Lecture 4: Text Representation I Count-based Representations QA

Pilsung Kang

School of Industrial Management Engineering

Korea University

Q1 (황성진)

3월 8일 월요일

황성진 오후 6:41

Topic 4: Text Representation I: Classic Methods 의 Bag of words 중 Term-Document Matrix 부분에서의 질문입니다.
Frequency representation의 경우 TF-IDF를 통해 가중치가 높은 단어를 골라내는 방식은 이해했으나
Binary의 경우 0/1의 값만을 가지고 어떻게 단어의 중요도(가중치)를 판정할지 직관적으로는 이해하기 어려워..
혹시 binary가 더 유효한 사례가 있다면 어떤 경우가 있는지 질문드리고 싶습니다. (편집됨)

2개의 답글 18일 전 마지막 답글

AI-I (김지나)

김지나 19일 전

안녕하세요. 좋은 질문 감사합니다.

말씀하신 대로 단어의 등장 여부에 따른 binary representation은 특정 document에서의 단어의 중요도를 표현하기에는 부족할 수 있다는 생각이 듭니다. 이보다는 TF-IDF 같은 빈도수 기반의 가중치가 더 많은 정보를 담고 있기 때문에 더 적절하다고 할 수 있을 것 같습니다.

하지만 binary term-document incidence matrix는 단순한 information retrieval에 쓰일 수 있습니다. information retrieval이란, 방대한 정보로부터 원하는 내용과 관련 있는 정보를 찾는 것으로, 셰익스피어의 작품집에서 Brutus와 Caesar를 포함하고, Calpurnia를 포함하지 않는 작품을 찾는 task(이러한 task를 boolean retrieval이라고 합니다.)를 예로 들 수 있습니다. 이를 위해서 컴퓨터가 셰익스피어의 모든 작품을 처음부터 linear하게 읽게 할 수 있지만 이는 매우 비효율적이므로, 이때 간단한 이진 연산만으로 문제를 해결할 수 있는 binary term-document incidence matrix를 사용합니다.

셰익스피어의 작품집을 첨부한 이미지의 binary term-document incidence matrix로 나타내면,
"셰익스피어의 작품 중 Brutus와 Caesar를 포함하고, Calpurnia를 포함하지 않는 작품"이라는 질문에 다음과 같은 간단한 비트 연산을 통해 답할 수 있게 됩니다.
110100 AND 110111 AND 101111 = 100100
100100에 해당하는 답은 "Antony and Cleopatra", "Hamlet"입니다.

이와 같이 binary term-document matrix는 boolean retrieval에서 이진 연산이 가능하기 때문에, 빈도수 기반의 matrix보다 빠르게 검색을 수행할 수 있다는 장점이 있습니다.

혹시 틀린 부분이 있다면 댓글로 알려주시면 감사하겠습니다.

참고자료: https://nlp.stanford.edu/IR-book/html/htmledition/an-example-information-retrieval-problem-1.html (편집됨)

AI-2 (안인범)

안인범 18일 전

좋은 질문 감사합니다. 저도 해당 내용이 궁금했는데, 강의 자료에 소개된 논문이 사례가 될 수 있을 것 같아 답변 남깁니다. Topic 4: Text Representation I: Classic Methods 강의 자료 27~29p에서 설명된 논문이 다양한 TF-IDF 조합에 따른 성능을 비교한 내용인데요, TF variants 중 'b'라고 표기된 boolean TF가 binary representation이고 'n'이 기본적인 frequency representation 입니다. 실험 결과를 보면, 첫번째/두번째 데이터셋 모두에서 TF가 'b'인 조합이 'n'인 경우보다 성능이 좋은 경우를 볼 수 있습니다. 예를 들어 첫번째 데이터셋의 경우, IDF와 Normalization이 같은 조건을 보면 bΔ(t')n 조합이 96.50%, nΔ(t')n이 95.80%로 boolean TF의 성능이 더 좋게 나오고, 두번째 데이터셋에서도 bΔ(t')n이 95.30%, nΔ(t')n이 94.54%로 boolean TF가 더 좋은 경우가 있음을 알 수 있습니다.

논문의 결론 부분에서도 Binary approach가 Raw term frequency 보다 더 좋은 성능을 보였다는 언급이 있었습니다. 해당 논문 내용도 같이 참고하시면 좋을 것 같습니다. https://www.aclweb.org/anthology/P10-1141.pdf 감사합니다. (편집됨)

AI-2 (안인범)

Table 5: Statistics about the data sets used.

Data set	#Documents	#Terms	#Unique Terms	Average #Terms per Document
Movie Reviews	2,000	1,336,883	39,399	668
Multi-Domain Sentiment Dataset (MDSD)	8,000	1,741,085	455,943	217
BLOGS06	17,898	51,252,850	367,899	2,832

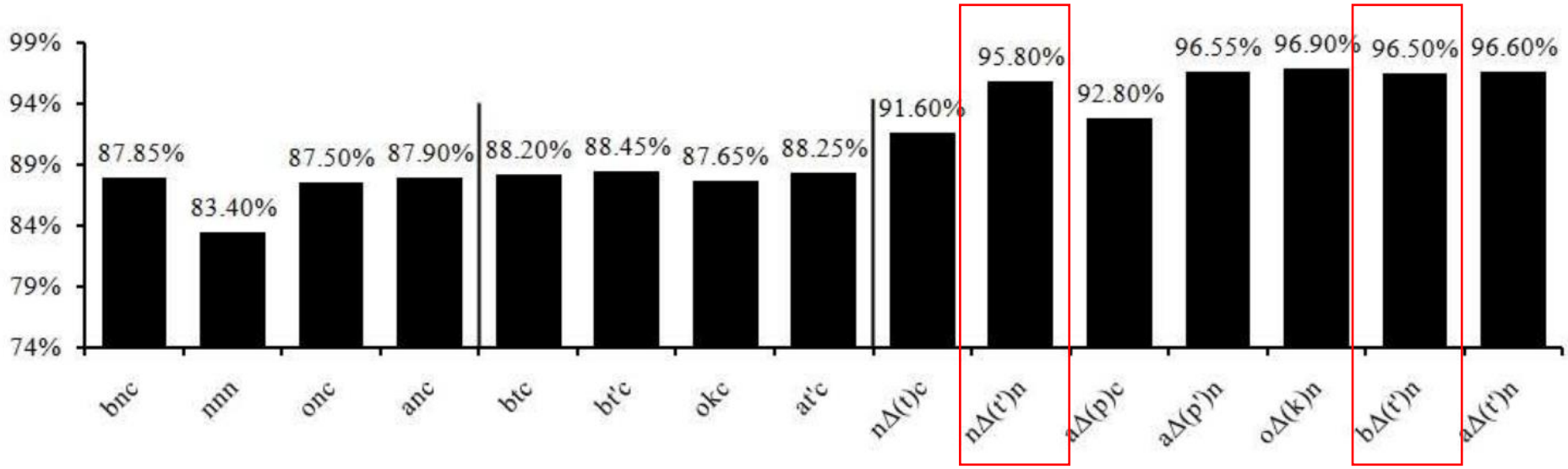


Figure 1: Reported accuracy on the Movie Review data set.

AI-3 (강필성)

International Journal of Engineering & Technology, 7 (3.33) (2018) 15-22

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

The Performance Comparison of the Classifiers According to Binary Bow, Count Bow and Tf-Idf Feature Vectors for Malware Detection

Young-Man Kwon[1], So-Hee Jun[2], Won-Mo Gal[3] and Myung-Jae Lim[4*]

[1]Department of Medical IT, College of Healthy Industry, Eulji University, Korea
[2]Department of Medical IT, College of Healthy Industry, Eulji University, Korea
[3]Department of Environmental Health and Safety, College of Healthy Industry, Eulji University, Korea
[4]Department of Medical IT, College of Healthy Industry, Eulji University, Korea
**Corresponding author E-mail: lk04@eulji.ac.kr*

Abstract

In this paper, we compared the performance of the classifiers according to feature vectors with Binary BOW, Count BOW and TF-IDF for malware detection. We used the feature of Opcode that extracted from PE file. For performance comparison, we measured the AUC score for the classifiers those are DT, KNN, MLP, MNB and SVM. As a result, we recommend neural network (MLP) and instance-based model (KNN) because they show the high AUC score and accuracy regardless of the unbalanced dataset and the feature vector. If you use classical classifiers, we recommend DT because it guarantees high AUC score and accuracy regardless of the same condition as the above. If you use SVM, you have to do Robust scaling to resolved outlier and unbalanced dataset. If you use MNB, you need to use N-gram technique to improve AUC score.

Keywords: *Malware Detection; Feature Selection; Machine Learning; BOW (Bag of words); TF-IDF*

AI-3 (강필성)

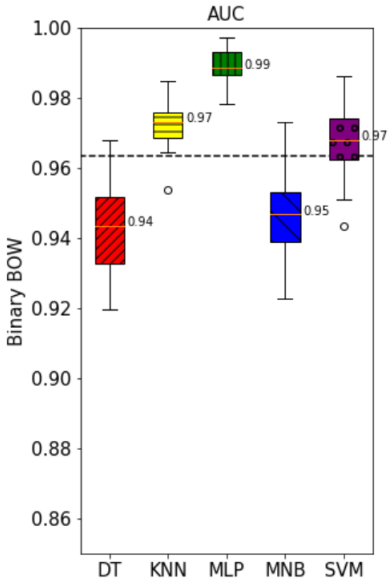
Table 1: The average and standard deviation of AUC score and accuracy in Binary BOW feature vectors

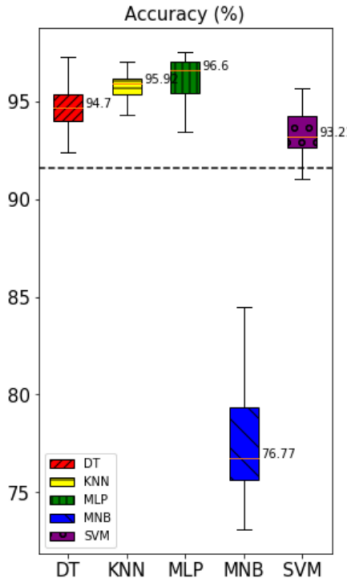
Feature Vector	Classifier	AUC		Accuracy (%)	
		AVG	STD	AVG	STD
Binary BOW	DT	0.942	0.011	94.755	1.048
	KNN	0.972	0.006	95.851	0.658
	MLP	0.989	0.005	96.232	1.045
	MNB	0.947	0.012	77.699	2.953
	SVM	0.968	0.010	93.306	0.988

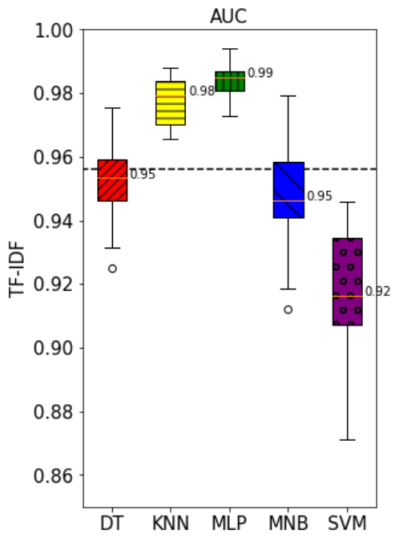
Table 2: The average and standard deviation of AUC score and accuracy in Count BOW and TF-IDF feature vector

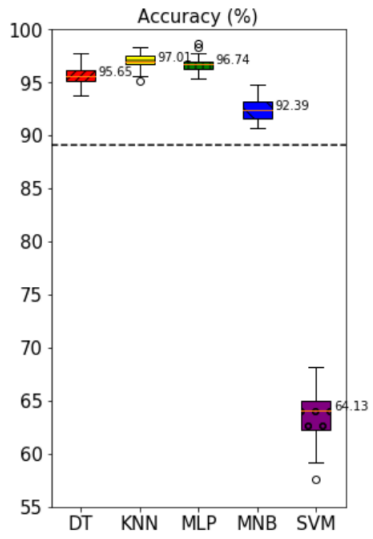
Feature Vector	Classifier	AUC		Accuracy (%)	
		AVG	STD	AVG	STD
Count BOW	DT	0.960	0.012	96.467	0.890
	KNN	0.976	0.010	96.476	0.976
	MLP	0.964	0.009	90.897	2.776
	MNB	0.911	0.021	75.833	7.632
	SVM	0.962	0.006	82.681	3.296
TF-IDF	DT	0.953	0.011	95.634	0.933
	KNN	0.978	0.007	96.984	0.727
	MLP	0.984	0.006	96.757	0.749
	MNB	0.948	0.015	92.527	0.959
	SVM	0.918	0.018	63.641	2.302

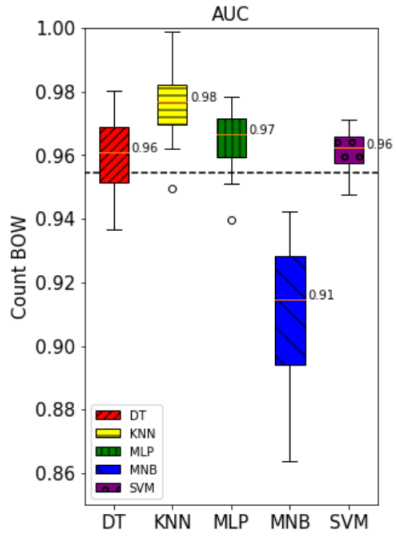
AI-3 (강필성)

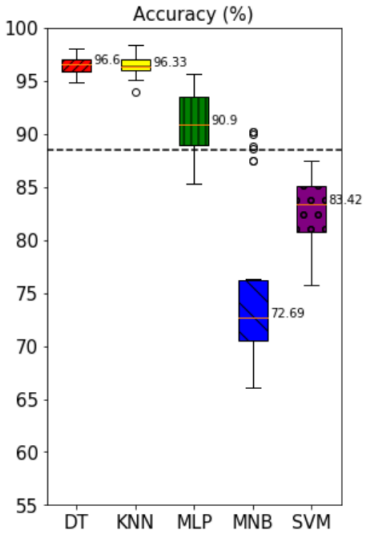

AUC
Binary BOW
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.94
0.97
0.99
0.95
0.97
DT
KNN
MLP
MNB
SVM


Accuracy (%)
95
90
85
80
75
94.7
95.9
96.6
93.2
76.77
DT
KNN
MLP
MNB
SVM


AUC
TF-IDF
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.95
0.98
0.99
0.95
0.92
DT
KNN
MLP
MNB
SVM


Accuracy (%)
100
95
90
85
80
75
70
65
60
55
95.65
97.01
96.74
92.39
64.13
DT
KNN
MLP
MNB
SVM


AUC
Count BOW
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.96
0.98
0.97
0.96
0.91
DT
KNN
MLP
MNB
SVM


Accuracy (%)
100
95
90
85
80
75
70
65
60
55
96.6
96.33
90.9
83.42
72.69
DT
KNN
MLP
MNB
SVM

Q2 (소규성)

3월 9일 화요일

소규성 오후 10:15

Lecture4: Text Representation1의 불용어(stop words)에 관해 궁금증이 생겨 질문 올립니다.

불용어가 사전적인 정의로는 '분석에 큰 의미가 없는 단어 토큰'이라고 하는데, 찾아본 바로는 이미 정의된 사전을 그대로 이용하거나 개발자가 직접 정의하는 경우가 많은 것 같습니다. 다만 동일한 사전을 이용하는 경우 수행하려는 task의 도메인에 따라 성능에 악영향을 미치지는 않을까 싶고 (예를 들어 수업자료 14pg의 "토하다" 등은 소설 요약 등에서는 중요할 수 있을 것 같습니다), 직접 정의하는 경우 정성적인 작업량이 굉장히 클 뿐더러 추후 구축하는 모델의 성능에 어떤 영향을 미칠 지 예상이 어려울 것 같습니다.

이와 관련해,

1. 불용어를 정의하는 데 있어서 효과적으로 정량적인 기준을 이용하는 방법은 어떤 방법이 있는지,
2. 실제 NLP task에서 정의된 불용어 사전이 성능에 미치는 영향을 측정할 수 있는지 혹은 컨트롤 할 수 있는지

궁금합니다. 미리 답변 감사합니다! (편집됨)

4개의 답글 19일 전 마지막 답글

A2-1 (차형주)

차형주 19일 전

안녕하세요, 저도 해당 부분이 궁금하여 찾아보았는데 아쉽게도 불용어에 대한 정량적인 효과를 측정하는 방법론은 없는 것 같았습니다.

불용어를 다루는 방식이 '사전' 처럼 일종의 데이터 베이스로 다루고 있는 상황이기에 불용어 성능 조절 또한 데이터베이스를 수정하는 방식으로 진행되고 있는 듯 합니다.

NLP에 대한 깊은 이해가 없어서 최대한 쉬운 튜토리얼을 찾아보았는데, 참고해보셔도 좋을 것 같습니다.

추후 더 좋은 방안을 찾게되면 또 댓글달겠습니다!

https://wikidocs.net/77135
https://omicro03.medium.com/%EC%9E%90%EC%97%B0%EC%96%B4%EC%B2%98%EB%A6%AC-nlp-5%EC%9D%BC%EC%B0%A8-%EB%B6%88%EC%9A%A9%EC%96%B4-277694095a6f

> **wikidocs.net**
> **위키독스**
> 온라인 책을 제작 공유하는 플랫폼 서비스

> **Medium**
> **자연어처리(NLP) 5일차 (불용어)**
> 19.06.07
> **Reading time**
> 5 min read
> 2019년 6월 7일

A2-2 (김탁영)

김탁영 19일 전

먼저 질문주셔서 감사합니다. 저도 잘 알지는 못하지만 1번에 대한 답변을 찾다가 아래 논문을 발견하였습니다. 불용어를 정의하는 데에도 몇 가지 방법이 있는 듯 하여 간략하게 공유드립니다.

- 논문: A Systematic Review on Stopword Removal Algorithms (2018)
- 링크: http://www.ijfrcsce.org/download/browse/Volume_4/April_18_Volume_4_Issue_4/1524218332_20-04-2018.pdf

[1] Classical Method

우리가 모두 아는 방법이고, 미리 정의된 리스트로부터 불용어를 걸러냅니다.

[2] Zipf's Law (Z-Methods)

미리 정의된 리스트에 더하여 ① 가장 자주 등장하는 단어 ② 딱 한 번 등장하는 단어 ③ 낮은 IDF 값을 갖는 단어를 삭제하는 방법입니다.

[3] Mutual Information Method

Supervised 방법론이고, 단어와 문서 클래스 간의 mutual information을 계산합니다. 예를 들어 '별로' 라는 단어는 해당 문서가 '부정적'이라는 것에 영향을 많이 미치는 단어이기 때문에 information이 높다고 간주하고 삭제하지 않습니다.

[4] Term based Random Sampling

불용어를 web document에서 수동적으로 찾아서 제거하는 방법입니다. 먼저 랜덤하게 단어를 선택하고 이 단어를 포함하는 문서를 모두 찾습니다. 이 문서 집합은 하나의 chunk라고 하며, 이 chunk에서 가장 정보량이 적은 단어를 추출하여 불용어라고 규정합니다. 정보량을 측정할 때에는 chunk 내의 단어 분포와 전체 문서 내의 단어 분포 사이의 KL Divergence를 구합니다. (제가 이해한 바로는 이런데 혹시 틀린 부분이 있으면 지적부탁드립니다..!)

리뷰 페이퍼에서 언급한 내용은 위 4가지이며, 4장짜리 논문이기 때문에 한 번 훑어보셔도 좋을 것 같습니다. 감사합니다.

👍 1

A2-3 (김수빈)

김수빈 19일 전

안녕하세요. 좋은 질문 감사합니다.
평소에 당연하게 받아들이며 그냥 넘겼던 내용인데, 질문자님의 질문 덕분에 한번 더 고민해보고 리서치 해볼 수 있었습니다.

저는 질문자님이 의문점을 제기해주신 "사전 정의된 불용어와 성능의 관계"에 초점을 두고 더 조사를 해보았는데요,
불용어로 인해 오히려 성능이 저하된 한 사례(Task : Twitter sentiment classification)를 바탕으로, 여러 불용어 방법론들에 대하여 다음 세 가지 관점으로 어떻게 실험을 진행했는지 잘 정리가 된 논문을 발견하여 공유드립니다.

Assessment of the impact of removing
stopwords

1. Data Sparsity
2. Size of classifier's feature space
3. Performance(accuracy / F-measure)

http://www.lrec-conf.org/proceedings/lrec2014/pdf/292_Paper.pdf
불용어 제거 방법론에 따라, 또 제거 유무에 따른 실험 결과가 잘 정리되어 있어 가볍게 읽어보기 좋은 것 같습니다.

물론 이건 한 예시이고, 다른 task에 대해서는 설명이 되지 않을 수 있지만, 위의 세 가지 평가지표처럼(특히 1번 2번) 단순한 '불용어 제거'에서 더 나아가 embedding된 data 자체의 특성에 대해서 어떤 trade-off가 있을 수도 있는지까지 함께 고려할 수 있다는 것이 생각해볼 만한 점인 것 같습니다.

+) 덧붙여, 막연한 개인적인 생각으로는 데이터만 충분하다면 아주 기초적인 불용어(관사 등) 외에는 굳이 제거할 필요가 있을까 하는 생각이 듭니다. 제거해야 한다면, domain에 대해 automatic하게 불용어를 제거하는 방법론은 없을까, 불용어를 모델이 학습하여 일정 확률 이하는 drop 시켜도 비슷한 효과를 낼 수 있지 않을까 하는 생각으로 더 조사해보았으나 아직 괜찮은 논문을 발견하진 못했습니다. 이에 대해 추가적인 의견이나 피드백이 있다면 언제든 공유해주시면 감사하겠습니다.

👍 1

A2-3 (김탁영 + 김수빈 + 강필성)

- 현재 김탁영 학생의 답변과 관련된 논문의 링크가 작동하지 않음
  - ✓ 김수빈 학생이 참고한 논문이 2014년 논문이고 김탁영 학생이 참고한 논문이 2018년 논문인데 두 논문에서 사용한 불용어 제거 방법론이 동일함
  - ✓ [1] Classical Method
    - 우리가 모두 아는 방법이고, 미리 정의된 리스트로부터 불용어를 걸러냅니다.
  - ✓ [2] Zipf's Law (Z-Methods)
    - 미리 정의된 리스트에 더하여 ① 가장 자주 등장하는 단어 ② 딱 한 번 등장하는 단어 ③ 낮은 IDF 값을 갖는 단어를 삭제하는 방법입니다.
  - ✓ [3] Mutual Information Method
    - Supervised 방법론이고, 단어와 문서 클래스 간의 mutual information을 계산합니다. 예를 들어 '별로' 라는 단어는 해당 문서가 '부정적'이라는 것에 영향을 많이 미치는 단어이기 때문에 information이 높다고 간주하고 삭제하지 않습니다.
  - ✓ [4] Term based Random Sampling
    - KL Divergence 이용

A2-3 (김수빈) + 강필성

- Reduction rate & Data sparsity

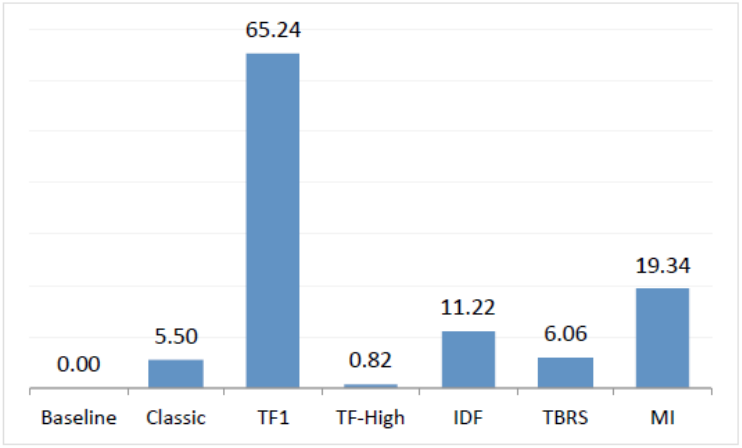


Figure 5: Reduction rate on the feature space of the various stoplists

$$S_d = \frac{\sum_j^n N_j}{n \times m} \quad (3)$$

Where N_j is the the number of *zero* elements in column j (i.e., tweet i). Here $S_d \in [0, 1]$, where high S_d values refer to high sparsity degree and vice versa.

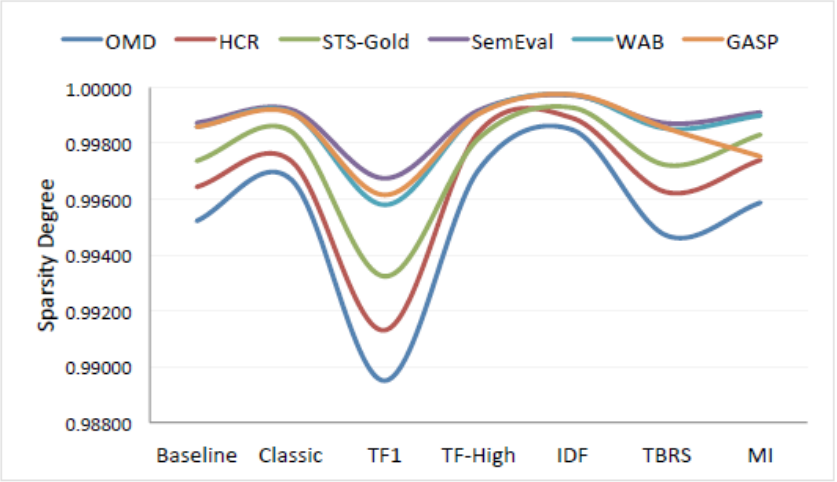
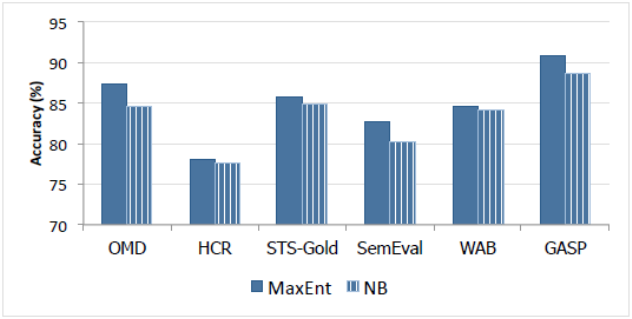


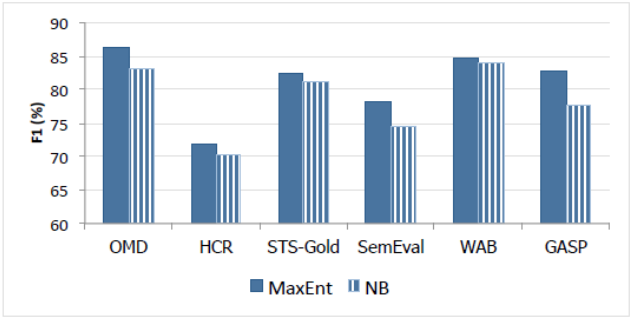
Figure 7: Stoplist impact on the sparsity degree of all datasets

A2-3 (김수빈) + 강필성

- Average accuracy
  - ✓IDF가 낮은 단어를 제외하는 것은 성능을 유의미하게 저하시킴
  - ✓이 외의 방식들은 제거하더라도 성능 저하가 크기 않거나 약간 상승함

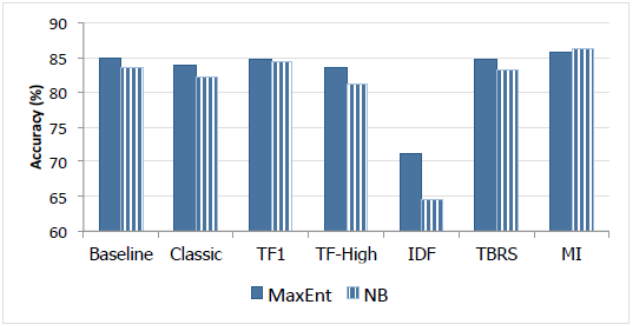


(a) Average Accuracy

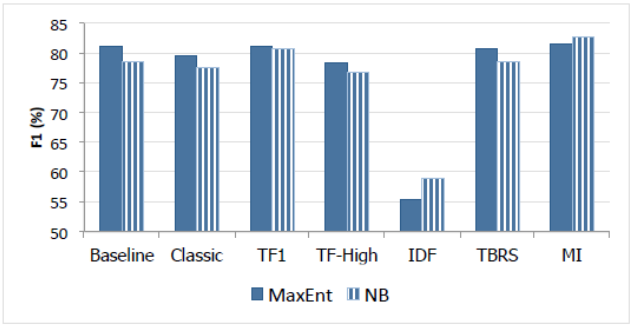


(b) Average F-measure

Figure 3: The baseline classification performance in Accuracy and F-measure of MaxEnt and NB classifiers across all datasets



(a) Average Accuracy



(b) Average F-measure

Figure 4: Average Accuracy and F-measure of MaxEnt and NB classifiers using different stoplists

A2-4 (임새린)

임새린 19일 전

위에서 설명을 잘해주셔서 약간 첨언을 하자면 stopwords의 사용은 task별로 그리고 dataset별로 나눌 수 있을 것 같습니다.
dataset별로는 각 dataset에 맞는 stopwords를 정의해야 하며 방법론들은 위에서 잘 설명해 주셨으니 넘어가겠습니다.
Task별로 크게 나누어 보자면,

1. Remove Stopwords

- Text Classification(Spam Filtering, Language Classification 등)
- Caption Generation
- Auto-tag Generation

2. Avoid Stopwords Removal

- Machine Translation
- Language Modeling
- Text Summarization
- Question-Answering problems

위와 같이 나눌 수 있습니다.

Stopwords를 사용하면 당연히 정보에 손실이 생기고 문맥 파악에 있어서 불이익이 있으므로 문맥 파악이 중요한 task에서는 잘 하지 않는 것으로 보입니다. (편집됨)

A2-4 (임새린 + 강필성)

- History of NLP https://en.wikipedia.org/wiki/Natural_language_processing

Natural language processing

From Wikipedia, the free encyclopedia

Natural language processing (**NLP**) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. The result is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

Challenges in natural language processing frequently involve speech recognition, natural language understanding, and natural-language generation.

An automated online assistant providing customer service on a web page, an example of an application where natural language processing is a major component.[1]

Contents [hide]

History [edit]

Further information: History of natural language processing

Natural language processing has its roots in the 1950s. Already in 1950, Alan Turing published an article titled "Computing Machinery and Intelligence" which proposed what is now called the Turing test as a criterion of intelligence, a task that involves the automated interpretation and generation of natural language, but at the time not articulated as a problem separate from artificial intelligence.

A2-4 (임새린 + 강필성)

- 응답하라 1990

세계의 슈퍼컴퓨터 [편집]

> 현재 이 문단은 주로 **대한민국에 한정된 내용**만을 다루고 있습니다.
> 다른 국가·지역에 대한 내용을 **보충**하여 문서의 균형을 맞추어 주세요. 내용에 대한 의견이 있으시면 토론 문서에서 나누어 주세요. (2015년 6월)

대한민국의 슈퍼컴퓨터 [편집]

대한민국 최초의 슈퍼컴퓨터는 1988년 11월 연구전산망(KREOnet)의 중앙전산기로 도입된 2GFlops 성능의 Cray-2S이며, 1993년 11월에는 슈퍼컴퓨터 2호기로 16GFlops 성능의 Cray-C90가 도입되었다.

기준으로 대한민국은 세계 랭킹 18 위의 슈퍼컴퓨터 1대를 보유하고 있으며, 한국과학기술정보연구원(KISTI) 산하에 슈퍼컴퓨팅센터를 두고 있고 그 밖에 한국슈퍼컴퓨팅센터협의회(KSCA)가 구성되어 모두 15개 기관이 가입하고 있다.

국내 슈퍼컴퓨터 목록 [편집]

- 한국과학기술정보연구원(KISTI)[3]
  - 1호 : 1988년 KISTI는 2 기가플롭스 속도인 국내 1호 슈퍼컴퓨터 Cray-2S를 미국에서 도입했다. 자동차 설계, 기후 예측 등의 응용 분야와 계산과학 분야에 주로 사용
  - 2호 : 1993년 2호기 슈퍼컴퓨터 Cray C90를 도입했다. 자동차 설계, 기후 예측 등의 응용 분야와 계산과학 분야에 주로 사용
  - 3호 : 3호기 슈퍼컴퓨터는 2001년부터 2003년까지 NEC SX-5,6과 IBM p690을 순차적으로 도입하였다. 정식 명칭 "노벨". IBM p690 와 NEC SX-5 벡터형 슈퍼컴퓨터 2가지로 구성. 이론속도 4.3 테라플롭스, 실측속도 2.8 테라플롭스. 2003년 12월 도입. 2004년 11월 기준으로 세계 156위. 단백질 접힘 구조 분석, 유체역학 등 첨단 응용과학에 사용[4][5][6]
  - 4호 : 2010년 5월 31일 독일 함부르크에서 개최된 국제슈퍼컴퓨팅컨퍼런스(ISC'10)에서 제35차 슈퍼컴 리스트가 발표되었다. 대한민국 최고 속도의 슈퍼컴퓨터는 "KISTI 슈퍼컴퓨터 4호기"로서, 당시 세계 15위를 기록했다. 대한민국이 보유한 슈퍼컴으로는 유일하게 500위 내에 들었다. 썬마이크로시스템즈 제품이며 300 테라플롭스의 속도를 낸다.

A2-4 (임새린 + 강필성)

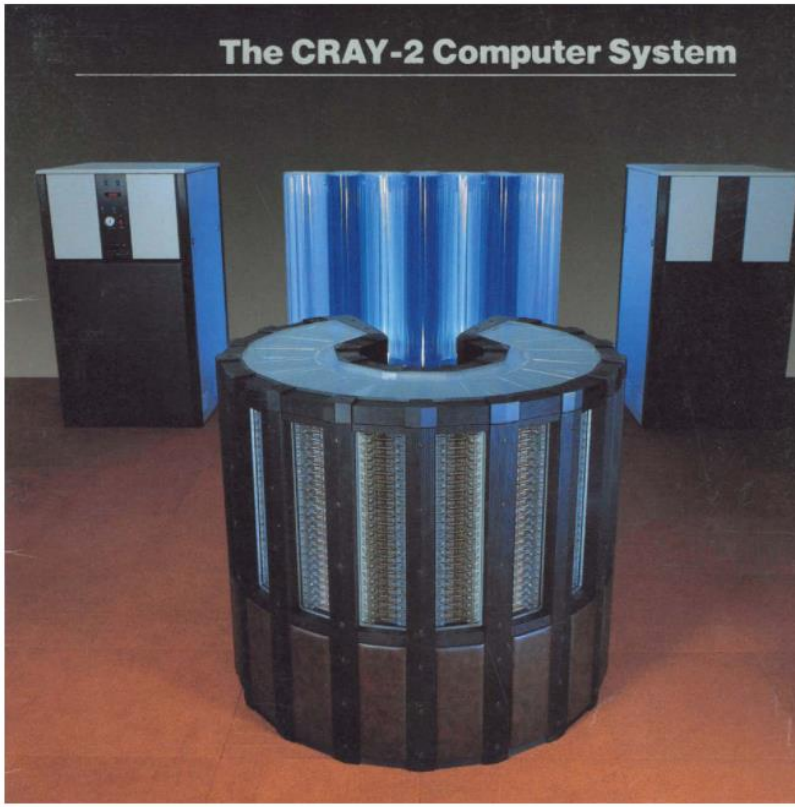
- 응답하라 1990

A2-4 (임새린 + 강필성)

Cray 2 v iPhone XS: Fight!

Diego Doval Dec 17, 2018 · 2 min read

How does the iPhone XS compare to the most powerful and expensive supercomputer from 30 years ago?



	Cray 2 (1985)	iPhone XS (2018)
Price (2017 USD)	> $30,000,000	~$900
Main Processors	4	1 A12X
Memory (RAM)	256 Megaword	4 Gigabytes

* megaword -> a 'word' varies but the Cray-2 used 64-bit words with 8 bit parity checks

	Cray 2 (1985)	iPhone XS (2018)
Storage (max)	~32 GB*	512 GB

* Max storage required 32 disk drives of 1.2 GB each

	Cray 2 (1985)	iPhone XS (2018)
Peak Power Consumption	195,000 Watts	< 1 Watt*

*It's very hard to compare peak power given iPhone has so many other functions, so let's just go with < 1 W

	Cray 2 (1985)	iPhone XS (2018)
Peak Performance	1.9 GFLOP *	~1 GFLOP*

*GFLOP = Billions of Floating Point operations per second

https://medium.com/@diego./cray-2-v-iphone-xs-fight-6f05b494efe1

Single-Core Performance

	iPhone 8 Plus	iPhone11,6
Single-Core Score	**4307**	**4813**
AES	3081 2.32 GB/sec	3760 2.83 GB/sec
LZMA	3798 5.93 MB/sec	4331 6.77 MB/sec
JPEG	4144 33.3 Mpixels/sec	4536 36.5 Mpixels/sec
Canny	4353 60.4 Mpixels/sec	4915 68.2 Mpixels/sec
Lua	4782 4.92 MB/sec	5219 5.36 MB/sec
Dijkstra	4481 3.03 MTE/sec	6067 4.11 MTE/sec
SQLite	4275 118.5 Krows/sec	4773 132.3 Krows/sec
HTML5 Parse	4636 21.1 MB/sec	5045 22.9 MB/sec
HTML5 DOM	5860 5.31 MElements/sec	6723 6.09 MElements/sec
Histogram Equalization	4058 126.8 Mpixels/sec	4207 131.5 Mpixels/sec
PDF Rendering	4350 115.6 Mpixels/sec	4813 127.9 Mpixels/sec
LLVM	9058 622.8 functions/sec	10284 707.1 functions/sec
Camera	5102 14.1 images/sec	5413 15.0 images/sec
SGEMM	2575 54.4 Gflops	2667 56.4 Gflops
SFFT	3329 8.30 Gflops	3552 8.86 Gflops
N-Body Physics	3484 2.60 Mpairs/sec	4801 3.59 Mpairs/sec
Ray Tracing	4364 637.2 Kpixels/sec	4914 717.6 Kpixels/sec
Rigid Body Physics	4173 12218.6 FPS	4682 13708.3 FPS
HDR	5050 18.3 Mpixels/sec	5425 19.7 Mpixels/sec
Gaussian Blur	4682 82.0 Mpixels/sec	5112 89.6 Mpixels/sec
Speech Recognition	4112 35.2 Words/sec	5498 47.0 Words/sec
Face Detection	4970 1.45 Msubwindows/sec	5914 1.73 Msubwindows/sec
Memory Copy	5123 14.2 GB/sec	5351 14.8 GB/sec
Memory Latency	3899 111.0 ns	3999 108.3 ns
Memory Bandwidth	3274 17.5 GB/sec	3359 17.9 GB/sec

Multi-Core Performance

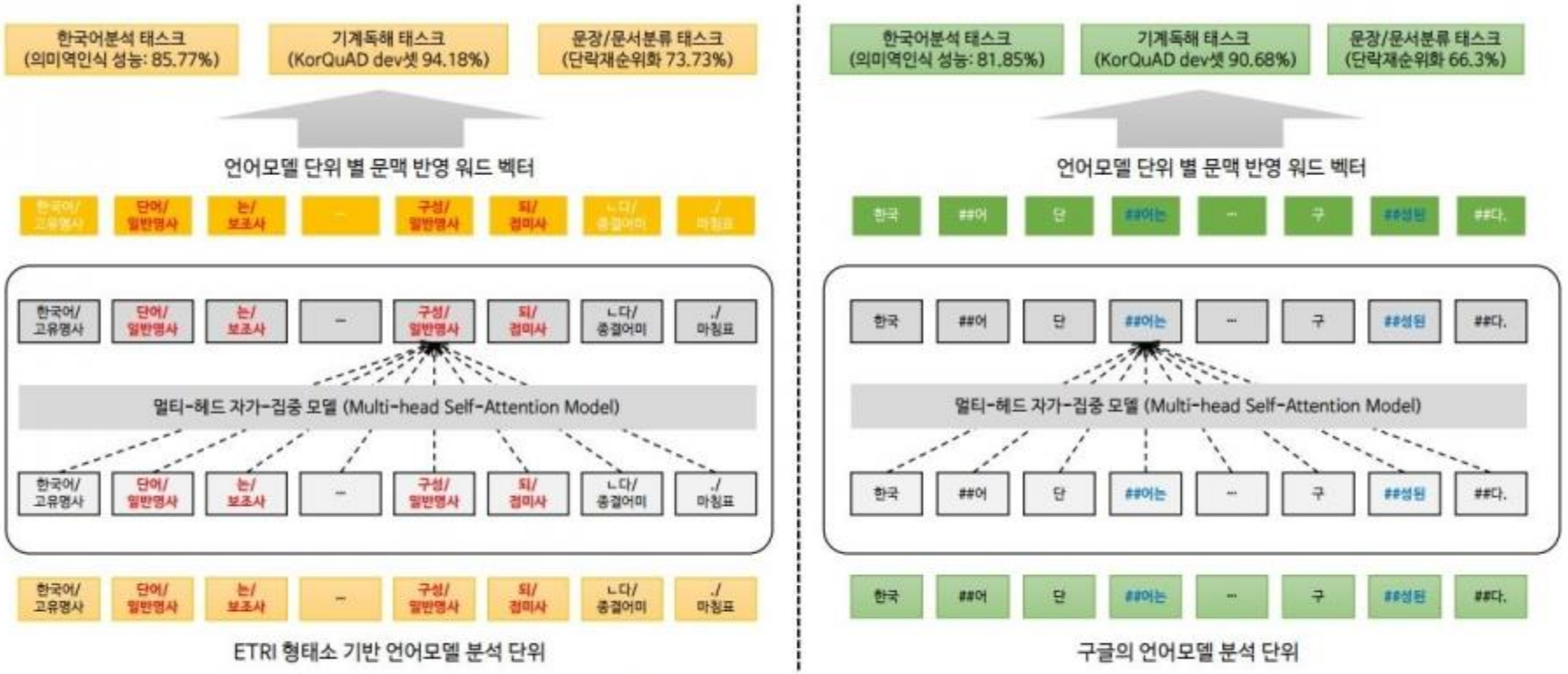
	iPhone 8 Plus	iPhone11,6
Multi-Core Score	**10800**	**10266**
AES	7876 5.93 GB/sec	8629 6.50 GB/sec
LZMA	11311 17.7 MB/sec	10994 17.2 MB/sec
JPEG	13761 110.7 Mpixels/sec	11753 94.6 Mpixels/sec
Canny	13829 191.8 Mpixels/sec	12474 173.0 Mpixels/sec
Lua	13352 13.7 MB/sec	12359 12.7 MB/sec
Dijkstra	13091 8.86 MTE/sec	14131 9.56 MTE/sec
SQLite	12210 338.5 Krows/sec	12126 336.2 Krows/sec
HTML5 Parse	14023 63.7 MB/sec	12896 58.5 MB/sec
HTML5 DOM	17579 15.9 MElements/sec	15147 13.7 MElements/sec
Histogram Equalization	12099 378.1 Mpixels/sec	10888 340.2 Mpixels/sec
PDF Rendering	11765 312.6 Mpixels/sec	10806 287.1 Mpixels/sec
LLVM	25754 1.77 Kfunctions/sec	23359 1.61 Kfunctions/sec
Camera	15811 43.8 images/sec	13275 36.8 images/sec
SGEMM	5886 124.4 Gflops	5475 115.7 Gflops
SFFT	9579 23.9 Gflops	8801 21.9 Gflops
N-Body Physics	9729 7.27 Mpairs/sec	10782 8.05 Mpairs/sec
Ray Tracing	10805 1.58 Mpixels/sec	8782 1.28 Mpixels/sec
Rigid Body Physics	13459 39399.6 FPS	11580 33899.4 FPS
HDR	15148 54.9 Mpixels/sec	13570 49.2 Mpixels/sec
Gaussian Blur	10682 187.1 Mpixels/sec	10217 179.0 Mpixels/sec
Speech Recognition	9953 85.2 Words/sec	11984 102.5 Words/sec
Face Detection	14382 4.20 Msubwindows/sec	14511 4.24 Msubwindows/sec
Memory Copy	5872 16.3 GB/sec	6464 17.9 GB/sec
Memory Latency	3560 121.6 ns	3822 113.3 ns
Memory Bandwidth	3268 17.5 GB/sec	3472 18.5 GB/sec

A2-4 (임새린 + 강필성)

- 과거 NLP Preprocessing의 목적: 다이어트
  - ✓ Stemming/Lemmatization: 단어의 Variation들을 모두 처리하는 것이 불가능하니까 줄이자!
  - ✓ Stopwords: 문법적인 기능만 존재하거나 큰 의미 없는 단어들 제외하자!
  - ✓ Feature Selection: Downstream Task를 수행하는데 불필요한 토큰들 쓰지 말자!

A2-4 (임새린 + 강필성)

- 최근 NLP 트렌드
  - ✓기-승-전-언어모델



예문: 한국어 단어는 형태소로 구성된다.

〈ETRI 형태소 기반 언어모델과 구글 언어모델 비교〉

http://www.businesskorea.co.kr/news/articleView.html?idxno=32740