001.8 Text classification / Класифікація тексту

Класифікація тексту є фундаментальним завданням у обробці природної мови (NLP), яка включає класифікацію тексту за попередньо визначеними класами або мітками. Загальні застосування класифікації тексту включають виявлення спаму, аналіз настроїв, класифікацію тем і визначення мови. Існує кілька методів класифікації тексту, починаючи від традиційних підходів машинного навчання до передових методів глибокого навчання.

1. Методи на основі правил (Rule-based Methods)

Системи на основі правил класифікують текст на основі попередніх визначених шаблонів або правил, таких як регулярні вирази, відповідність ключових слів або інших лінгвістичних шаблонів:

- Для класифікації тексту вручну набір правил (manually defined). Ці правила можуть написати певні ключові слова або фрази, пов'язані з категорією.
- Текст зіставляється з цими правилами для визначення класу.

Наприклад, спам-фільтр на основі правил може позначати електронні листи, що містять слово «безкоштовно (free)» або такі фрази, як «купити зараз (buy now)», як спам.

Методи прости в реалізації, легкі для розуміння та модифікації. Але вони не масштабуються для великих наборів даних (large datasets) або складних завдань (complex tasks). Негнучкий і бронестійкий (Inflexible and prone) до помилок з невидимими даними (unseen data) або іншим.

2. Традиційні методи на основі машинного навчання (Traditional Machine Learning-based Methods)

Традиційні методи (Learning-based Methods) виконують контрольне навчання, коли класифікатор (classifier) навчається за позначеними даними, щоб вивчити закономірності та робити прогнози. Текст, як правило, перетворюється на числові функції за допомогою таких методів, як Bag of Words (BoW) або Term Frequency-Inverse Document Frequency (TF-IDF), перш ніж передаватись до класифікатора.

Загальні алгоритми машинного навчання (Common Machine Learning Algorithms):

Naive Bayes − імовірний класифікатор (probabilistic classifie), заснований на теоремі Байєса. Він припускає, що ознака (слова)/features (words) є умовно незалежними від оцінки мітки класу, яка повідомляється як «наївне (naive)» припущення. Щоб напрацювати Naive Bayes, нам потрібно заповнити текст у числові функції за допомогою таких методів, як BoW або TF-IDF. Потім ми навчаємо класифікатор Naive Bayes на цих ознаках. Метод швидкий і ефективний, добре працює з невеликими наборами даних. Але припущення про незалежність функцій (feature independence) часто нереалістичне.

Машини опорних векторів (Support Vector Machines, SVM) призначені для пошуку гіперплощини, яка найкраще розділяє різні класи в просторі позначки. У класифікації

тексту SVM часто ефективно за рахунок вашої здатності обробляти простори великої розмірності (high-dimensional spaces). У класифікаторі SVM текст векторизується на функції (наприклад, TF-IDF або BoW), а потім SVM розробляє межу рішень(high-dimensional data), які розділяють текст на різні категорії (categories). SVM добре працює з даними великого розміру (high-dimensional data), часто даючи точні результати. Однак це може бути вільним для великих наборів даних. Також він чутливий до вибору ядра та параметрів.

Логістична регресія (Logistic Regression) − це лінійний класифікатор (linear classifier), який моделює ймовірність належності тексту до певного класу за допомогою логістичної функції. Під час класифікації текст перетворюється на векторні ознаки (feature vectors), а потім до цих ознак використовується логістична регресія для прогнозування ймовірних класів (predict class probabilities). Метод простий у реалізації та інтерпретації. Він добре працює з великими наборами даних (large datasets). Але він припускає лінійну роздільність класів (linear separability of classes), які можуть не виконуватися в деяких випадках.

K-найближчі сусіди (K-Nearest Neighbors, KNN) KNN − це непараметричний алгоритм, який класифікує текст на основі основних класів його «K» найближчих сусідів у просторі позначки. Під час класифікації KNN текст перетворюється на векторні позначки, а потім KNN класифікує новий текст на основі міток найближчих навчальних прикладів. KNN простий і ефективний для невеликих наборів даних (small datasets), не потребує навчання; він спирається на навчання на основі екземплярів. Але це обчислювально довго для великих наборів даних. Його продуктивність може зменшитися із зашумленими даними (noisy data).

3. Методи на основі глибокого навчання (Deep Learning)

Методи глибокого навчання (Deep Learning-based Methods) досягли найсучаснішої продуктивності в класифікації тексту, особливо з великими наборами даних (large datasets). Ці методи автоматично представляють ознаки із текстовими даними (feature representations from text data), що робить їх дуже ефективними для складних завдань НЛП.

Загальні підходи до глибокого навчання (Common Deep Learning Approaches):

Згорточні нейронні мережі (Feature representations from text data CNN) спочатку були розроблені для обробки зображень, CNN також можна замінити для завдань класифікації тексту. CNN витягують із тексту локальні шаблони (local patterns) (наприклад, n-grams) і агрегують їх для створення класифікації (classification). Класифікатор CNN намагається перетворити текст на вбудовування (embeddings) (наприклад, Word2Vec, GloVe), а потім застосовує згорткові фільтри (convolutional filters) для захоплення локальних шаблонів (capture local patterns). На останньому кросі він використовує об'єднання шарів (pooling layers), щоб зменшити розмір і додати результат до класифікатора. CNN добре вловлює локальні відносини (capturing local dependencies) наприклад, n-grams слів), ефективний для завдань, які включають короткий текст (short text). Але він може боротися з довгостроковими залежностями (long-range dependencies).

Повторювані нейронні мережі (RNN) і LSTM (мережі довгострокової пам'яті) / Recurrent Neural Networks (RNNs) and LSTMs (Long Short-Term Memory Networks)

RNN і LSTM розроблені для наступних даних, таких як текст, щоб вони могли довго фіксувати взаємозв'язок між словами. LSTM, зокрема, вирішують проблему зникнення градієнта, дозволяючи моделі зберігати інформацію про течію довгих комп'ютерів. По-перше, RNN / LSTM перетворює текст на вбудовування. Після цього вбудовування передаються в RNN або LSTM, які разом обробляють текст. Остаточний результат передається на повністю підключений рівень для класифікації. Ці методи ефективні для фіксації довготривалих залежностей і контексту в тексті. Вони обробляють розвиток змінної тривалості. Але вони потребують збільшення, особливо для довгих захворювань. Крім того, вони не можуть добре працювати з дуже великими наборами даних, якщо їх не оптимізувати.

Трансформатори (Transformers), такі як BERT (Bidirectional Encoder Representations from Transformers) і GPT (Generative Pre-trained Transformer), зробили революцію в класифікації тексту, захоплюючи глибоку контекстну інформацію. BERT читає текст в обох напрямках, дозволяючи зрозуміти повний контекст слова в реченні. Класифікатор використовує попередньо навчену трансформаторну модель, як-от BERT або GPT, на ефективну модель на завдання класифікації тексту з позначеними даними. Потім трансформатор генерує контекстні вбудовування, які використовують для класифікації тексту. **Transformers** − це найсучасніша продуктивність за багатьма тестами класифікації тексту. Вони справляються зі складними довготривалими залежностями та контекстом кращими, ніж традиційними методами. Однак трансформатори вимагають великих обчислювальних ресурсів як для навчання, так і для висновків, і можуть знадобитися значні позначені дані для точного налаштування.

4. Ансамблеві методи (Ensemble Methods)

Методи ансамблю об'єднують кілька класифікаторів, щоб підвищити продуктивність шляхом зменшення помилок або дисперсії. Загальні техніки ансамблю включають **bagging**, **boosting**, та **stacking**.

Типовий процес класифікації (Typical classification process) включає :

- Навчання кількох базових класифікаторів (наприклад, SVM, Logistic Regression, Random Forest)).
- Об'єднайте свої прогнози за допомогою таких методів, як мажоритарне голосування (majority voting (bagging)) або зважене усереднення (weighted averaging (boosting)).

Приклади класифікаторів ансамблю (ensemble classifiers):

- **Random Forest** : після кількох дерев рішень (multiple decision trees), де шкірне дерево (each tree) навчається на випадковій підмножині даних і функцій (random subset of the data and features).

- **Машини посилення градієнта (Gradient Boosting Machines, GBM)** : Після цього створені слабкі учні (weak learners) (як правило, дерево рішень/decision trees), при цьому кожен учень (learner) виправляє помилки попередніх.

Ансамблі можуть значно підвищити продуктивність шляхом комбінування різних моделей, зменшуючи ризик переобладнання, особливо в шумних наборах даних. Але вони дорогі з оглядової точки зору та складні у реалізації. Інтерпретація моделі ансамблю ускладнюється.

5. Попередньо підготовлені мовні моделі та донавчання (Transfer Learning)

Трансферне навчання (Transfer Learning) передбачає тонке налаштування великих попередніх навчених мовних моделей, таких як BERT, GPT або RoBERTa, для конкретних завдань класифікації тексту. Ці моделі попередньо навчаються на масивних корпусах і вивчають багаті мовні представлення, які можна перенести на конкретні завдання з мінімальними додатковими навчаннями.

Застосування класифікації включає попереднє навчання великої мовної моделі на загальному корпусі (наприклад, Wikipedia) і подальше тонке налаштування моделей на меншому, спеціальному наборі даних.

Ця група методів забезпечує найсучаснішу ефективність багатьох завдань НЛП, включаючи класифікацію тексту. Вони зменшують потребу у великих маркованих наборах даних, наступний етап попереднього навчання вже закріплює загальні лінгвістичні знання. У той же час вони вимагають значної обчислювальної потужності як для попереднього навчання, так і для тонкого налаштування. Також вони можуть вимагати поточного налаштування для досягнення оптимальних результатів для конкретних завдань.

6. Гібридні методи

Гібридні методи (Hybrid Methods) поєднують підходи (наприклад, на основі правил і машинного навчання (rule-based and machine learning) або машинного навчання (machine learning) і глибокого навчання (deep learning)), щоб використати силу кожного методу. Гібридний підхід (hybrid approach) до класифікації включає використання системи на основі правил для класифікації основних завдань та моделі машинного навчання для більш складних випадків. Крім того, метод використання нейронних мереж з традиційними методами машинного навчання для підвищення точності.

Змішані підходи можуть підвищити точність і гнучкість, використовуючи різні переваги. Вони ефективні для предметно-спеціальних завдань, де легко застосовуються правила та шаблони. Але вони також складніші у впровадженні та обслуговуванні, обчислювально дороги, якщо поєднувати великі моделі.

Резюме методів класифікації тексту:

метод	плюси	мінуси	Загальні випадки використання
Методи на основі правил	Простий, зрозумілий	Обмежена масштабованість, негнучкість	Виявлення спаму, системи на основі ключових слів

(Rule-based Methods)			
Наївний Байєс	Швидко, добре працює з невеликими наборами даних	Припускає незалежність функцій	аналіз настроїв, виявлення спаму
SVM	Хороша продуктивність із великими даними	Повільно для великих наборів даних	Класифікація документів, тематичне моделювання
Логістична регресія	Простий, зрозумілий	Припускає лінійну роздільність	Аналіз настроїв, категоризація новин
KHH	Просто, не вимагає навчання	Обчислювально дорого	Категоризація тексту, визначення мови
CNN	Добре вловлює локальні відносини	Бореться з довготривалими залежностями	Аналіз настроїв, класифікація новин
RNN/LSTM	Захоплює наступні та довгострокові відносини	Обчислювально дорого	Наступна класифікація тексту, відповідь на питання
Трансформатори (наприклад, BERT)	Найсучасніше виконання	Вимагає значних обчислювальних ресурсів	Аналіз настроїв, класифікація тем
Ансамблеві методи	Покращує точність, зменшує переобладнання	Обчислювально дорого	Класифікація документів, виявлення спаму
Передача навчання	Висока точність з мінімальними даними, що стосуються конкретного завдання	Потрібні великі раніше підготовлені моделі	Аналіз настрою, предметно-спеціальна класифікація

ВИСНОВОК:

Вибір методу класифікації тексту залежить від складності завдання, розміру та характеру набору даних і доступних обчислювальних ресурсів. Традиційні методи машинного навчання, такі як Naive Bayes і SVM, все ще корисні для більш простих завдань, тоді як підходи до глибокого навчання, зокрема моделі на основі трансформаторів, такі як BERT, забезпечують найсучаснішу продуктивність для більш складних завдань.

Ось простий приклад того, як класифікувати текст за допомогою класифікатора Naive Bayes з бібліотекою scikit-learn. Це один із найбільш розширених підходів до

класифікації тексту, і ми будемо використовувати векторизатор TF-IDF для перетворення тексту в об'єкті.

Кроки:

1. **Встановіть scikit-learn, якщо у вас його ще немає:**

```
pip install scikit-learn
```

Приклад коду для класифікації тексту:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split
from sklearn import metrics

# Example data: texts and corresponding labels
texts = [
      "I love this movie, it's fantastic!",
      "This is the worst movie I have ever seen.",
      "Amazing acting, I really enjoyed this film.",
      "Terrible plot, I hated this movie.",
      "The story was great, highly recommend it!",
      "Awful, simply awful.",
      "Best movie I've watched all year.",
      "This movie was a waste of time."
]

labels = [
      "positive", "negative", "positive", "negative",
      "positive", "negative", "positive", "negative"
]

# Step 1: Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(texts, labels,
test_size=0.25, random_state=42)

# Step 2: Create a pipeline that combines the TF-IDF Vectorizer and Naive
Bayes Classifier
model = make_pipeline(TfidfVectorizer(), MultinomialNB())

# Step 3: Train the model with the training data
model.fit(X_train, y_train)

# Step 4: Predict the labels for the test set
predicted_labels = model.predict(X_test)

# Step 5: Evaluate the model's performance
```

```
accuracy = metrics.accuracy_score(y_test, predicted_labels)
print(f"Accuracy: {accuracy:.2f}")

# Step 6: Predict sentiment for new texts
new_texts = ["The movie was amazing!", "It was a terrible movie."]
predictions = model.predict(new_texts)
for text, sentiment in zip(new_texts, predictions):
      print(f"Text: {text} -> Predicted Sentiment: {sentiment}")
```

Пояснення коду:

1. Підготовка даних:

- `texts`: список прикладів речень із позитивними та негативними настроями.

- `labels`: відповідний список міток настрою ("позитивний" і "негативний").

2. Поїзд/тестовий розділ:

- Ми використовуємо `train_test_split,` щоб розділити набір даних на навчальний (75%) і тестовий (25%) набори.

3. Трубопровід:

- Створюється конвеєр, який складається з двох кроків :

1. `TfidfVectorizer`: перетворює текстові дані в численні функції за допомогою частотних термінів, зворотної частотної документації (TF-IDF).

2. `MultinomialNB`: Наївний класифікатор Байєса застосовано до функцій TF-IDF для класифікації тексту.

4. Навчання:

- Ми навчаємося моделі за допомогою навчальних даних (`X_train, y_train`).

5. Прогнозування та оцінка:

- Модель передбачає настрій для тестового набору (`X_test`), і ми оцінюємо точність за допомогою `metrics.accuracy_score` .

6. Прогноз нового тексту:

- Модель передбачає настрої щодо нових, ще не бачених текстів (наприклад, «Фільм був чудовий!» і «Це був жахливий фільм»).

Вихід:

```
Точність: 1,00

Текст: Фільм був чудовий! -> Прогнозований настрій: позитивний

Текст: Це був жахливий фільм. -> Прогнозований настрій: негативний

  Accuracy: 1.00

Text: The movie was amazing! -> Predicted Sentiment: positive

Text: It was a terrible movie. -> Predicted Sentiment: negative
```

Ключові моменти:

- **TF-IDF Vectorizer:** перетворює текст на числові характеристики, які відображають важливість слова в контексті набору даних.
- **Multinomial Naive Bayes:** простий, але ефективний класифікатор для текстових даних, який добре працює з окремими функціями (discrete features), такими як підрахунок слів (word counts) або результат TF-IDF.
- **Pipeline**: запускає процес шляхом об'єднання векторизатора та класифікатора.

У цьому прикладі демонструється базовий робочий процес для класифікації тексту за допомогою класифікатора Naive Bayes із scikit-learn. Ви можете розширити це, використовуючи більш просунуті моделі, такі як SVM, Random Forests або навіть методи глибокого навчання (deep learning methods).