length), |
|  |

|  |
| --- |
| SpatialDropout1D(0.2), |
|  |

|  |
| --- |
| Conv1D(64, 5, activation='relu'), |
|  |

|  |
| --- |
| Bidirectional(LSTM(64, dropout=0.2, recurrent\_dropout=0.2)), |
|  |

|  |
| --- |
| Dense(512, activation='relu'), |
|  |

|  |
| --- |
| Dropout(0.5), |
|  |

|  |
| --- |
| Dense(512, activation='relu'), |
|  |

|  |
| --- |
| Dense(1, activation='sigmoid') |
|  |

|  |
| --- |
| ]) |

**Экспериментируйте с BERT**

Трансферное обучение - очень важная часть в области машинного обучения.

Он фокусируется на хранении знаний, полученных при решении одной проблемы, и применении их к другой, но связанной проблеме. (Ссылка 2)

BERT предназначен для предварительной подготовки глубоких двунаправленных преобразователей для понимания языка. Мы можем просто добавить пару слоев на основе модели BERT перед обучением. Например, в нашем случае на основе слоя sequence\_output мы добавляем 4 более плотных слоя с выпадением и регуляризатором. На последних уровнях, поскольку мы выполняем двоичную классификацию, num\_classes равно 2. Ниже мы можем заполнить всю модель BERT.

|  |
| --- |
|  |
| bert\_layer = hub.KerasLayer(bert\_module\_url, |
|  |

|  |
| --- |
| signature='tokens',  |
|  |

|  |
| --- |
| signature\_outputs\_as\_dict=True, |
|  |

|  |
| --- |
| trainable=False) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| REST\_METRICS = [ |
|  |

|  |
| --- |
| tf.keras.metrics.AUC(name='auc'), |
|  |

|  |
| --- |
| ] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| METRICS = ACC + REST\_METRICS |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| input\_word\_ids = Input(shape=(max\_len,), dtype=tf.int32, name="input\_word\_ids") |
|  |

|  |
| --- |
| input\_mask = Input(shape=(max\_len,), dtype=tf.int32, name="input\_mask") |
|  |

|  |
| --- |
| segment\_ids = Input(shape=(max\_len,), dtype=tf.int32, name="segment\_ids") |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| \_, sequence\_output = bert\_layer([input\_word\_ids, input\_mask, segment\_ids]) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| x = tf.keras.layers.GlobalAveragePooling1D()(sequence\_output) |
|  |

|  |
| --- |
| regularizer = tf.keras.regularizers.L1L2(l2=0.00008) |
|  |

|  |
| --- |
| x = tf.keras.layers.Dense(96, activation="relu", kernel\_regularizer=regularizer)(x) |
|  |

|  |
| --- |
| x = tf.keras.layers.Dropout(rate=0.3)(x) |
|  |

|  |
| --- |
| x = tf.keras.layers.Dense(64, activation="relu", kernel\_regularizer=regularizer)(x) |
|  |

|  |
| --- |
| x = tf.keras.layers.Dropout(rate=0.3)(x) |
|  |

|  |
| --- |
| x = tf.keras.layers.Dense(16, activation="relu", kernel\_regularizer=regularizer)(x) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| pred = tf.keras.layers.Dense(num\_classes, activation='sigmoid')(x) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| nlp\_bert\_model = Model(inputs=[input\_word\_ids, input\_mask, segment\_ids], outputs=pred) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| optimizer = tf.keras.optimizers.Adam(lr=1e-3) |
|  |

|  |
| --- |
| nlp\_bert\_model.compile( |
|  |

|  |
| --- |
| optimizer=optimizer, |
|  |

|  |
| --- |
| loss=loss, |
|  |

|  |
| --- |
| metrics=METRICS |
|  |

|  |
| --- |
| ) |

**CNN против LSTM против BERT**

На основе всех трех моделей мы рассчитываем некоторые показатели производительности, такие как точность, отзыв, AUC и точность. Также мы обучили наши модели по 15 эпохам.

Мы можем обнаружить, что BERT имеет параметры более чем в 167 раз больше, чем другие, требует больше времени на обучение и дает более хорошие результаты производительности. Мы используем BERT bert\_en\_uncased\_L-24\_H-1024\_A-16.

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
| +------+-----------+--------+------+----------+-----------------------+--------------+ |
|  |

|  |
| --- |
| | | PRECISION | RECALL | AUC | ACCURACY | TRAINING TIME (hours) | TOTAL PARAMS | |
|  |

|  |
| --- |
| +------+-----------+--------+------+----------+-----------------------+--------------+ |
|  |

|  |
| --- |
| | CNN | 0.75 | 0.72 | 0.75 | 0.73 | small time | 1,695,249 | |
|  |

|  |
| --- |
| | LSTM | 0.81 | 0.82 | 0.8 | 0.81 | small time | 1,991,329 | |
|  |

|  |
| --- |
| | BERT | 0.89 | 0.89 | 0.94 | 0.9 | 24 h | 335,247,554 | |
|  |

|  |
| --- |
| +------+-----------+--------+------+----------+-----------------------+--------------+ |
|  |

|  |
| --- |
|  |

**Заключение**

В нашем случае BERT действительно повышает производительность НЛП, но на это уходит много времени. В зависимости от частоты переобучения модели мы можем выбрать, какую версию использовать. Например, нам нужно переобучать модель еженедельно, мы определенно можем использовать BERT для высокой точности, но если нам нужно переобучать модель ежедневно, мы можем использовать другие методы, чтобы сократить время обучения. Однако давайте поприветствуем BERT в нашем мире. Если у вас есть какие-либо вопросы, прокомментируйте здесь, и мы сможем обсудить больше.