문학 작품의 텍스트마이닝: 年代, 文體, 語彙 특징에 초점을 맞추어

朴鎭浩(서울대 국어국문학과)

인문학 연구의 전통적인 방법은 텍스트를 많이, 그리고 깊이 읽는 것이다. 이런 작업 가운데 어떤 것은 앞으로도 당분간은 인간 고유의 몫으로 남겠지만, 어떤 것은 인간보다 기계가 더 잘 할 수도 있다는 전망이 대두되고 있다. 인문학에서 다루는 텍스트의 상당 부분이 전산화되고 텍스트를 기계적으로 처리하는 기술이 발달함에 따라 이러한 전망이 더 구체화되고 있다. 본 발표에서는 텍스트마이닝 기술을 인문학 연구에 활용한 사례를 몇 가지 소개하고자 한다.

1. 韓國語史 자료의 연대 추정

韓國語史 자료 중 刊本이고 刊記가 있는 것은 간행 연대가 분명하여 적극적으로 이용되어 왔으나, 筆寫本은 대개 연대가 불분명하여 韓國語史 연구에 상대적으로 덜 사용되어 왔다. 그러나 필사본 자료의 양이 상당하므로, 어떻게든 연대를 추정하여 韓國語史 연구에 활용할 수 있으면 더 좋을 것이다.

韓國語史 전문가는 한국어의 역사에서 특정 시기에 특징적인 表記, 단어, 언어 현상 등을 단서로 삼아 연대를 추정한다. 그런데 연대 추정을 위해 고려해야 할 자질(feature)의 수가 너무 많기 때문에 이것을 일일이 규칙화하기는 어렵다. 최근 발전하고 있는 딥러닝은 신경망 모델이 알아서 자질을 찾아 주기 때문에 손쉽게 연대 추정 모델을 만들 수 있다.

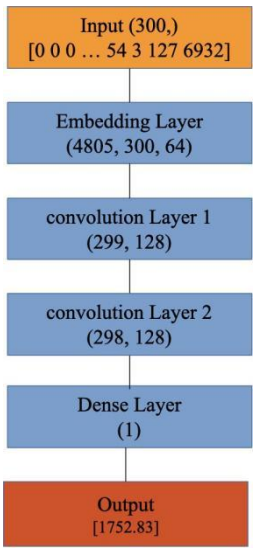
우선 연대를 알고 있는 한글 고문헌 자료 가운데 전산 입력이 되어 있는 것을 기초 자료로 하였다. 이 코퍼스를 일정한 길이(글자 수)로 잘라서 샘플로 만드는데, 하나의 샘플의 길이를 어느 정도로 할지가 문제이다. 하나의 샘플의 길이가 너무 짧으면 연대 추정을 위한 단서가 너무 적게 들어

있을 우려가 있다. 한편 하나의 샘플의 길이가 너무 길면, 신경망을 학습시키기 위한 훈련 자료의 수가 부족할 수 있다. 샘플의 길이를 달리하면서 실험을 해 본 결과 300자 정도로 하는 것이 적당하다고 결론을 내렸다.

샘플 길이	샘플 수	Training set (64%)	Validation set (16%)	Test set (20%)
100	145700	93248	23312	29140
200	72850	46624	11656	14570
300	48504	31042	7761	9701
350	41605	22627	6657	8321
400	36425	23312	5828	7285

<표 1> 국어사 자료의 샘플 길이와 샘플 수 사이의 trade-off 관계

<그림 1> 韓國語史 자료 연대 추정을 위한 CNN 모델의 구조



우선 Convolutional Neural Network(CNN)으로 실험하였다. 2000 epoch 동안 학습시킨 후, test set으로 성능을 측정한 결과 MAE(mean absolute error)가 22.36년으로 나왔다.

이 모델이 1446년~1460년 사이에 간행된 자료들의 연대를 추정할 때 중요한 근거로 삼은 feature들을 heat map으로 뽑아 보았더니, ‘ㅸ’, ‘ㅶ’ 같은 초성 글자들, ‘ ’, ‘ 卿’, ‘그 ’ 같은 표현들이 추출되었다. 韓國語史 전문가와 비슷한 판단 근거를 신경망도 사용하고 있는 것이다.

텍스트는 sequential data이므로 CNN보다는 RNN(recurrent neural network)이 더 적절하다. 그래서 2개의 convolution layer 대신에 1개의 LSTM(long short-term memory) layer(RNN layer의 일종)을 넣은 새로운 신경망으로 동일한 데이터를 학습시켜 보았다. 50 epoch 동안 모델을 훈련시킨 후, test set으로 모델의 성능을 측정하니 MAE가 10.34855년으로 나왔다. CNN 모델에 비해 loss가 절반 이하로 줄어들었다.

이 LSTM 모델을 15세기 문헌에 적용해 보았다(아래 그림 참조). 대체로 실제 연대와 가깝게 추정하고 있다.

〈그림 2〉 훈련된 LSTM 모델을 15세기 문헌에 적용한 결과

	1	2	3
1	문헌명	실제연대	추정연대
2	훈민정음해례	1446	1460.30
3	석보상절03	1447	1456.74
4	석보상절06-24	1447	1455.41
5	석보상절20	1447	1460.62
6	용비어천가	1447	1457.98
7	월인천강지곡	1447	1455.63
8	월인석보01-10_22	1459	1458.38
9	월인석보04	1459	1459.87
10	월인석보07	1459	1457.72
11	월인석보11-18	1459	1458.27
12	월인석보15	1459	1459.20
13	월인석보20	1459	1458.96
14	월인석보21	1459	1458.66
15	월인석보23	1459	1458.66
16	월인석보25	1459	1458.08
17	능엄경	1461	1460.41
18	법화경1	1463	1461.51
19	법화경2	1463	1461.42
20	법화경3-6	1463	1460.60
21	법화경7	1463	1460.47
22	금강경	1464	1461.12
23	반야심경	1464	1461.57
24	상원사중창권선문	1464	1464.66
25	선종영가집_상	1464	1461.79
26	선종영가집_하	1464	1465.29
27	아미타경	1464	1459.75
28	원각경	1465	1462.38
29	구급방언해	1466	1464.63
30	목우자수심결	1467	1465.74
31	몽산법어	1467	1478.02
32	내훈	1475	1478.02
33	두시언해03	1481	1483.05
34	두시언해05	1481	1503.84
35	두시언해06-10	1481	1484.35
36	두시언해09	1481	1482.13
37	두시언해11-15	1481	1486.55
38	두시언해16-19	1481	1486.11
39	두시언해20-22	1481	1481.10
40	두시언해23-25	1481	1484.06
41	삼강행실도_런던	1481	1476.09
42	금강경삼가해	1482	1473.82
43	남명집	1482	1478.54
44	불정심다라니경	1485	1470.32
45	구급간이방1-2	1489	1488.72
46	구급간이방3_6	1489	1485.94
47	구급간이방7	1489	1486.22
48	육조단경_상	1496	1495.94
49	육조단경_중	1496	1493.20
50	육조단경_하	1496	1497.09
51	진언권공	1496	1501.94
52	개간법화경	1500	1483.57
53	평균		1471.06

위의 모델을 韓國學中央研究院 藏書閣에 소장된 한글 筆寫本 문헌에 적용에 보았다.

〈그림 3〉 훈련된 모델을 장서각 소장 필사본 자료에 적용한 결과

	1	2		1	2		1	2
1	문헌명	추정연대	14	선보집략언해	1889.14	27	임신평난록	1846.02
2	계해반정록	1813.10	15	선택요람	1850.39	28	정미가례시일기	1735.48
3	고문백선	1775.74	16	션부군언행유사	1762.04	29	조야기문	1754.03
4	국조고사	1772.78	17	신미록	1763.65	30	조야첨재	1795.24
5	낙성비룡	1749.29	18	실록초본	1888.66	31	조야회통	1794.13
6	동뢰홀기	1885.67	19	엄씨효문청행록	1811.59	32	학석집(한글)	1835.28
7	동뢰홀기2	1793.40	20	열성지장통기	1754.76	33	한조삼성기봉	1814.73
8	명행정의록	1801.34	21	열성후비지문	1743.02	34	함녕전진표리홀기	1822.46
9	무오연행록	1775.83	22	완월회맹연	1797.45	35	현몽쌍룡기	1790.74
10	벽허담관제언록	1798.29	23	유씨삼대록	1774.83	36	홍경내전	1840.48
11	병자록	1735.41	24	유이양문록	1800.71	37	화씨충효록	1804.54
12	사문대의록	1749.08	25	육상궁묘현의	1632.83	38	화정선행록	1795.10
13	산성일기_병자	1789.86	26	윤하정삼문취록	1807.06	39	평균	1793.09

이 자료를 오랫동안 연구해 온 연구자들에 따르면, 이 모델의 연대 추정이 상당히 정확하다고 한다. 대체로 10개 중 8개 정도는 놀랄 만큼 정확하게 연대를 추정했다는 것이다. 그리고 연대 추정이 꽤 빗나간 나머지 2개도, 추정이 빗나간 이유를 알 수 있었다. 하나는 온전한 문장이 아니라 단어 목록 형식으로 된 문헌이어서, 연대 추정을 위한 단서가 매우 적었을 것으로 추정된다. 다른 하나는, 그 문헌 안에 어떤 역사적 사건이 언급되어 있어서 그 덕분에 이 문헌이 편찬된 연대의 상한선을 확정할 수 있었다.

그런데 이 문헌의 표기법이나 언어 현상은 매우 보수적이어서, 같은 시대의 다른 문헌들과 달리 'ㄷ' 구개음화 같은 현상이 반영되어 있지 않다.

신경망 모델은 역사적 사건에 대한 지식은 없으므로, 그저 表記法이나 언어 현상만 보고서 연대를 추정했으니, 훨씬 예전 문헌으로 추정한 것도 나름 합리적인 판단이었다고 할 수 있다.

2. 한국 근현대 소설의 문체 분석

우리가 소설을 읽다 보면 이 소설의 문체가 어떠하다는 인상을 받곤 한다. 문체적 특징 중에는 질적인 것도 있지만 수량화할 수 있는 것도 있다.

106개의 한국 근현대 소설을 선정하여 우선 형태소분석한 뒤, 다음의 5가지 계량적 문체 지표를 조사하였다.

① 連結語尾 대 轉成語尾 비율 (paratactic 대 hypotactic 비율)

② 동사 대 형용사 비율 (서사적 문체 대 묘사적 문체)

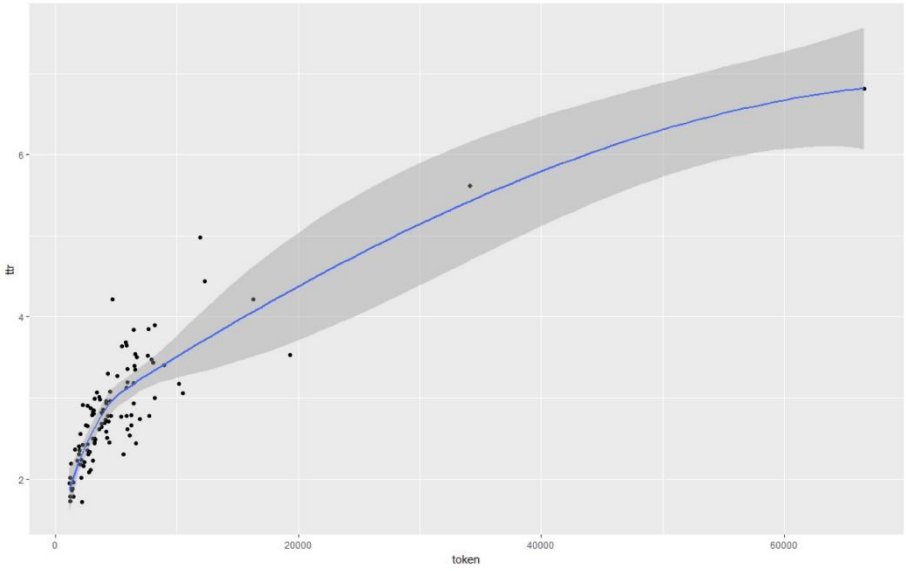
③ 보조용언 '있-' 비율 (서사적 문체 대 묘사적 문체)

④ 문장 길이(어절 수) (만연체 대 간결체)

⑤ TTR(type-token ratio)의 변형 (어휘 다양성)

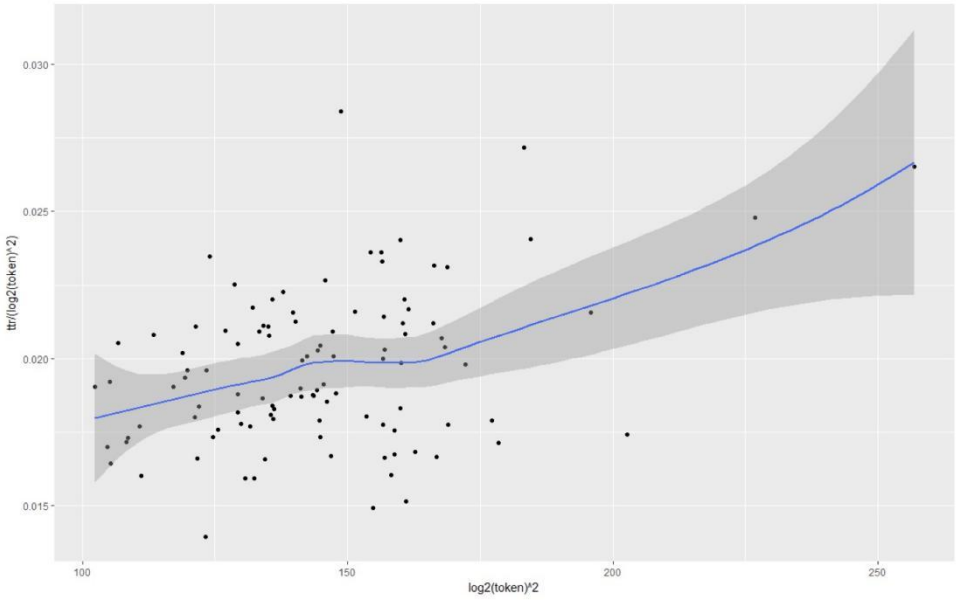
어휘 다양성을 측정하기 위해 가장 손쉽게 사용할 수 있는 지표는 타입-토큰 비율(TTR)이다. 그런데 TTR은 텍스트 크기의 영향을 많이 받는다는 문제가 있다.

<그림 4> 토큰 수(가로축)에 따른 TTR 값(세로축)



작품의 토큰 수가 증가함에 따라 TTR도 가파르게 증가하고 있다. 또한 관측값들이 원점 가까이에 몰려 있어 극도로 편향된 분포를 보이는데, 분포의 편향성을 완화하기 위해 X축에 로그 변형을 하고 TTR을 $\log(token)^2$으로 나눠주면 다음과 같이 된다. 이제 기울기가 거의 수평선에 가깝게 완만해졌고, 관측값들이 넓게 퍼져 있다. 어휘다양성의 지표로서 이 $TTR/\log(token)^2$ 값을 사용하기로 한다.

<그림 5> $\log(token)$(가로축)과 $TTR/\log(token)^2$(세로축)의 관계



106개 소설에 대한 통계 조사 표는 다음과 같다.

〈그림 6〉 106개 소설에 대한 통계 조사 표

```
import pandas as pd
df = pd.read_csv('novel.tsv', sep='\t')
df
```

✓ 1.3s

	author	title	pages	para	hypo	para_hypo	vb	adj	vb_adj	iss	iss_word	word	sent	sent_len	token	type	ttr	ttr2
0	강신재	젊은_느티나무	7.0	313	195	1.61	346	157	2.20	25	0.016858	1483	131	11.32	1476	754	1.96	17.66
1	계용묵	백치_아다다	13.0	795	579	1.37	1006	299	3.36	10	0.002706	3695	288	12.83	3780	1432	2.64	18.69
2	김동리	등신불	15.0	831	583	1.43	992	283	3.51	50	0.012355	4047	267	15.16	4151	1417	2.93	20.28
3	김동리	바위	7.0	382	227	1.68	495	124	3.99	22	0.011506	1912	155	12.34	1946	842	2.31	19.36
4	김동리	역마	18.0	924	567	1.63	1102	297	3.71	67	0.014839	4515	247	18.28	4574	1648	2.78	18.77
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
101	황순원	독짓는_늙은이	6.0	483	298	1.62	610	131	4.66	22	0.010005	2199	159	13.83	2252	775	2.91	23.43
102	황순원	목넘이_마을의_개	17.0	1215	962	1.26	1631	433	3.77	56	0.008906	6288	421	14.94	6405	1669	3.84	24.00
103	황순원	별	7.0	603	410	1.47	779	214	3.64	42	0.014228	2952	234	12.62	2882	1003	2.87	21.75
104	황순원	소나기	7.0	346	197	1.76	523	146	3.58	37	0.018272	2025	275	7.36	1976	840	2.35	19.62
105	황순원	학	3.0	177	121	1.46	280	47	5.96	13	0.012116	1073	118	9.09	1114	571	1.95	19.04

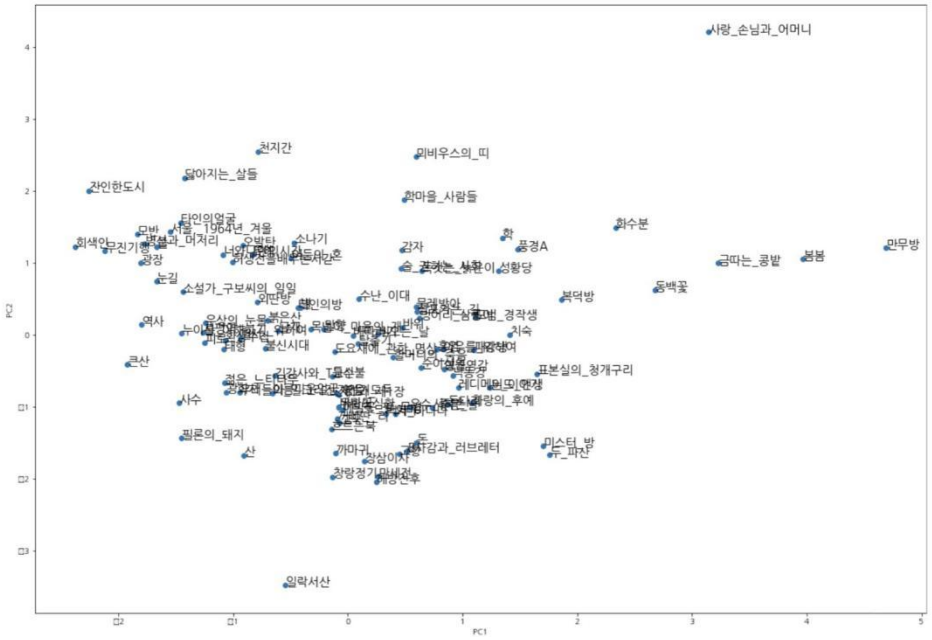
106 rows × 18 columns

여기서 para_hypo(연결어미-전성어미 비율), vb_adj(동사-형용사 비율), iss_word(보조용언 '있-' 비율), sent_len(평균 문장 길이), ttr2(TTR의 변형)의 5개 컬럼만 추출한 뒤, standard scaling을 실시하였다.

그 다음에 차원 축소를 실시하였다. 인간은 2차원 평면에서 관측점들의 분포 패턴은 쉽게 파악할 수 있으나 4차원 이상의 고차원 공간에 대한 직관이나 파악 능력은 형편없이 떨어진다. 따라서 다변량 데이터를 2차원으로 축소해서 시각화하는 것이 좋다. 차원을 축소하다 보면 데이터가 가지

고 있는 정보 중 일부가 손실되게 마련이다. 정보 손실을 최소화하면서 차원을 축소하는 다양한 기법이 개발되어 있는데, 주성분분석(PCA)도 그 중 하나이다. 106개 소설의 5개 문체 변수에 PCA를 적용한 결과는 다음과 같다.

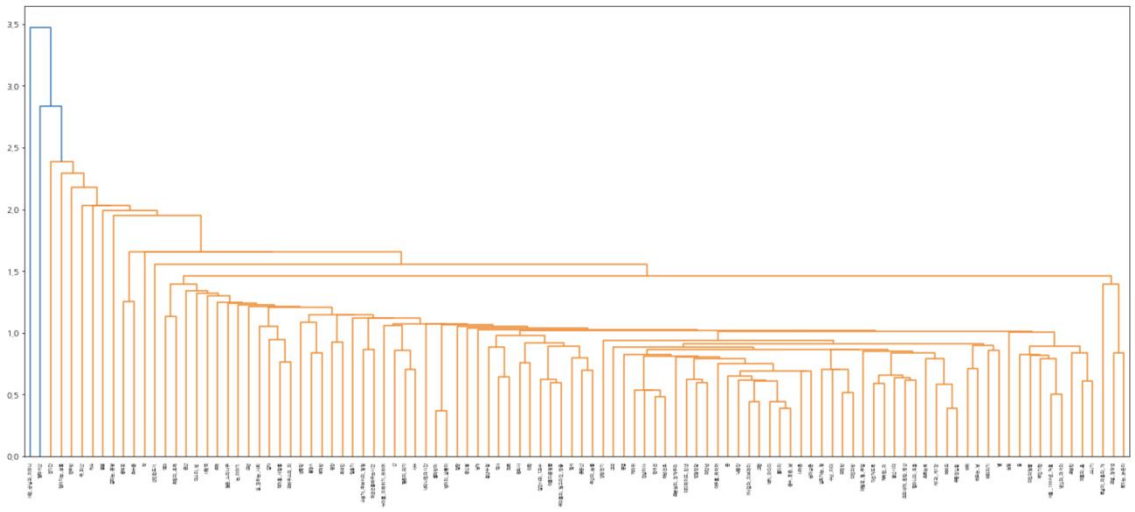
<그림 7> 106개 소설의 5개 문체 변수에 PCA를 적용한 결과



‘사랑손님과 어머니’는 1인칭 관찰자 시점의 소설로서, 어린 아이의 관점에서 묘사되어 있기 때문에 문체가 특이하게 나타났다. ‘일락서산’도 다른 작품들과 사뭇 다른 문체로 나타났으며, 김유정의 작품들이 다른 작가의 작품들과는 다른 특이성을 보이면서도 자기들끼리는 상당히 인접해서 나타나고 있는 것이 눈길을 끈다. 요컨대, 이러한 다변량 통계 분석을 통해, 다수의 변수에 대한 측정값이 비슷한 개체들을 한눈에 볼 수 있다.

다음으로 위계적 군집분석을 실시하였다.

<그림 8> 106개 소설에 대한 위계적 군집분석 결과



위와 같이 군집분석의 결과로서 얻어지는 plot을 dendrogram이라고 한다. 여기서 어떤 개체와 어떤 개체가 긴밀하게 묶이는지 알 수 있고, 둘 이상의 개체/군집을 연결하는 수평선의 높이가 유사도를 반영하므로 특정 지점에서 수평선을 그음으로써 전체 데이터를 일정한 개수의 군집으로 나눌 때 어떻게 나뉘는지도 알 수 있다.

3. 杜甫 시의 특징적 표현

중국 한시의 역사에서 당대(唐代)가 전성기라 할 만하며, 이백과 더불어 두보는 당시를 대표하는 시인이다. 중국뿐 아니라 조선에서도 두보의 시가 애송되었으며, 15세기에 두보의 시 거의 전부를 한국어로 번역하여 "杜詩諺解"가 간행되기도 했다. 필자는 "杜詩諺解"를 읽으면서 두보의 시가 지닌 분위기를 어느 정도 직관적으로 느낄 수 있게 되었고, 두보의 시에 자주 나타나는 특징적인 표현들에 대해서도 약간을 감을 잡게 되었다. 그런데 이러한 주관적 감각을 좀 더 객관적으로 알아보고 검증할 수 있으면 좋을 것이다. 이를 위해 수행한 작업 과정을 아래에 소개한다.

"詩詞名句網"이라는 웹사이트에 올라 있는 약 29만 2천 수의 한시 데이터를 웹크롤링으로 수집하였다. 수집된 자료는 다음과 같은 테이블 형태로 정리하였다.

〈그림 9〉 詩詞名句網에서 수집한 한시 데이터 테이블

```
import pandas as pd
poems = pd.read_csv("all.tsv", sep="\t")
poems
```

	poet_id	poet	poem_serial	title	poem
0	1	李白	1	将进酒·君不见黄河之水天上来	君不见黄河之水天上来，奔流到海不复回·君不见高堂明镜悲白发，朝如青丝暮成雪·人生得意须尽欢，...
1	1	李白	2	静夜思	床前明月光，疑是地上霜·举头望明月，低头思故乡。
2	1	李白	3	蜀道难	噫吁嚱，危乎高哉！蜀道之难，难于上青天！蚕丛及鱼凫，开国何茫然！尔来四万八千岁，不与秦塞通人...
3	1	李白	4	梦游天姥吟留别	海客谈瀛洲，烟涛微茫信难求·越人语天姥，云霞明灭或可睹·天姥连天向天横，势拔五岳掩赤城·天台...
4	1	李白	5	望庐山瀑布	日照香炉生紫烟，遥看瀑布挂前川·飞流直下三千尺，疑是银河落九天。
...	...	...	...	...	...
291929	13073	X	435	杂歌谣辞·得宝歌	得宝弘农野，弘农得宝那。潭里船车闹，扬州铜器多·三郎当殿坐，听唱得宝歌。
291930	13073	X	440	与诸公送陈郎将归衡阳	Y
291931	13073	X	627	画	远观山有色，近听水无声·春去花还在，人来鸟不惊。
291932	13073	X	861	暮冬阌乡遇萧霸赴任	Y
291933	13086	聂冠卿	1	多丽(李良定公席上赋)	想人生，美景良辰堪惜。闻其间、赏心乐事，就中难事并得。况东城、凤台沙苑，泛清波、浅照金碧。露...

291934 rows × 5 columns

두보 시의 특징적인 표현을 알아보려고 할 때, 단순히 빈도가 높은 것을 뽑는 것은 문제가 있다. 그 표현이 두보의 작품뿐 아니라 다른 사람의 작품에서도 빈도가 높을 수 있기 때문이다. 따라서 어떤 단어의 중요도를 알아볼 때, 절대 빈도뿐 아니라 데이터 전체에서의 빈도도 함께 고려할 필요가 있다. 이러한 아이디어는 정보검색 분야에서 일찍부터 주목을 받아 다듬어져 왔는데, TF-IDF가 대표적인 지표이다. TF-IDF는 특정 단어가 특정 문서에 얼마나 특징적인가를 나타낸다.

$$idf(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

t: 단어, D: 문서들의 집합, d: 문서

그런데 두보 시에 특징적인 표현은 漢字 한 글자일 수도 있고 두 글자 연쇄일 수도 있다. 따라서 이 TF-IDF 값을 unigram(하나의 토큰)에 대해서도 구하고 bigram(두 토큰 연쇄)에 대해서도 구할 필요가 있다.

<그림 11> 29만 2천개 한시에서 약 12000개 한자 unigram의 TF-IDF

	...	虢	諚	一	丁	七	万	丈	三	上	...	龑	龙	龚	龛	龟	龠	龡	龢	poet
0	0.0	0.0	0.0	0.066687	0.0	0.0	0.049475	0.000000	0.048357	0.044067	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	李白
1	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.152979	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	李白
2	0.0	0.0	0.0	0.042317	0.0	0.0	0.094186	0.000000	0.000000	0.111855	...	0.0	0.037661	0.0	0.0	0.0	0.0	0.0	0.0	李白
3	0.0	0.0	0.0	0.021165	0.0	0.0	0.094213	0.052879	0.000000	0.000000	...	0.0	0.037672	0.0	0.0	0.0	0.0	0.0	0.0	李白
4	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.142360	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	李白
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
291929	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.093212	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	X
291930	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	X
291931	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	X
291932	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	X
291933	0.0	0.0	0.0	0.037751	0.0	0.0	0.000000	0.000000	0.000000	0.000000	...	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	聂冠卿

291934 rows × 12128 columns

<그림 12> 29만 2천개 한시에서 약 270만개 한자 bigram의 TF-IDF

	.....朝	虢蝓	諚我	諚杀	諚风	一一	一丁	一七	一万	一丈	...	龠龙	龡笙	龡豳	龡邹	龢其	龢弗	龢意	龢气	龢英	poet
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	李白
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	李白
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	李白
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	李白
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	李白
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
291929	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	X
291930	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	X
291931	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	X
291932	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	X
291933	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	聂冠卿

934 rows × 2698432 columns

너무 많은 unigram, bigram을 다 고려하기는 어려우므로, 두보의 시에서 TF-IDF 값이 높은 것 100개씩만을 추출했다.

<그림 13> ungram 중 두보 시에서 TF-IDF 값이 높은 것 100개

```
dufu = dfx.loc[poems.poet=='杜甫']
dufu2 = dufu.drop('poet', axis=1)
kws = dufu2.sum().sort_values(ascending=False)
kws100 = kws[:100]
kws100.index
```

```
Index(['江', '不', '日', '人', '白', '云', '山', '风', '无', '有', '天', '时', '何', '马',
       '老', '高', '水', '客', '子', '见', '春', '长', '为', '生', '青', '秋', '城', '相',
       '来', '自', '行', '一', '归', '多', '未', '花', '中', '清', '寒', '落', '地', '已',
       '远', '年', '今', '门', '南', '峡', '君', '万', '出', '回', '知', '闻', '此', '得',
       '在', '北', '复', '如', '故', '上', '雨', '病', '西', '愁', '东', '舟', '书', '里',
       '王', '野', '暮', '朝', '下', '色', '夜', '意', '乱', '开', '道', '深', '鸟', '草',
       '我', '身', '月', '入', '气', '黄', '将', '过', '公', '独', '空', '石', '近', '家',
       '飞', '尽'],
      dtype='object')
```

<그림 14> bigram 중 두보 시에서 TF-IDF 값이 높은 것 100개

```
dufu = dfx.loc[poems.poet=='杜甫']
dufu2 = dufu.drop('poet', axis=1)
kws = dufu2.sum().sort_values(ascending=False)
kws100 = kws[:100]
kws100.index
```

```
Index(['万里', '风尘', '白帝', '回首', '巫峡', '白头', '不见', '故人', '乾坤', '江汉', '干戈', '江边',
       '老夫', '江湖', '何时', '戎马', '朝廷', '落日', '天寒', '舟楫', '不可', '今日', '蛟龙', '秋风',
       '将军', '柴门', '老翁', '颜色', '天下', '天地', '清秋', '何处', '时危', '不复', '丧乱', '相见',
       '寂寞', '消息', '盗贼', '巫山', '十年', '闻道', '为客', '几时', '多病', '萧萧', '白首', '白日',
       '孤城', '老病', '江山', '洞庭', '杖藜', '风吹', '君不', '白发', '不得', '衰年', '文章', '天子',
       '艰难', '兵戈', '如何', '清江', '江上', '迟暮', '草堂', '春色', '峡中', '帝城', '卧病', '高秋',
       '群盗', '道路', '至今', '中原', '人生', '萧条', '平生', '他乡', '浮云', '安得', '三峡', '不知',
       '故园', '战伐', '江城', '江流', '骑马', '白马', '衰谢', '百年', '衣裳', '寒江', '宾客', '长安',
       '丈人', '主人', '麒麟', '怅望'],
      dtype='object')
```

이렇게 두보 시에서 TF-IDF 값이 높은 것들을 뽑음으로써, 두보 시의 특징적인 표현들을 어느 정도는 알 수 있으나, 여기에 만족하지 않고 기계학습 기법을 적용해 보기로 한다. 唐代의 시인 중 500수 이상의 작품이 있는 시인 12명을 뽑아서, unigram과 bigram 200개의 변수를 바탕으로 두보의 작품과 그 외의 시인의 작품을 구별하는 기계학습 모델을 만들어 본다. 이 모델이 어느 정도의 정확도를 보일 경우, 이 모델이 입력 작품을 두보의 작품으로 판단할 때 어떤 변수를 중요한 단서로 활용했는지 알아보고자 한다.

우선 100개의 unigram을 포함한 테이블과 100개의 bigram을 포함한 테이블을 하나로 통합한다.

<그림 15> 100개 unigram 테이블과 100개 bigram 테이블을 통합한 결과

	江	不	日	人	白	云	山	风	无	有	...	百年	衣裳	寒江	宾客	长安	丈人	主人	麒麟	怅望	poet
0	0.0	0.150160	0.000000	0.064322	0.048344	0.000000	0.000000	0.000000	0.000000	0.075076	...	0.0	0.0	0.0	0.0	0.0	0.0	0.046548	0.0	0.0	李白
1	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	李白
2	0.0	0.095287	0.025038	0.081633	0.030678	0.025289	0.045700	0.000000	0.000000	0.071461	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	李白
3	0.0	0.057188	0.050091	0.040828	0.030686	0.101184	0.022857	0.021502	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	李白
4	0.0	0.000000	0.116160	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	李白
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
291929	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	X
291930	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	X
291931	0.0	0.132354	0.000000	0.141736	0.000000	0.000000	0.158696	0.000000	0.155514	0.165433	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	X
291932	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	X
291933	0.0	0.000000	0.000000	0.036412	0.000000	0.045120	0.000000	0.000000	0.039951	0.042500	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	聂冠卿

291934 rows × 201 columns

여기서 12명의 唐代 시인의 작품만 추출한다.

<그림 16> 당대 시인 12명 데이터만 추출한 테이블

```
poets = ['杜甫', '李白', '白居易', '刘禹锡', '元稹', '李商隐', '韦应物', '杜牧', '刘长卿', '贯休', '齐己', '徐铉'] # 500수 이상의 작품이 있는 당대 시인
idx = [poet in poets for poet in df.poet]
df_tang = df.loc[idx,:]
df_tang
```

Python

	江	不	日	人	白	云	山	风	无	有	...	百年	衣裳	寒江	宾客	长安	丈人	主人	麒麟	怅望	poet
0	0.0	0.150160	0.000000	0.064322	0.048344	0.000000	0.000000	0.000000	0.000000	0.075076	...	0.0	0.0	0.0	0.0	0.0	0.0	0.046548	0.0	0.0	李白
1	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	李白
2	0.0	0.095287	0.025038	0.081633	0.030678	0.025289	0.045700	0.000000	0.000000	0.071461	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	李白
3	0.0	0.057188	0.050091	0.040828	0.030686	0.101184	0.022857	0.021502	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	李白
4	0.0	0.000000	0.116160	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	李白
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
180284	0.0	0.000000	0.000000	0.092689	0.000000	0.000000	0.000000	0.000000	0.101699	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	徐铉
180285	0.0	0.000000	0.000000	0.098146	0.073766	0.060808	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	徐铉
180286	0.0	0.000000	0.000000	0.175016	0.131542	0.000000	0.000000	0.092173	0.000000	0.102138	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	徐铉
180287	0.0	0.130004	0.085403	0.000000	0.000000	0.086257	0.000000	0.073321	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	徐铉
180288	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	徐铉

10256 rows × 201 columns

본고는 두보 시의 특징에만 관심이 있으므로, 12명의 시인을 모두 구별할 필요는 없고, 두보의 시인지 다른 시인의 시인지만 구분하면 된다. 즉 이진 분류(binary classification) 과제라 할 수 있다. PyCaret 라이브러리를 실행한 결과는 다음과 같다.

<그림 17> 두보 시와 다른 시인의 시를 분류하는 이진 분류 과제에 PyCaret을 적용한 결과

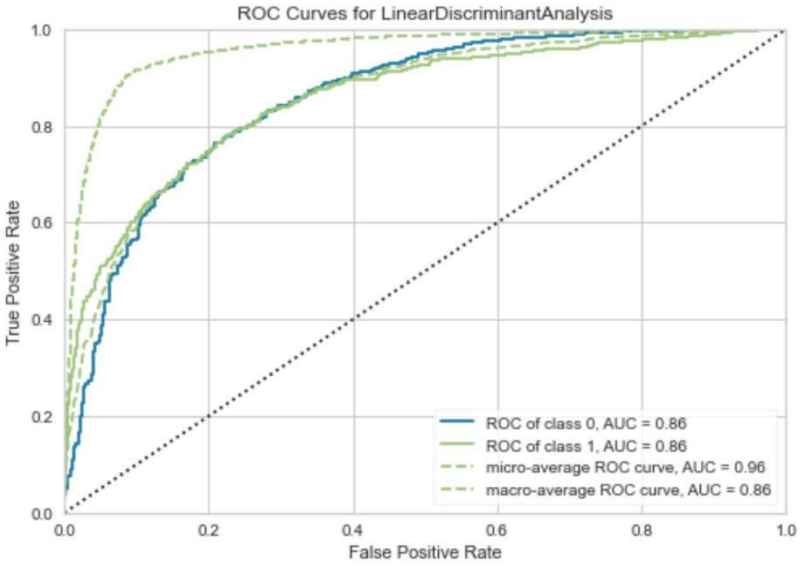
```
compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lda	Linear Discriminant Analysis	0.9109	0.8483	0.4252	0.6776	0.5207	0.4745	0.4911	0.0970
catboost	CatBoost Classifier	0.9107	0.8543	0.2656	0.8553	0.4032	0.3694	0.4450	5.0400
xgboost	Extreme Gradient Boosting	0.9061	0.8363	0.2826	0.7392	0.4071	0.3670	0.4184	1.1280
lightgbm	Light Gradient Boosting Machine	0.9060	0.8393	0.2619	0.7579	0.3879	0.3500	0.4092	0.7210
gbc	Gradient Boosting Classifier	0.9019	0.8291	0.1937	0.7969	0.3103	0.2781	0.3613	0.4450
ada	Ada Boost Classifier	0.9000	0.8033	0.2875	0.6410	0.3949	0.3492	0.3836	0.1220
et	Extra Trees Classifier	0.9000	0.8330	0.1632	0.8057	0.2694	0.2410	0.3325	0.2420
rf	Random Forest Classifier	0.8975	0.8296	0.1242	0.8616	0.2150	0.1922	0.3006	0.2890
ridge	Ridge Classifier	0.8908	0.0000	0.0585	0.8180	0.1076	0.0942	0.1931	0.1050
qda	Quadratic Discriminant Analysis	0.8893	0.8085	0.4495	0.5220	0.4809	0.4195	0.4222	0.0720
lr	Logistic Regression	0.8873	0.7952	0.0195	0.6933	0.0377	0.0326	0.1035	0.0440
svm	SVM - Linear Kernel	0.8859	0.0000	0.0049	0.3333	0.0096	0.0080	0.0357	0.1000
knn	K Neighbors Classifier	0.8851	0.5969	0.0183	0.4381	0.0348	0.0261	0.0700	0.1800
nb	Naive Bayes	0.8671	0.8179	0.5177	0.4352	0.4717	0.3966	0.3992	0.1070
dt	Decision Tree Classifier	0.8440	0.6186	0.3264	0.3226	0.3235	0.2356	0.2361	0.1810

```
LinearDiscriminantAnalysis(covariance_estimator=None, n_components=None,
                           priors=None, shrinkage=None, solver='svd',
                           store_covariance=False, tol=0.0001)
```

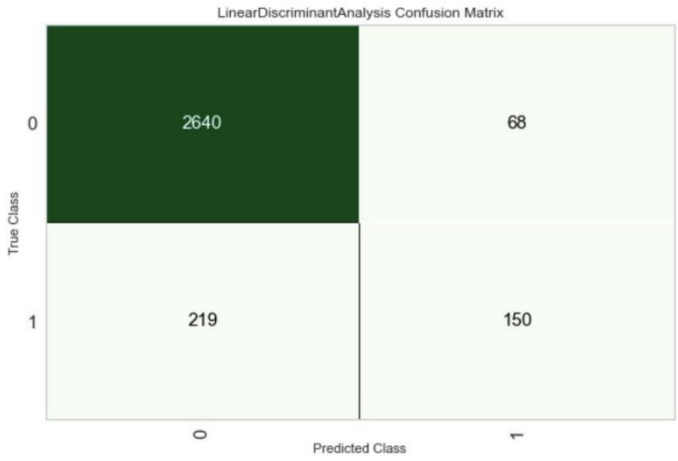
가장 성능이 좋은 선형판별분석(LDA) 모델의 ROC 곡선은 다음곽 같다. 랜덤한 판단(좌하-우상 대각선)에 비해 ROC 곡선이 좌상 꼭지점에 가까울수록 좋은 것이다.

<그림 18> 두보 식 판별을 위한 LDA 모델의 ROC 곡선



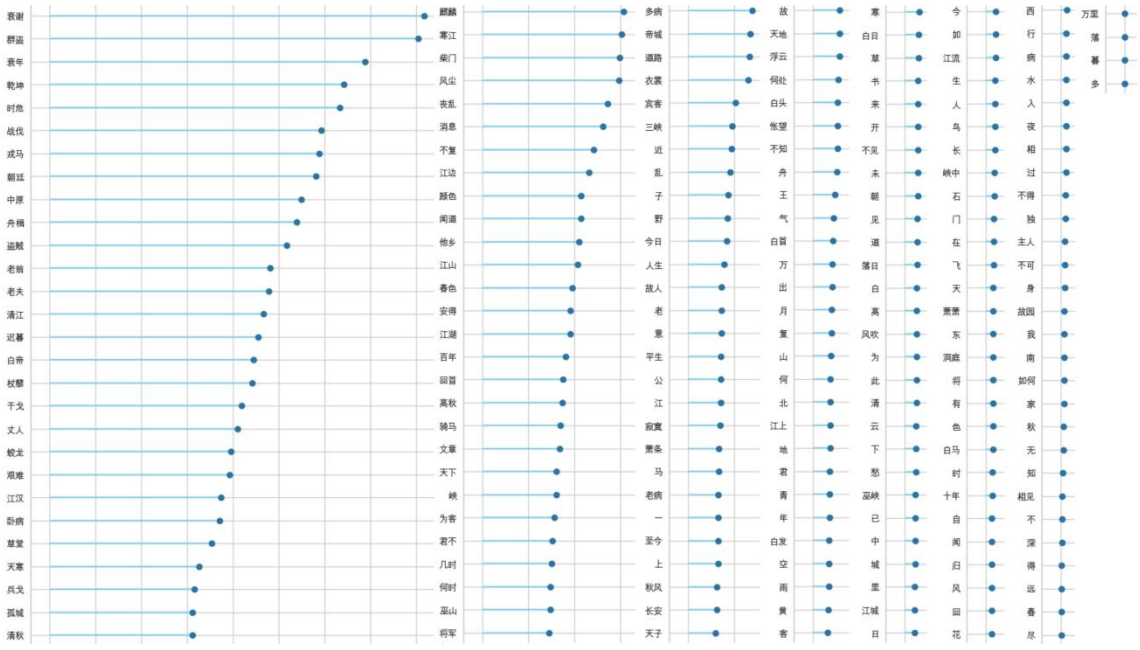
혼동행렬(confusion matrix)은 아래와 같다. 두보 아닌 사람의 시를 두보 시로 오인한 경우는 적으나(68회), 두보의 시를 잡아내지 못하고 놓친 경우는 꽤 많다(219회). 이렇게 모델이 두보의 시임을 알아채지 못하고 놓친 작품들에 대해서는 두 가지 해석이 가능하다. 첫째, 해당 작품에 두보의 특징이 별로 드러나지 않았을 가능성이 있다. 두보가 아무리 위대한 시인이라 하더라도 모든 작품에서 자신의 개성을 분명히 드러내는 것은 아닐 것이다. 두보의 특징이 별로 드러나지 않은 작품에 대해 모델은 두보의 작품이 아니라고 판단하는 것도 이해할 만하다. 둘째, 모델이 아직 두보의 특징을 충분히 포착하지 못하고 있을 가능성이 있다. TF-IDF를 바탕으로 하여 unigram 100개, bigram 100개를 사용했는데, 두보 시의 특징을 포착하기에 역부족이었을 수 있다.

<그림 19> 두보 식 판별을 위한 LDA 모델의 혼동행렬



다음으로 모델이 200개의 변수 중 두보 시를 판별하기 위해 어느 변수를 중요하게 사용했는지를 알아보자. PyCaret이 제공하는 Feature Importance Plot이 이 용도에 제격이다.

<그림 20> 두보 시 판별을 위한 LDA 모델의 Feature Importance Plot



수십 개의 bigram이 최상위에 랭크돼 있고, unigram도 그보다 하위이기는 하지만 수십 개를 찾아 주었다. 두보의 시를 많이 읽어본 사람이 질적

으로 평가해도 대체로 수긍할 만한 것들이다. 아무래도 unigram보다는 bigram이 특정 작자의 개성을 포착하는 데는 더 유용하다. trigram, four-gram 등 더 많은 변수들을 집어넣어 모델을 키우면 성능도 더 좋아질 것이고, 두보 시의 특징적 표현들을 더 많이 찾아낼 수 있을 것이다.