유사도 기반 커널을 활용한 SVM 의 텍스트 분류 성능 개선

이상혁
성균관대학교 통계학과
leesanghyuk2000@gmail.com

1. 서 론

최근 인공지능 모델이 고도화됨에 따라 텍스트 분류(Text Classification) 분야는 Transformer, BERT 와 같은 복잡한 딥러닝(Deep Learning) 기반의 모델을 바탕으로 발전했다. 그러나 딥러닝의 경우 연산량이 많기 때문에 컴퓨팅 파워를 많이 소모한다. 이에 따라 전통적인 머신러닝 모델(ex. KNN, Logistic Regression, SVM 등)을 통해서 텍스트 분류 문제를 해결할 수 있다는 아이디어를 바탕으로 본 프로젝트를 진행하고자 한다. 이때 전통적인 머신러닝 모델은 자연어 처리(Natural Language Processing) 에 적합한 모델은 아니다. 따라서 자연어를 전통적인 머신러닝 모델의 입력 값으로 만들어주는 커널(Kernel)을 정의하고자 한다. 특히 커널을 기반으로 최적화를 거쳐 분류를 진행하는 모델인 Support Vector Machine 에 자연어의 특징을 반영할 수 있는 커널을 적용하여 텍스트 분류 성능을 개선해보고자 한다. 기존 연구에서는 문자열 커널(String Kernel) (Lodhi H et al. 2000)을 활용하여 SVM 의 텍스트 분류 성능을 개선했다. 본 프로젝트에서는 문자열 커널을 소개한 논문의 아이디어에 착안하여 이와 유사한 커널을 정의하고, 실험을 통해 정의한 커널을 적용한 SVM 의 텍스트 분류 성능을 실험하고자 한다.

2. 방 법

일반적으로 전통적인 머신러닝 모델로 텍스트 분류를 수행하기 위해서는 자연어를 벡터로 바꿔주는 과정이 필요하다. 텍스트 임베딩(Text Embedding) 혹은 TF-iDF 와 같은 방식이 주로 사용된다. 이때 커널을 적용하여 자연어의 특성을 반영하면서 이를 벡터로 만들 수 있다. 먼저 Hu, Lodhi et al. (2000)에서 제안한 문자열 커널의 경우는 다음과 같이 정의된다.

$$K(s,t) = \sum_{u \in \sum^k} \phi_u(s) \cdot \phi_u(t)$$

[수식 1] 문자열 커널

문자열 커널은 문서를 연속된 문자 시퀀스로 보고, 특정 길이의 서브시퀀스를 비교하여 문서 간 유사성을 측정하는 방법이다. 따라서 문자열 커널을 통해 두 문자열 간의 공통 서브시퀀스를 찾고, 그 빈도와 연속성을 고려하여 가중치를 부여하는 방법이다.

본 프로젝트에서는 참고한 문헌에서 제안된 문자열 커널의 아이디어를 착안하여 TF-IDF 행렬을 활용하는 가중 자카드 유사도 커널(Weighted Jaccard Similarity Kernel)을 활용하여 SVM 의 텍스트 분류 성능을 개선하고자 한다. 가중 자카드 유사도 커널 다음과 같이 정의하고자 한다.

$$K_{WJ}(x,y) = \frac{\sum_{i=1}^{n} \min(x_i,\ x_y)}{\sum_{i=1}^{n} \max(x_i,\ x_y)}$$

[수식 2] 가중 자카드 유사도 커널

이때 x_i 와 y_i 는 s 각각 TF-IDF 로 벡터화된 두 문서의 단어 빈도이다. 또한 자카드 유사도를 측정하기 위해 분모에는 두 벡터에서 각 단어의 최대값이, 분자에는 두 벡터에서 각 단어의 최소값이 오게 된다. 즉 전체 서브 시퀀스 중에서 공통 단어에 집중하여 이에 가중치를 부여하는 커널 함수이다.

자연어를 자카드 유사도 행렬로 매핑하는 예시는 다음과 같다.

문서 1	*"I love machine learning"*
문서 2	*"machine learning is fun"*
문서 3	*"deep learning and AI"*
문서 4	*"I love AI"*

[표 1] 문서 예시

[표 1]과 같이 4 개의 문서가 있다고 했을 때, TF-IDF 행렬은 다음과 같다.

ai	and	deep	fun	is	learning	love	machine
0.0	0.0	0.0	0.0	0.0	0.4968	0.6136	0.6136
0.0	0.0	0.0	0.5745	0.5745	0.3667	0.0	0.4530
0.4530	0.5745	0.5745	0.0	0.0	0.3667	0.0	0.0
0.7071	0.0	0.0	0.0	0.0	0.0	0.7071	0.0

[표 2] 4 개 문서에 대한 TF-IDF 행렬

TF-IDF 를 통해 각 문서를 벡터화하고, 가중 자카드 유사도 커널을 적용하여 다음과 같이 매핑한다.

	문서 1	문서 2	문서 3	문서 4
문서 1	1.0	0.2852	0.1102	0.2430
문서 2	0.2852	1.0	0.1026	0.0
문서 3	0.1102	0.1026	1.0	0.1546
문서 4	0.2430	0.0	0.1546	1.0

[수식 3] 가중 자카드 유사도 행렬 예시

이를 통해 단어의 중요도와 문서 사이의 관계를 반영하여 SVM 이 레이블과 문서의 관계를 학습하도록 하여 텍스트 분류 성능을 최적화하고자 한다.

3. 실험 계획

3-1. 실험 환경

CPU: Apple M2 (8 코어: 4 성능 + 4 효율)
GPU: Apple M2 (10 코어, 내장 GPU)
OS 버전: macOS Sequoia 15.2

3-1. 실험 1

첫번째 실험은 가중 자카드 유사도 커널의 적용한 SVM 의 다중 분류 성능을 검증한다. "scikit-learn"의 datasets API 에서 제공하는 "fetch_20newsgroups" 데이터셋을 활용한다. 해당 데이터셋은 약 18,000 개의 샘플로 이뤄져 있고, 20 개의 카테고리로 나뉘어져 있다. datasets API 공식 문서에서는 해당 데이터셋을 분류 문제에 적용할 경우, 몇가지 메타데이터를 제거하고 진행할 것을 권장하기 때문에 이 또한 반영한다. 단 컴퓨팅 자원의 한계로 학습 데이터는 클래스 비율을 고려하여 3,000 개를 랜덤샘플링하여 진행하고자 한다.

모델의 성능을 검증하기 위해 다음의 모델의 성능과 비교하고자 한다.

Target 모델	SVM with weighted jaccard similarity kernel
비교 모델	SVM with linear kernel
	Logistic Regression
	Multinomial Naive Bayes
	SGD Classifier
	K-Nearest Neighbors (K=3, 5)
	Decision Tree
	Random Forest

[표 3] 실험 1 에 대한 타겟&비교 모델

이때 비교 모델군의 경우 가중 자카드 유사도 커널의 효과를 검증해야 하기 때문에 텍스트 데이터에 TF-IDF 만 적용하여 벡터화를 진행한다.

각 모델 간의 성능을 비교하기 위해 "Accuracy", "Precision", "Recall", "F1 Score", "Training Time(s)", "Prediction Time(s)"를 지표로 활용한다.

3-2. 실험 2

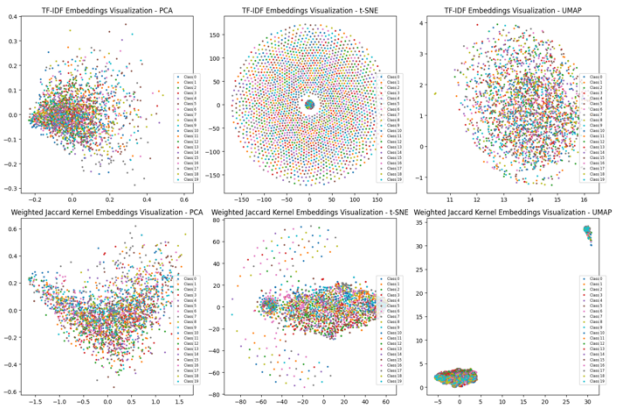
두번째 실험은 가중 자카드 유사도 커널의 적용한 SVM 의 이진 분류 성능을 검증한다 "keras"의 "imdb" 데이터셋을 활용한다. 해당 데이터셋은 영화 리뷰에 대한 긍/부정을 예측하는 데 활용되며 50,000 개의 데이터로 구성되어 있다. 또한 가장 빈도수가 높은 10,000 개의 단어로 학습 데이터를 구성한다. 단 컴퓨팅 자원의 한계로 빈도수와 학습 데이터는 클래스 비율을 고려하여 5,000 개를 랜덤샘플링하여 진행하고자 한다. 비교 모델과 성능 지표는 [실험 1]과 동일하다.

3-3. 실험 3

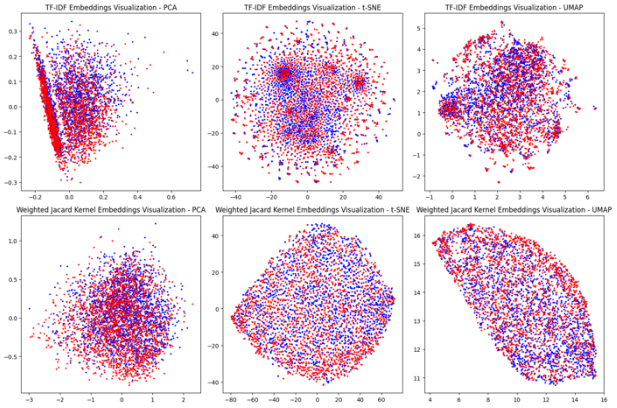
세번째 실험은 imdb 데이터셋의 빈도수가 높은 단어의 개수와 샘플 사이즈 변경하며 가중 자카드 유사도 커널의 성능이 가장 높을 때의 유사도 행렬의 형상, 샘플 사이즈 등을 분석하고자 한다. 성능 지표는 [실험 1]과 동일하다.

4. 실험 결과

본격적인 실험에 앞서서 데이터셋을 2 차원으로 차원 축소하여 시각화했다. 이를 통해 가중 자카드 유사도 커널을 적용하기 전후의 분리 가능성을 확인해봤다. 시각화는 1 행에는 TF-IDF 적용 후 PCA, TSNE, UMAP 으로 시각화한 결과를, 2 행에는 가중 자카드 유사도 커널을 적용 후 PCA, TSNE, UMAP 으로 시각화한 결과를 표시했다.



[그림 1] 20newsgroup 데이터셋 임베딩 시각화

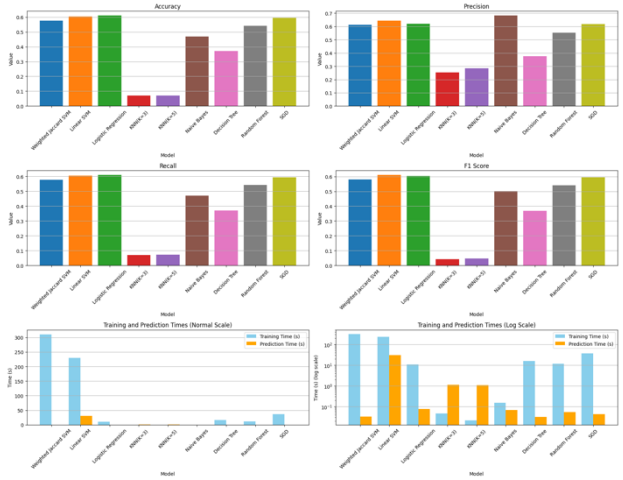


[그림 2] IMDB 데이터셋 임베딩 시각화

시각화를 통해서는 텍스트 분류 문제가 단순한 선형 분리에 적합한지 판단하기 어렵다고 생각한다. 따라서 직접 실험을 진행하며 다양한 모델과 가중 자카드 유사도 커널의 성능을 비교하여 다중/이진 분류의 성능을 직접적으로 비교하고자 한다.

4-1. 실험 1

실험 1 의 결과는 다음과 같다.



[그림 3] 20newsgroup 데이터셋 다중 분류 성능 비교

Model	Accuracy	Precision	Recall	F1 Score
Weighted Jaccard SVM	0.5767	0.6118	0.5767	0.5815
Linear SVM	0.6050	0.6427	0.6050	**0.6108**
Logistic Regression	**0.6100**	0.6210	**0.6100**	0.6046
KNN(K=3)	0.0700	0.2526	0.0700	0.0409
KNN(K=5)	0.0717	0.2858	0.0717	0.0465
Naive Bayes	0.4700	**0.6816**	0.4700	0.5023
Decision Tree	0.3700	0.3746	0.3700	0.3691
Random Forest	0.5417	0.5518	0.5417	0.5416
SGD	0.5950	0.6169	0.5950	0.5949

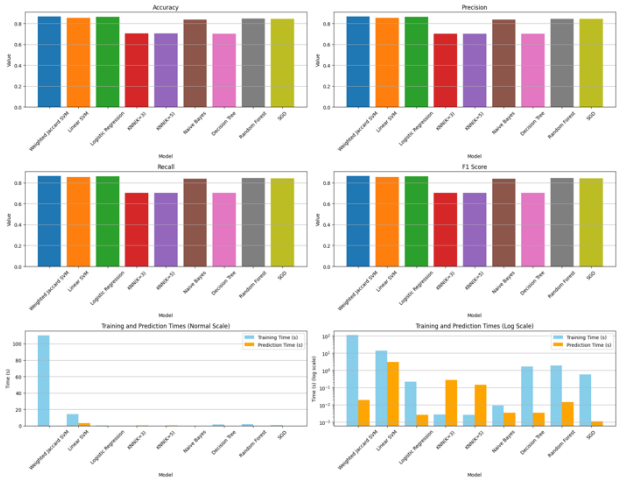
Model	Training Time (s)	Prediction Time (s)
Weighted Jaccard SVM	310.7459	0.0325
Linear SVM	229.8699	30.0676
Logistic Regression	10.6320	0.0757
KNN(K=3)	0.0457	1.1104
KNN(K=5)	**0.0213**	1.0626
Naive Bayes	0.1502	0.0685

Decision Tree	15.7754	**0.0319**
Random Forest	11.4961	0.0531
SGD	36.0964	0.0422

[표 4] 실험 1 에 대한 성능 결과

4-2. 실험 2

실험 2 의 결과는 다음과 같다.



[그림 4] IMDB 데이터셋 이진 분류 성능 비교

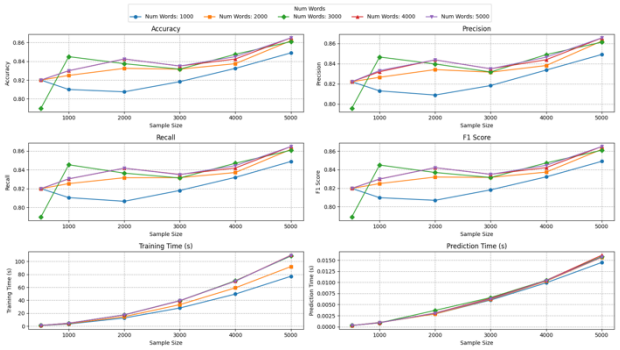
Model	Accuracy	Precision	Recall	F1 Score
Weighted Jaccard SVM	**0.8650**	**0.8655**	**0.8647**	**0.8648**
Linear SVM	0.8540	0.8540	0.8538	0.8539
Logistic Regression	0.8620	0.8622	0.8618	0.8619
KNN(K=3)	0.7030	0.7030	0.7031	0.7030
KNN(K=5)	0.7030	0.7033	0.7032	0.7030
Naive Bayes	0.8380	0.8380	0.8379	0.8379
Decision Tree	0.7020	0.7021	0.7021	0.7020
Random Forest	0.8450	0.8453	0.8453	0.8450
SGD	0.8430	0.8430	0.8429	0.8429

Model	Training Time (s)	Prediction Time (s)
Weighted Jaccard SVM	110.1964	0.0189
Linear SVM	14.1300	3.1175
Logistic Regression	0.2250	0.0027
KNN(K=3)	0.0027	0.2742
KNN(K=5)	**0.0026**	0.1454
Naive Bayes	0.0094	0.0035
Decision Tree	1.7195	0.0035
Random Forest	1.9304	0.0148
SGD	0.5881	**0.0011**

[표 5] 실험 2 에 대한 성능 결과

4-3. 실험 3

실험 3 의 결과는 다음과 같다.



[그림 5] 데이터 설정에 따른 가중 자카드 유사도 커널의 성능

sample_size	num_words	Accuracy	Precision	Recall	F1 Score	Training Time (s)	Prediction Time (s)
500	1000	0.8200	0.8221	0.8200	0.8197	**0.7933**	**0.000268**
	2000	0.8200	0.8221	0.8200	0.8197	0.9171	0.000271
	3000	0.7900	0.7958	0.7900	0.7890	1.0593	0.000288
	4000	0.8200	0.8221	0.8200	0.8197	1.0600	0.000296
	5000	0.8200	0.8221	0.8200	0.8197	1.0733	0.000283
1000	1000	0.8100	0.8130	0.8105	0.8097	3.1543	0.000973
	2000	0.8250	0.8265	0.8253	0.8249	3.6798	0.000957
	3000	0.8450	0.8465	0.8453	0.8449	4.3152	0.000876
	4000	0.8300	0.8320	0.8304	0.8298	4.4847	0.000889

	5000	0.8300	0.8331	0.8305	0.8297	4.2902	0.000913
2000	1000	0.8075	0.8089	0.8066	0.8069	12.5442	0.002893
	2000	0.8325	0.8341	0.8316	0.8319	14.6098	0.002892
	3000	0.8375	0.8396	0.8365	0.8369	17.4123	0.003682
	4000	0.8425	0.8437	0.8417	0.8421	17.2200	0.003095
	5000	0.8425	0.8437	0.8417	0.8421	17.3672	0.003052
3000	1000	0.8183	0.8183	0.8181	0.8182	27.9494	0.006006
	2000	0.8317	0.8315	0.8317	0.8316	33.1129	0.006139
	3000	0.8317	0.8317	0.8313	0.8315	39.2638	0.006564
	4000	0.8350	0.8349	0.8351	0.8349	39.0794	0.006394
	5000	0.8350	0.8349	0.8348	0.8349	39.3854	0.006248
4000	1000	0.8325	0.8336	0.8320	0.8322	49.5002	0.009894
	2000	0.8375	0.8381	0.8371	0.8373	59.1334	0.010266
	3000	0.8475	0.8489	0.8469	0.8472	70.0937	0.010465
	4000	0.8425	0.8439	0.8419	0.8421	69.5326	0.010420
	5000	0.8450	0.8464	0.8444	0.8446	69.3678	0.010393
5000	1000	0.8490	0.8491	0.8488	0.8489	76.9827	0.014473
	2000	0.8620	0.8624	0.8617	0.8619	92.0063	0.015621
	3000	0.8610	0.8613	0.8607	0.8609	108.3439	0.015808
	4000	**0.8650**	0.8654	**0.8647**	**0.8649**	109.5650	0.016100
	5000	**0.8650**	**0.8655**	**0.8647**	0.8648	109.2007	0.015676

[표 6] 실험 3 에 대한 성능 결과

5. 결론

5-1. 실험 1

Scikit-learn 의 20newsgroup 데이터셋을 활용한 다중 분류의 경우, F1 스코어는 Linear SVM 모델이 가장 높았다. 이어서 Logistic Regression, SGD, Weighted Jaccard SVM 순으로 높았다. 또한 SVM 모델이 성능은 모델 중에서 가장 높지만, Training Time(s)이 가장 오래 걸리는 모델로 확인할 수 있다. 이를 통해 모델의 성능과 학습 시간에는 Trade-off 가 존재함을 확인할 수 있다. 전반적으로 성능이 좋지 않은 KNN(K=5)와 Decision Tree 모델이 각각 Training Time(s)과 Prediction Time(s)에서 가장 우수한 성능을 보였다.

여기서 확인한 점은 성능이 가장 좋은 Linear SVM 의 경우에도 F1 스코어가 0.6 을 살짝 넘는 수준이라는 것이다. 이를 통해 전통적인 머신러닝 모델이 텍스트 다중 분류에 있어서는 효과적이지 않다는 것을 알 수 있었다. 이는 [그림 1]에서 확인한 부분으로 그 근거를 유추할 수 있을 것 같다. 클래스가 20 개이기 때문에 시각화에서도 명확한 패턴을 확인할 수가 없었다. 머신러닝 모델 또한 복잡한 데이터 속에서 명확한 패턴을 학습하지 못한 것으로 해석할 수 있다.

5-2. 실험 2

Keras 의 IMDB 데이터셋을 활용한 이진 분류의 경우, 모든 성능 지표에서 Weighted Jaccard SVM 이 가장 높았다. 또한 F1 스코어는 Weighted Jaccard SVM, Logistic Regression, Linear SVM 순으로 가장 높았다. 또한 SVM 계열의 모델의 성능이 마찬가지로 높았지만, Training Time(s)이 가장 긴 상위 2 개의 모델 또한 SVM 이다. 실험 1 과 마찬가지로 성능과 학습 시간에는 Trade-off 가 존재함을 확인할 수 있다.

그러나 Weighted Jaccard SVM 의 경우 Prediction Time(s)가 매우 짧게 나타났다. 성능 또한 모든 지표에서 가장 좋은 것으로 나타나고 있기 때문에 이진 분류 작업에서는 우수한 모델이라는 것을 확인할 수 있다. 기존 다중 분류보다는 문제가 단순해지면서 0 과 1 클래스 사이의 관계를 문서 간의 관계로 반영하여 패턴을 도출하여 적절하게 분류 작업을 수행할 수 있다고 생각한다.

5-3. 실험 3

Weighted Jaccard SVM 의 이진 분류 성능이 우수하게 도출되었기 때문에, 데이터셋의 크기를 조절하면서 그 성능을 확인해봤다. Training Time(s)과 Prediction

Time(s)의 경우 샘플 사이즈(sample_size)와 빈도수가 높은 단어 개수(num_words)가 커질수록 늘어나고 있다. 또한 대부분의 경우에서 샘플 사이즈와 빈도수가 높은 단어 개수가 많아질수록 성능 지표가 개선되고 있다. 이례적으로 샘플 사이즈가 1000 인 경우에는 빈도수가 높은 단어의 개수가 3000 개인 경우에 성능 지표가 4 가지 분야에서 모두 높게 나타나고 있다. 이외의 대부분의 경우에서는 약간의 역전 현상만 발생하고, 모두 일정하게 증가하는 모습을 보인다.

최종적으로 가중 자카드 유사도 커널은 이진 분류에 적합한 SVM 커널이며, 실험을 통해 그 효과를 입증할 수 있다는 것이 본 프로젝트의 성과로 뽑을 수 있다.

6. 자유 논의

6-1. 컴퓨팅 자원의 한계

Proposal 단계에서는 실험 1 과 2 모두 10,000 개의 데이터셋을 활용하고자 했다. 그러나 여러 번 코드를 실행하면서 10,000 개의 데이터셋에 대해 실험을 하는 것이 불가능 했다. 특히 가중 자카드 유사도 커널의 경우 다음과 같이 작동한다.

```
Function weighted_jaccard_kernel(X1, X2):
    Create a kernel_matrix with size (len(X1), len(X2)), initialized with zeros

    For each vector vec1 in X1 (indexed by i):
        For each vector vec2 in X2 (indexed by j):
            Compute the element-wise minimum between vec1 and vec2
            Compute the sum of the element-wise minimum values -> min_sum
            Compute the element-wise maximum between vec1 and vec2
            Compute the sum of the element-wise maximum values -> max_sum

            If max_sum is not zero:
                kernel_matrix[i][j] = min_sum / max_sum
            Else:
                kernel_matrix[i][j] = 0

    Return kernel_matrix
```

[표 7] Psedo Code for Weighted Jaccard Similarity Kernel

이 경우 계산 복잡도는 $O(n \cdot m \cdot d)$이다. 이때 n, m, d는 각각 $X1$의 행의 개수, $X2$의 행의 개수, 열의 개수가 된다.

따라서 대규모 데이터셋에 대해서 모델을 적합하는 데에 오랜 시간이 걸리기 때문에 데이터셋을 실험 1, 2 에 대해서 각각 3,000 개와 5,000 개로 축소하여 진행했다. 따라서 이 부분은 실험의 한계라고 생각된다. 특히 Weighted Jaccard SVM 의 이진 분류 성능이 모델군 중에서 가장 뛰어난 것으로 확인했지만 데이터셋의 크기가 매우 작아 일반적인 성능이라고 확인하는 데에는 어려움이 있다고 생각한다. 또한 데이터셋의 크기가 커질수록 Weighted Jaccard SVM 의 적합이 오래 걸리기 때문에 해당 커널의 한계점 또한 명확하다.

6-2. 병렬화를 통한 연산 속도 개선

향후 데이터셋의 규모를 확대하기 위해 커널 계산 과정을 병렬화하여 대규모 데이터셋에서도 Weighted Jaccard SVM 의 가능성을 탐구하는 방향으로 나아가고자 합니다.

6-3. 커널 트릭(Kernel Trick)을 활용한 자연어 처리

최근 트랜스포머 기반의 모델이 발전함에 따라 기초 머신러닝 모델에 대한 중요도가 떨어지고 있는 상황이다. 그러나 가중 자카드 유사도 커널을 통해 자연어로 이뤄진 문서의 특정을 벡터화하여 모델의 입력값으로 만드는 시도를 통해 머신러닝 모델의 성능을 개선할 수 있었다. 이는 SVM 을 배우는 주요한 목적 중 하나인 커널 트릭을 활용하여 성능 개선을 이뤄낸 것으로 큰 의미를 갖는다고 생각한다. 특히, Weighted Jaccard SVM 은 이진 분류에서 Logistic Regression, Linear SVM 을 비롯한 대부분의 모델보다 높은 F1 Score 를 기록하며, 텍스트 데이터에서 커널 트릭의 강점을 입증했다.

참고문헌

[1] Lodhi, H., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2000). Text classification using string kernels. Advances in neural information processing systems, 13.

실험 환경 및 코드

Python 버전: Python 3.11.5

Python 코드 및 라이브러리 버전: Github