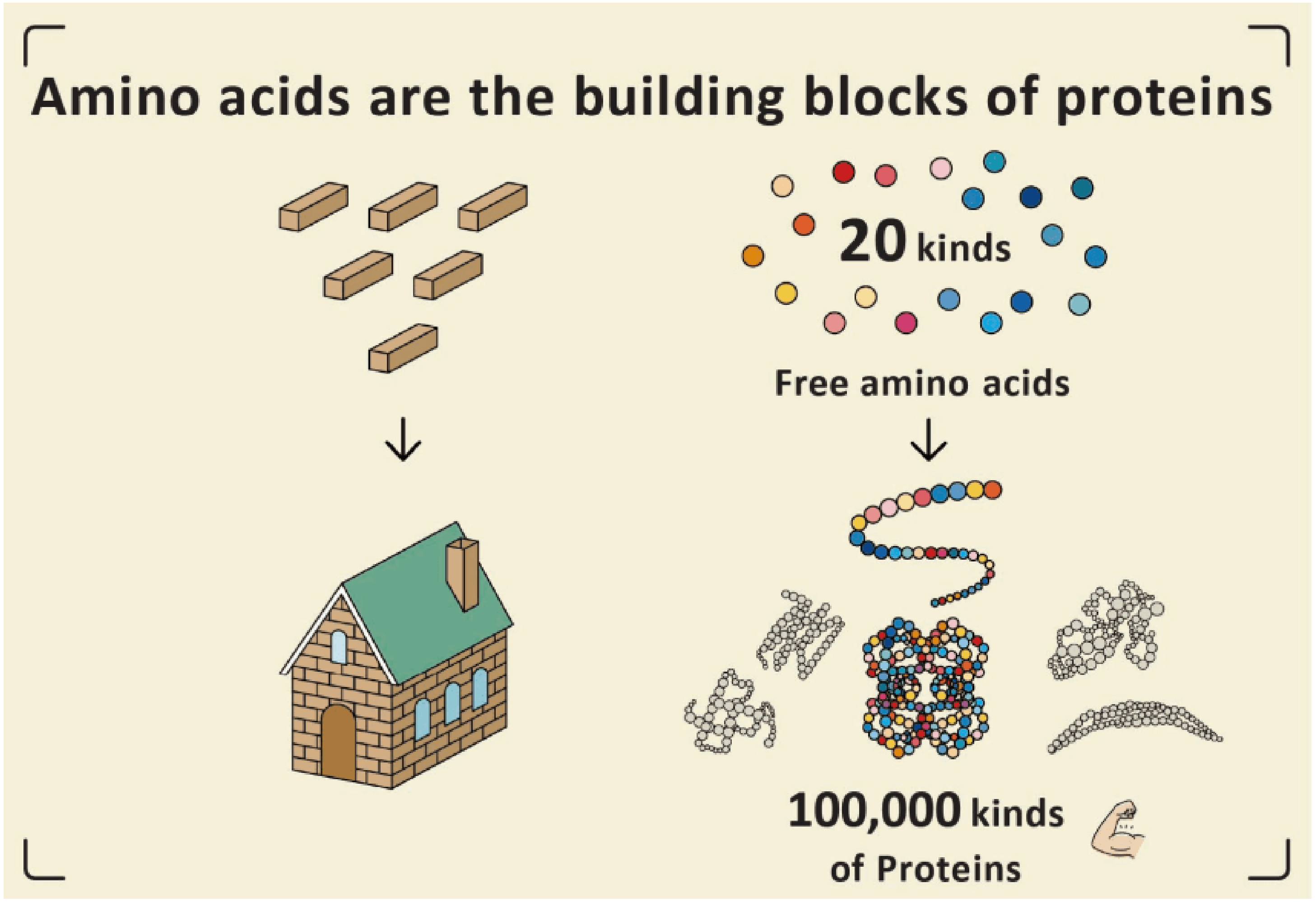
단백질 및 화합물을 위한 Word2Vec 알고리즘 기반의 특질 추출 기법

서울대학교 의생명지식공학연구실
이문환
munhwanlee@snu.ac.kr



https://www.ajinomoto.com/aboutus/amino-acids/what-are-amino-acids

Previous work: K-MER

- **K-mer**
  - 아미노산 서열을 K의 길이를 가진 부분 서열로 추출함
  - 해당 부분 서열의 분포를 통하여 해당 단백질의 특징을 분석함

	서열 조합 수	파일 크기
K=2	$20^2 = 400$	244KB
K=3	$20^3 = 8,000$	4875KB

ESSKVAITYADSGVSVDNGNNLVQTIKEMVRSTRRPGAD
SDIGGFGGLFDLAQAGFRQNEDTLLVGATDGVGTKLIIAQ
ETGIHNTVGIDLVAMNVNDLVVQ …

K=2

ES SS SK KV VA AI IT TY YA AD DS SG GV VS SV VD
DN NG GN NN NL LV VQ TI KE MV RS TR RP GA DS DI
GG FG GL FD LA QA GF RQ NE DT LL VG AT DG VG TK
LI IA QE TG IH NT VG ID LV AM NV ND LV VQ …

	AA	AR																			VV
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**…**	**390**	**391**	**392**	**393**	**394**	**395**	**396**	**397**	**398**	**399**
Seq1	7	7	0	6	0	3	3	7	3	1	…	3	0	0	2	2	1	2	0	1	3
Seq2	2	2	1	2	0	1	5	5	2	0	…	4	6	1	1	2	5	1	0	1	1
Seq3	6	5	0	2	1	2	3	4	2	6	…	2	1	1	2	2	1	1	1	2	5
Seq4	6	4	0	2	0	2	2	7	1	3	…	5	0	0	0	1	0	5	0	0	0
Seq5	8	3	4	1	1	1	3	6	0	4	…	3	0	1	2	0	3	4	0	0	2
Seq6	6	4	2	1	0	0	6	3	1	1	…	4	1	2	1	1	5	1	0	0	3
Seq7	9	5	1	1	0	2	3	7	2	1	…	8	4	1	4	2	2	2	0	1	1
…																					

Previous work: K-MER

- **K-mer**
  - 아미노산 서열을 K의 길이를 가진 부분 서열로 추출함
  - 해당 부분 서열의 분포를 통하여 해당 단백질의 특징을 분석함

ESSKVAITYADSGVSVDNGNNLVQTIKEMVRSTRRPGAD
SDIGGFGGLFDLAQAGFRQNEDTLLVGATDGVGTKLIIAQ
ETGIHNTVGIDLVAMNVNDLVVQ …

K=3

ESS SSK SKV KVA VAI AIT ITY TYA YAD ADS DSG SGV
GVS VDN GNN LVQ TIK EMV RST RRP GAD SDI GGF
GGL FDL AQA GFR QNE DTL LVG ATD GVG TKL IIA
QET GIH NTV GID LVA MNV NDL VVQ …

	서열 조합 수	파일 크기
K=2	$20^2 = 400$	244KB
K=3	$20^3 = 8,000$	4875KB

	AAA	AAR																	VVY	VVV
	0	1	2	3	4	5	6	7	8	9	...	7990	7991	7992	7993	7994	7995	7996	7997	7998	7999
Seq1	1	2	0	0	0	0	0	0	0	0	...	0	0	0	1	0	1	0	0	0	0
Seq2	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
Seq3	1	1	0	0	0	0	0	0	0	3	...	0	0	0	1	1	1	0	0	1	0
Seq4	2	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Seq5	1	1	1	0	0	0	0	0	0	1	...	2	0	0	0	0	0	0	0	0	0
Seq6	0	1	0	0	0	0	1	1	0	0	...	0	1	0	0	0	1	0	0	0	1
Seq7	1	0	0	0	0	1	0	2	1	0	...	0	0	0	0	0	0	1	0	0	0

…

Previous work: K-MER

- **K-mer**
  - 아미노산 서열을 K의 길이를 가진 부분 서열로 추출함
  - 해당 부분 서열의 분포를 통하여 해당 단백질의 특징을 분석함

	서열 조합 수	파일 크기
K=2	$20^2 = 400$	244KB
K=3	$20^3 = 8{,}000$	4875KB
K=4	$20^4 = 160{,}000$	97.5MB

ESSKVAITYADSGVSVDNGNNLVQTIKEMVRSTRRPGAD
SDIGGFGGLFDLAQAGFRQNEDTLLVGATDGVGTKLIIAQ
ETGIHNTVGIDLVAMNVNDLVVQ …

K=4

ESSK SSKV SKVA KVAI VAIT AITY ITYA TYAD YADS
GVSV DNGN NLVQ TIKE MVRS TRRP GADS DIGG
FGGL FDLA QAGF RQNE DTLL VGAT DGVG TKLI IAQE
TGIH NTVG IDLV AMNV NDLV VQ …

Sparsity Problem

	0	1	2	3	4	5	6	7	8	9	...	159990	159991	159992	159993	159994	159995	159996	159997	159998	159999
Seq1	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Seq2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Seq3	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
Seq4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Seq5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Seq6	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
Seq7	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
…																					

Previous work: K-MER

- **K-mer**
  - 아미노산 서열을 K의 길이를 가진 부분 서열로 추출함
  - 해당 부분 서열의 분포를 통하여 해당 단백질의 특징을 분석함

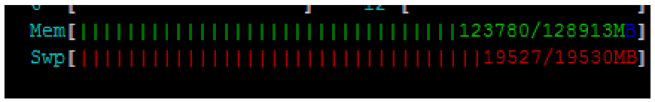
	서열 조합 수	파일 크기
K=2	20^2 = 400	244KB
K=3	20^3 = 8,000	4875KB
K=4	20^4 = 160,000	97.5MB
K=5	20^4 = 3,200,000	1.95GB

ESSKVAITYADSGVSVDNGNNLVQTIKEMVRSTRRPGAD SDIGGFGGLFDLAQAGFRQNEDTLLVGATDGVGTKLIIAQ ETGIHNTVGIDLVAMNVNDLVVQ …

K=5

ESSKV SSKVA SKVAI KVAIT VAITY AITYA ITYAD DSGVS VDNGN NLVQT IKEMV RSTRR PGADS DIGGF GGLFD LAQAG FRQNE DTLLV GATDG VGTKL IIAQE TGIHN TVGID LVAMN VNDLV VQ …

Computational Problem



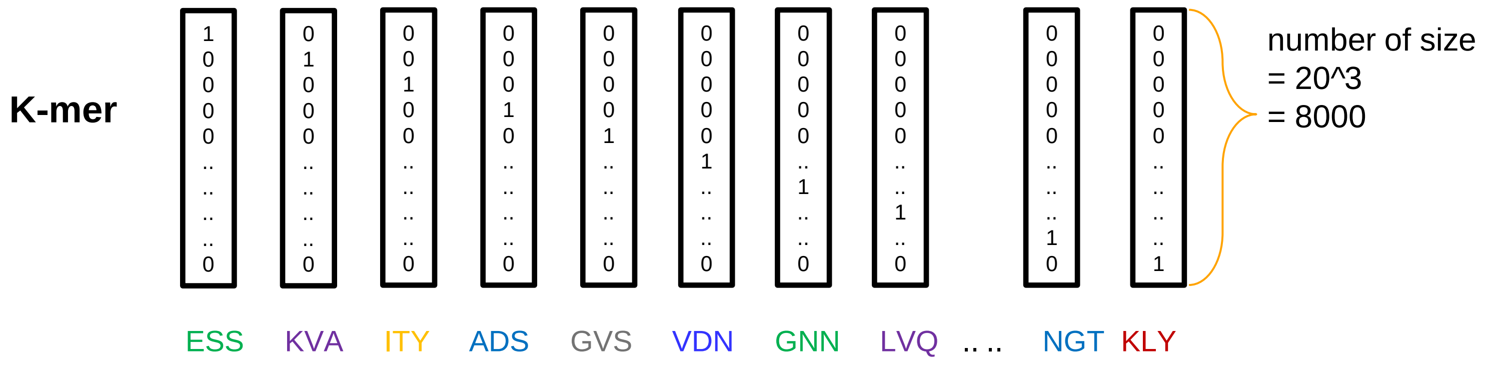
```
  0  [                    ]     12  [                    ]
  Mem[||||||||||||||||||||||||||||||||123780/128913MB]
  Swp[||||||||||||||||||||||||||||||||||19527/19530MB]
```

Previous work: K-MER

- Protein1

ESSKVAITYADSGVSVDNGNNLVQTIKEMVRSTRRPGADSDIGGFGGLFDLAQAGFRQNEDTLLVGATDGVGTKLIIAQETGIHNTVGIDLVAMNVND
LVVQGAEPLFFLDYFATGALDIQVASDFVSGVANGCIQSGCALVGGETSEMPGMYPPGHYDTNGTAVGAVLRQDILPKINEMAAGDVLLGLASSGVH
SNGFSLVRKIIQHVALPWDAPCPWDESKTLGEGILEPTKIYVKQLLPSIRQRLLLGLAHITGGGLVENIPRAIPDHLQARVDMSTWEVPRVFKWFGQAG
NVPHDDILRTFNMGVGMVLIVKRENVKAVCDSLTEEGEIIWELGSLQERPKDAPGCVIENGTKLY

- Protein (sub) sequence= k-words (k=3)

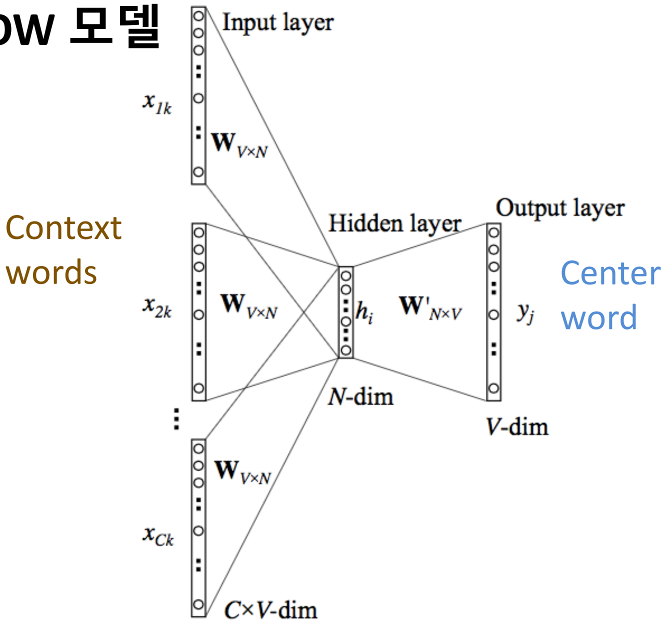


Word2Vec – Skip Gram 모델

- **Word2vec 모델 개요**
  - 모델의 가설: '비슷한 단어는 비슷한 맥락(context) 단어를 갖는다.'
  - 모델별 학습 차이

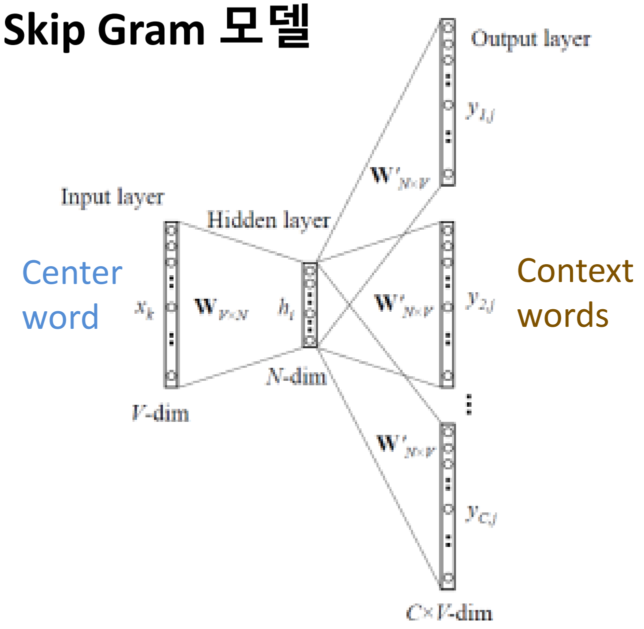
	입력 데이터	출력 데이터 (학습 목표)
CBOW	맥락(context) 단어들	가운데(center) 단어
Skip Gram	가운데(center) 단어	맥락(context) 단어들

CBOW 모델



https://miro.medium.com/max/1192/0*CCsrTAjN80MgswXG

Skip Gram 모델



https://cdn.analyticsvidhya.com/wp-content/uploads/2017/06/05000515/Capture2-276x300.png

ProtVec

- **Word2vec 모델 학습 데이터**
  - 말뭉치(Corpus): Wikipedia
  - 문장 추출
    - "Word2vec is a technique for natural language processing (NLP) published in 2013. "
    - "The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text."
    - "Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence."
    - "As the name implies, word2vec represents each distinct word with a particular list of numbers called a vector."
  - 단어 추출

<말뭉치(Corpus)>

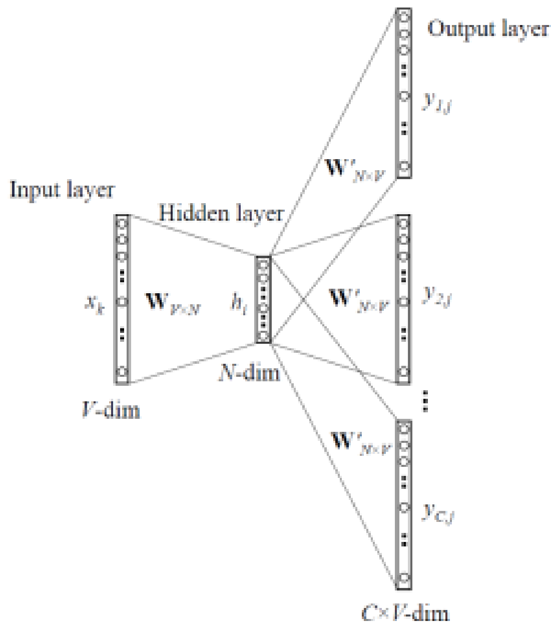
<문장>

Word2vec is a technique for …
The word2vec algorithm uses …
Once trained, such a model can …
As the name implies, word2vec …

…
…

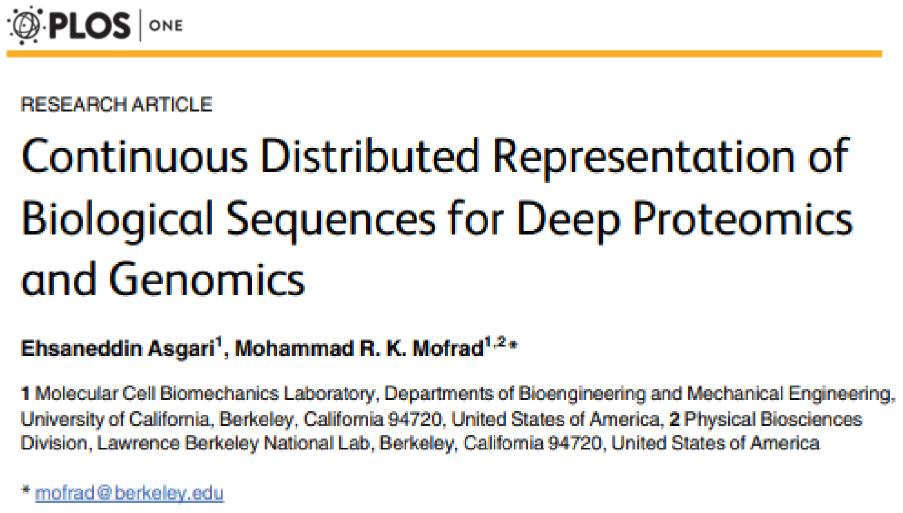
<단어>

Word2vec	Is	a	technique
The	word2vec	algorithm	uses
Once	trained	such	a
As	the	name	implies
…			



ProtVec – 문장 및 단어화

- **단백질의 아미노산 서열에 Word2Vec 적용**
  - 말뭉치(Corpus): Uniprot 단백질 데이터 베이스
    - -UniProt DB에는 약 56만개의 단백질이 존재함
  - 문장: 단백질 아미노산 서열
    - -띄어쓰기가 되어 있지 않은 문장
    - -ESSKVAITYADSGVSVDNGNNLVQTIKEMVRSTRRPGADSDI .....
  - 단어(word): N-gram으로 단어를 정의함
    - -3-gram으로 단어를 정의한 뒤, word2vec 모델로 단어벡터를 생성함
    - -ESS KVA ITY ADS GVS VDN GNN LVQ TIK EMV RST ...



PLOS | ONE

RESEARCH ARTICLE

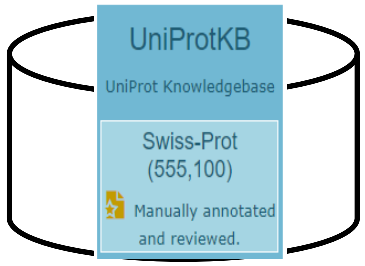
Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics

Ehsaneddin Asgari[1], Mohammad R. K. Mofrad[1,2]*

1 Molecular Cell Biomechanics Laboratory, Departments of Bioengineering and Mechanical Engineering, University of California, Berkeley, California 94720, United States of America, 2 Physical Biosciences Division, Lawrence Berkeley National Lab, Berkeley, California 94720, United States of America

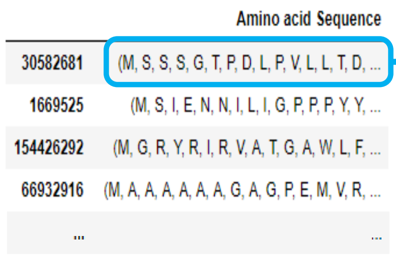
* mofrad@berkeley.edu

<말뭉치(Corpus)>



568,744

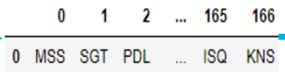
<문장>



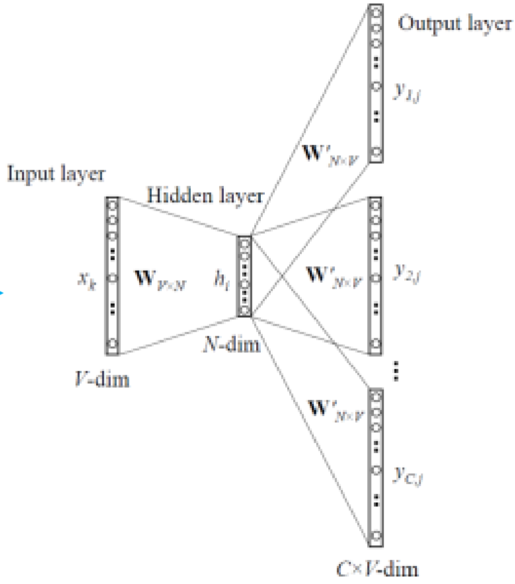
	Amino acid Sequence
30582681	(M, S, S, S, G, T, P, D, L, P, V, L, L, T, D, ...
1669525	(M, S, I, E, N, N, I, L, I, G, P, P, P, Y, Y, ...
154426292	(M, G, R, Y, R, I, R, V, A, T, G, A, W, L, F, ...
66932916	(M, A, A, A, A, A, A, G, A, G, P, E, M, V, R, ...
...	...

N-gram
(N=3)

<단어>



0	1	2	...	165	166
0 MSS	SGT	PDL	...	ISQ	KNS

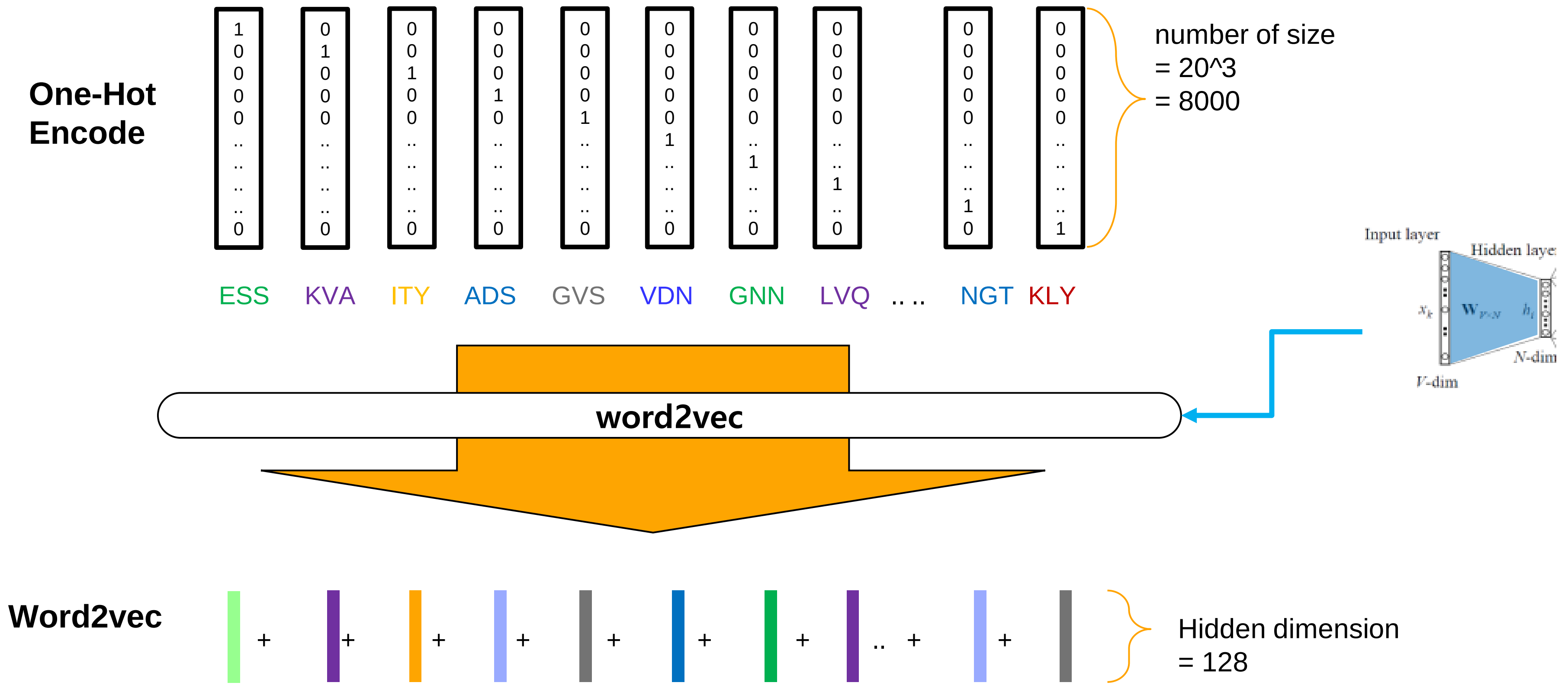


ProtVec – 적용

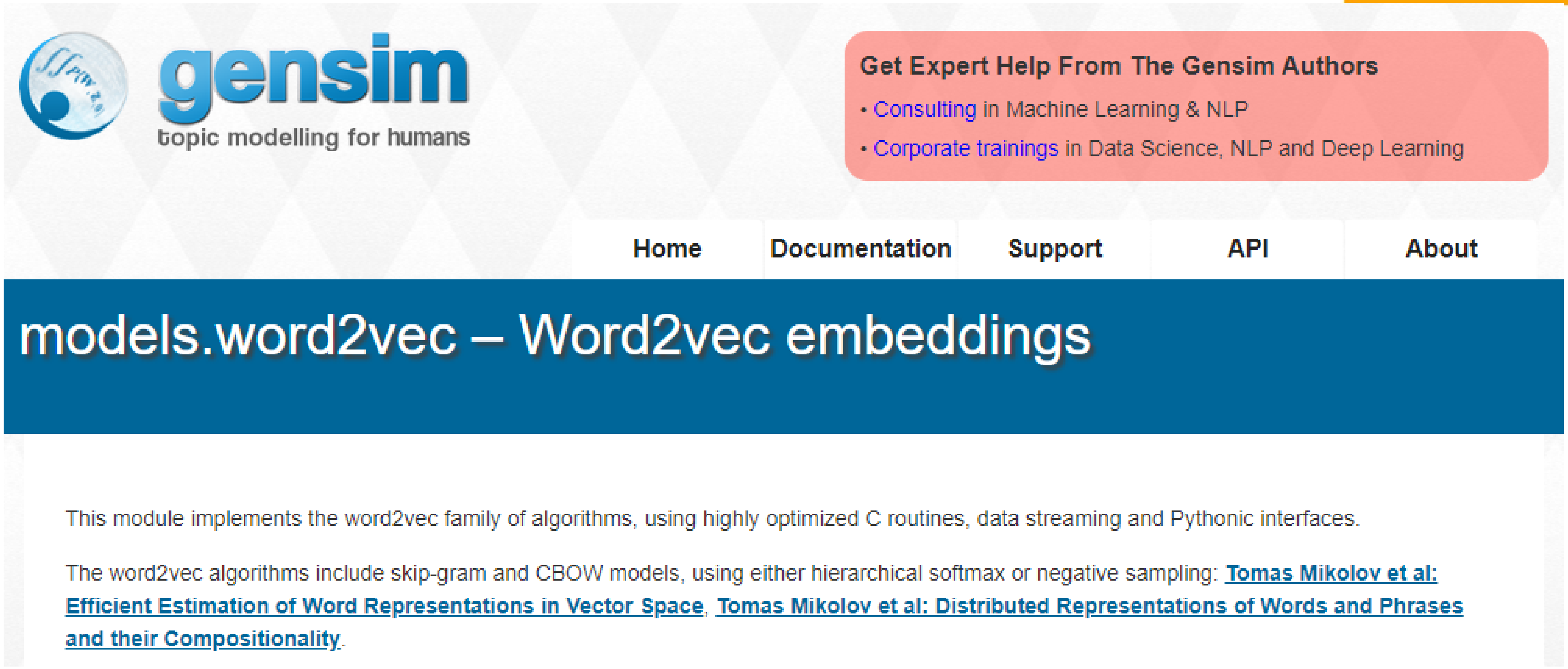
- Protein1

ESSKVAITYADSGVSVDNGNNLVQTIKEMVRSTRRPGADSDIGGFGGLFDLAQAGFRQNEDTLLVGATDGVGTKLIIAQETGIHNTVGIDLVAMNVND
LVVQGAEPLFFLDYFATGALDIQVASDFVSGVANGCIQSGCALVGGETSEMPGMYPPGHYDTNGTAVGAVLRQDILPKINEMAAGDVLLGLASSGVH
SNGFSLVRKIIQHVALPWDAPCPWDESKTLGEGILEPTKIYVKQLLPSIRQRLLLGLAHITGGGLVENIPRAIPDHLQARVDMSTWEVPRVFKWFGQAG
NVPHDDILRTFNMGVGMVLIVKRENVKAVCDSLTEEGEIIWELGSLQERPKDAPGCVIENGTKLY

- Protein (sub) sequence= k-words (k=3)



Codes for protein embedding



Get Expert Help From The Gensim Authors

- Consulting in Machine Learning & NLP
- Corporate trainings in Data Science, NLP and Deep Learning

models.word2vec – Word2vec embeddings

This module implements the word2vec family of algorithms, using highly optimized C routines, data streaming and Pythonic interfaces.

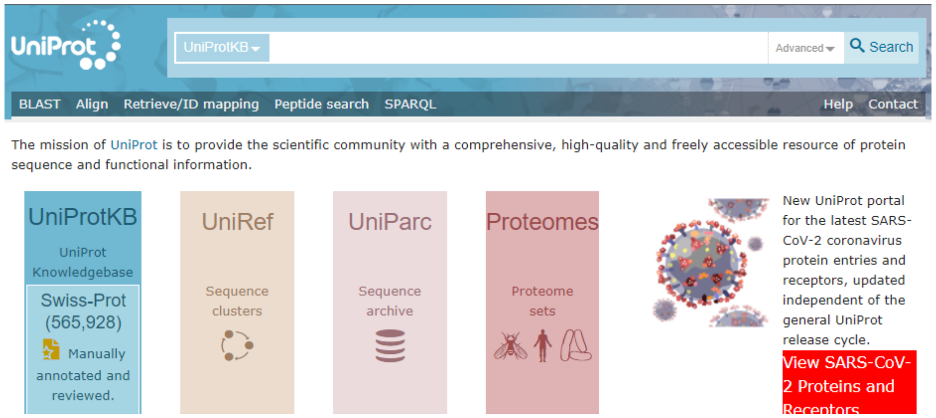
The word2vec algorithms include skip-gram and CBOW models, using either hierarchical softmax or negative sampling: Tomas Mikolov et al: Efficient Estimation of Word Representations in Vector Space, Tomas Mikolov et al: Distributed Representations of Words and Phrases and their Compositionality.

Codes for protein embedding

1. Get DB

```python
df_prot_seq = pd.read_csv('./in/prot/Chembl23_Table_prot-seq.csv', header=0)
df_prot_seq.head()
```

```
       target_ID                                          amino_seq
0  CHEMBL1907607  MSYSLYLAFVCLNLLAQRMCIQGNQFNVEVSRSDKLSLPGFENLTA...
1  CHEMBL2096683  MSYSLYLAFVCLNLLAQRMCIQGNQFNVEVSRSDKLSLPGFENLTA...
```



The mission of UniProt is to provide the scientific community with a comprehensive, high-quality and freely accessible resource of protein sequence and functional information.

UniProtKB
UniProt Knowledgebase
Swiss-Prot (565,928)
Manually annotated and reviewed.

UniRef
Sequence clusters

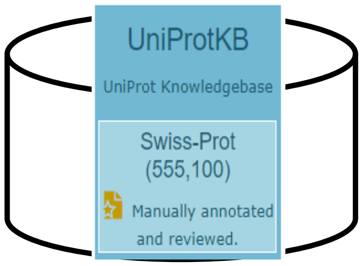
UniParc
Sequence archive

Proteomes
Proteome sets

New UniProt portal for the latest SARS-CoV-2 coronavirus protein entries and receptors, updated independent of the general UniProt release cycle.

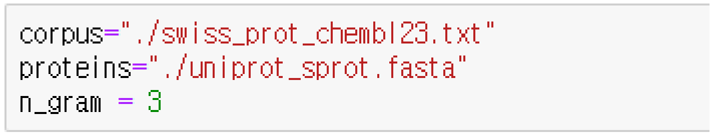
View SARS-CoV-2 Proteins and Receptors

<말뭉치(Corpus)>

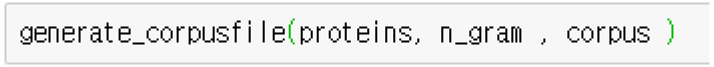


Codes for protein embedding

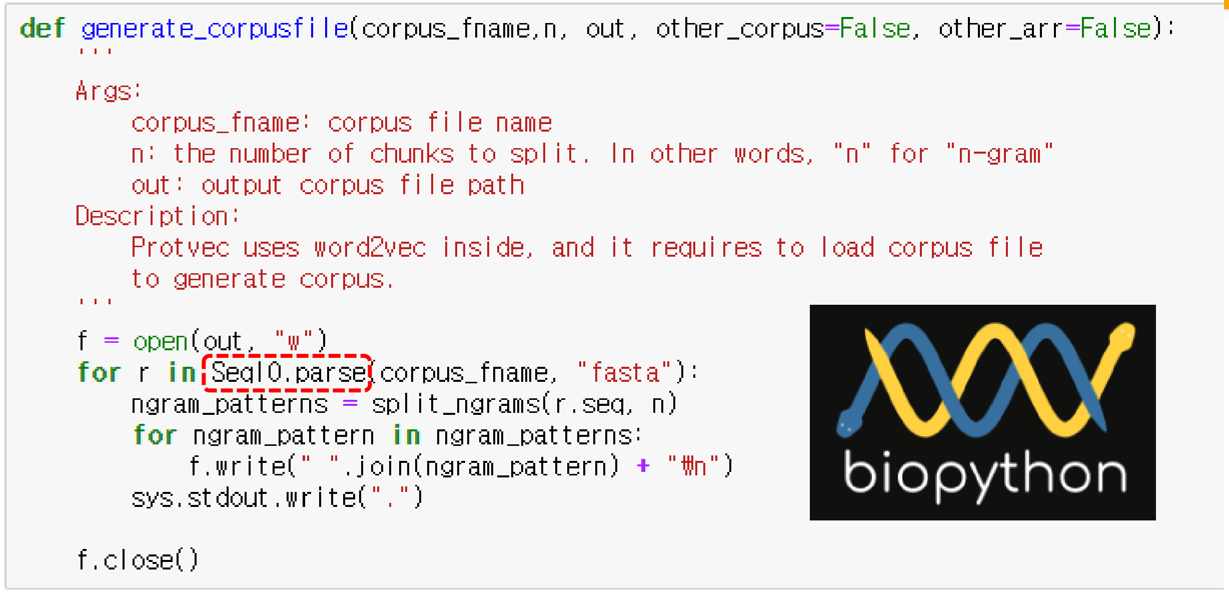
1.1 Build corpus



```
corpus="./swiss_prot_chembl23.txt"
proteins="./uniprot_sprot.fasta"
n_gram = 3
```



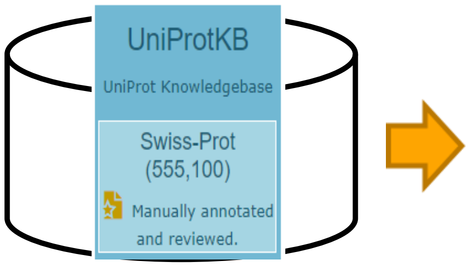
```
generate_corpusfile(proteins, n_gram , corpus )
```



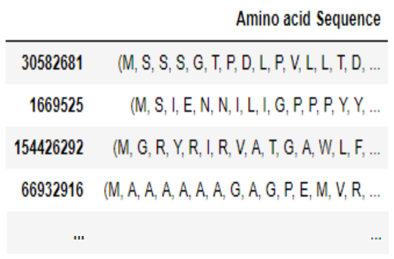
```
def generate_corpusfile(corpus_fname,n, out, other_corpus=False, other_arr=False):
    '''
    Args:
        corpus_fname: corpus file name
        n: the number of chunks to split. In other words, "n" for "n-gram"
        out: output corpus file path
    Description:
        Protvec uses word2vec inside, and it requires to load corpus file
        to generate corpus.
    '''
    f = open(out, "w")
    for r in SeqIO.parse(corpus_fname, "fasta"):
        ngram_patterns = split_ngrams(r.seq, n)
        for ngram_pattern in ngram_patterns:
            f.write(" ".join(ngram_pattern) + "₩n")
        sys.stdout.write(".")

    f.close()
```

biopython

<말뭉치(Corpus)>



<문장>



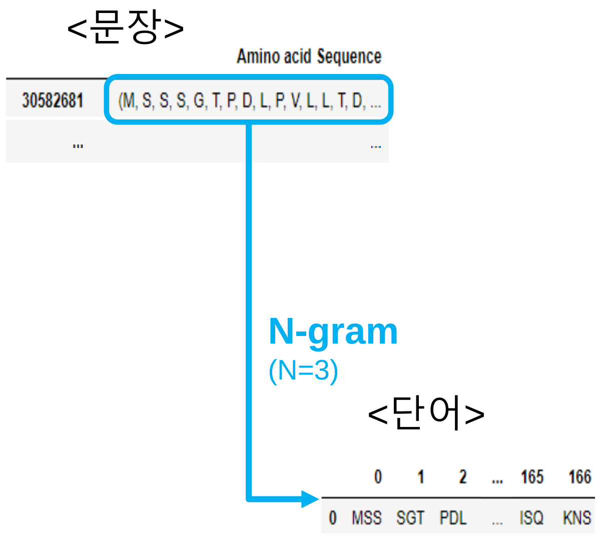
	Amino acid Sequence
30582681	(M, S, S, S, G, T, P, D, L, P, V, L, L, T, D, ...
1669525	(M, S, I, E, N, N, I, L, I, G, P, P, P, Y, Y, ...
154426292	(M, G, R, Y, R, I, R, V, A, T, G, A, W, L, F, ...
66932916	(M, A, A, A, A, A, A, G, A, G, P, E, M, V, R, ...
...	...

Codes for protein embedding

1.1 Build corpus

```
corpus="./swiss_prot_chembl23.txt"
proteins="./uniprot_sprot.fasta"
n_gram = 3
```

```
generate_corpusfile(proteins, n_gram , corpus )
```



<문장>
Amino acid Sequence
30582681
(M, S, S, S, G, T, P, D, L, P, V, L, L, T, D, ...
...
...
N-gram
(N=3)
<단어>
0 1 2 ... 165 166
0 MSS SGT PDL ... ISQ KNS

```
def generate_corpusfile(corpus_fname,n, out, other_corpus=False, other_arr=False):
    '''
    Args:
        corpus_fname: corpus file name
        n: the number of chunks to split. In other words, "n" for "n-gram"
        out: output corpus file path
    Description:
        Protvec uses word2vec inside, and it requires to load corpus file
        to generate corpus.
    '''
    f = open(out, "w")
    for r in SeqIO.parse(corpus_fname, "fasta"):
        ngram_patterns = split_ngrams(r.seq, n)
        for ngram_pattern in ngram_patterns:
            f.write(" ".join(ngram_pattern) + "₩n")
        sys.stdout.write(".")

    f.close()
```

```
def split_ngrams(seq, n):
    """
    'AGAMQSASM' => [['AGA', 'MQS', 'ASM'], ['GAM','QSA'], ['AMQ', 'SAS']]
    """
    seq_idxed=[]
    for idx in range(n):
        seq_idxed.append( zip(*[iter(seq[idx:])]*n) )
    str_ngrams = []

    for ngrams in seq_idxed:
        x = []
        for ngram in ngrams:
            x.append("".join(ngram) )
        str_ngrams.append(x)
    return str_ngrams
```

Codes for protein embedding

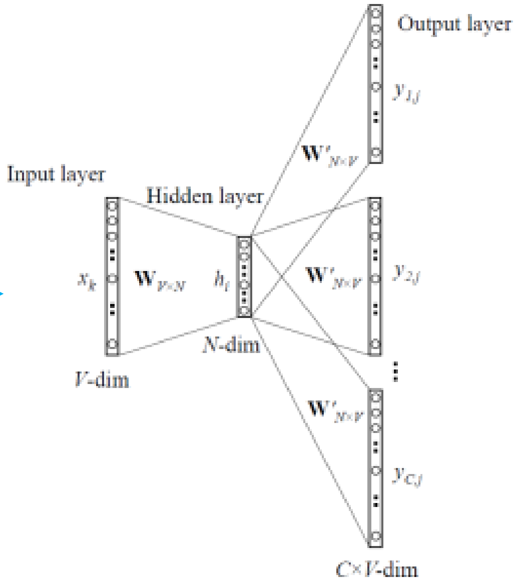
2 N-gram -> Protein Vector

```
vec_ret= ngram_to_Vec( ngram_ret, model_file )
len(vec_ret)
```

```
def ngram_to_Vec(protein_ngram, model_file, unseen='UNKNOWN'):
    iterNum=0
    protein_vec_sum=[]
    pv_swiss = biovec.models.load_protvec(model_file)
    keys = set( pv_swiss.wv.vocab.keys())
    ####
    if unseen:
        unseen_vec = pv_swiss[unseen]
    ####

    for i in xrange(len(protein_ngram)):
        channel=[]
        for j in xrange(len(protein_ngram[i])):
            sum_simple=0
            for idx in protein_ngram[i][j]:
                if idx not in keys:
                    sum_simple+= unseen_vec
                else:
                    sum_simple+= pv_swiss[idx]

            channel.append(sum_simple)
        protein_vec_sum.append(channel)
    return protein_vec_sum
```

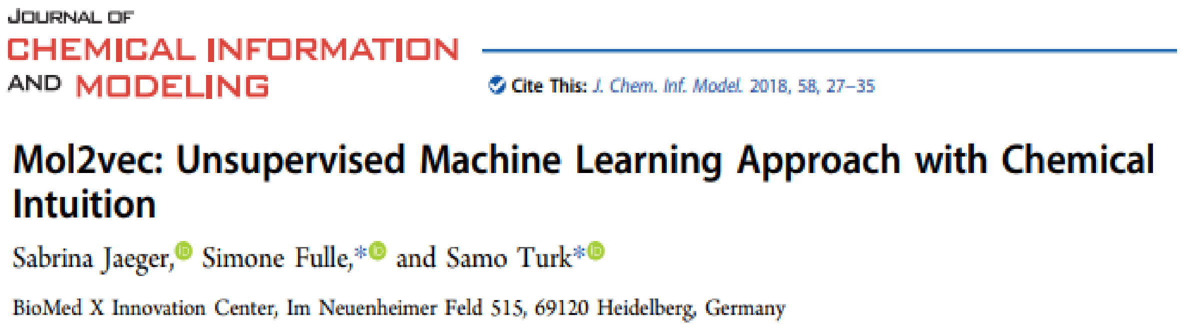
<단어>

	0	1	2	...	165	166
0	MSS	SGT	PDL	...	ISQ	KNS



Mol2vec – 문장 및 단어화

- **화합물의 2차원 그래프 데이터에 word2vec 적용**
  - 말뭉치(Corpus): ZINC 데이터 베이스
    - -ZINC DB에는 약 750만개의 화합물이 존재함
  - 문장: 약물 후보물질의 2차원 그래프
    - -띄어쓰기가 되어 있지 않은 문장
    - -ESSKVAITYADSGVSVDNGNNLVQTIKEMVRSTRRPGADSDI .....
  - 단어(word): Circular morgan 알고리즘으로 단어를 정의함
    - -약물의 각 Atom에서 특정 radius 범위 내의 node와 edge로 subgraph를 구성함
    - -word2vec 모델로 단어벡터를 생성함



JOURNAL OF CHEMICAL INFORMATION AND MODELING

Cite This: J. Chem. Inf. Model. 2018, 58, 27−35

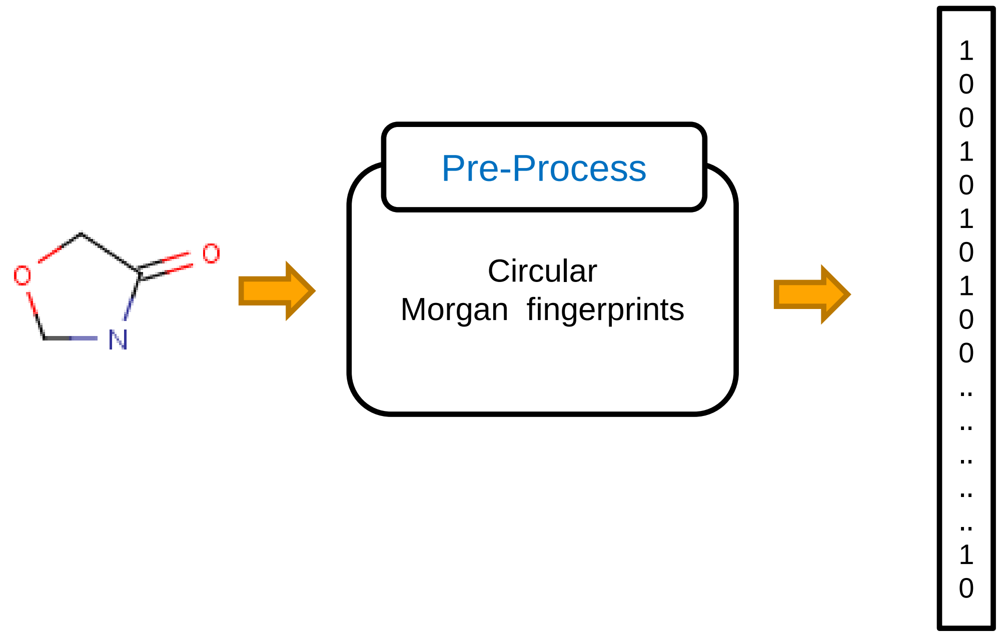
Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition

Sabrina Jaeger, Simone Fulle,* and Samo Turk*

BioMed X Innovation Center, Im Neuenheimer Feld 515, 69120 Heidelberg, Germany

Previous work: ECFPs

- **Overviews**
  - Extended-Connectivity Fingerprints (ECFPs) are circular topological fingerprints designed for molecular characterization, similarity searching, and structure-activity modeling.
  - They are among the most popular similarity search tools in drug discovery and they are effectively used in a wide variety of applications.

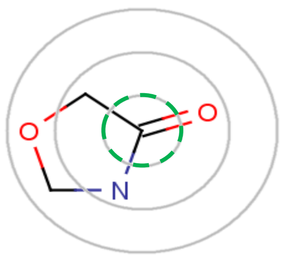


Previous work: ECFPs

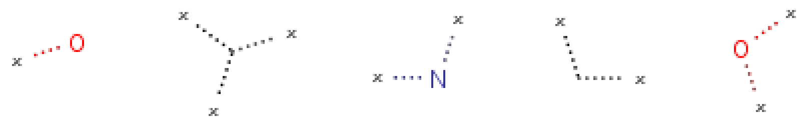
- **Generation Process**
  - **1. This identifier captures some local information.** **(e.g., atomic number, connection count, etc.)**
  - **2. local information are packed into a single integer value using hash function**
  - 3. Iterative updating of identifiers

1. Local info 추출

2. 고유값 부여



Diameter 0:



atomic number	1	0	1	0	1
connection count	1	3	2	2	2
etc	0	0	0	0	0
…	1	0	4	0	0
	0	0	1	1	1
	..	..	..	..	..
	1	0	4	2	2
	0	1	0	2	2

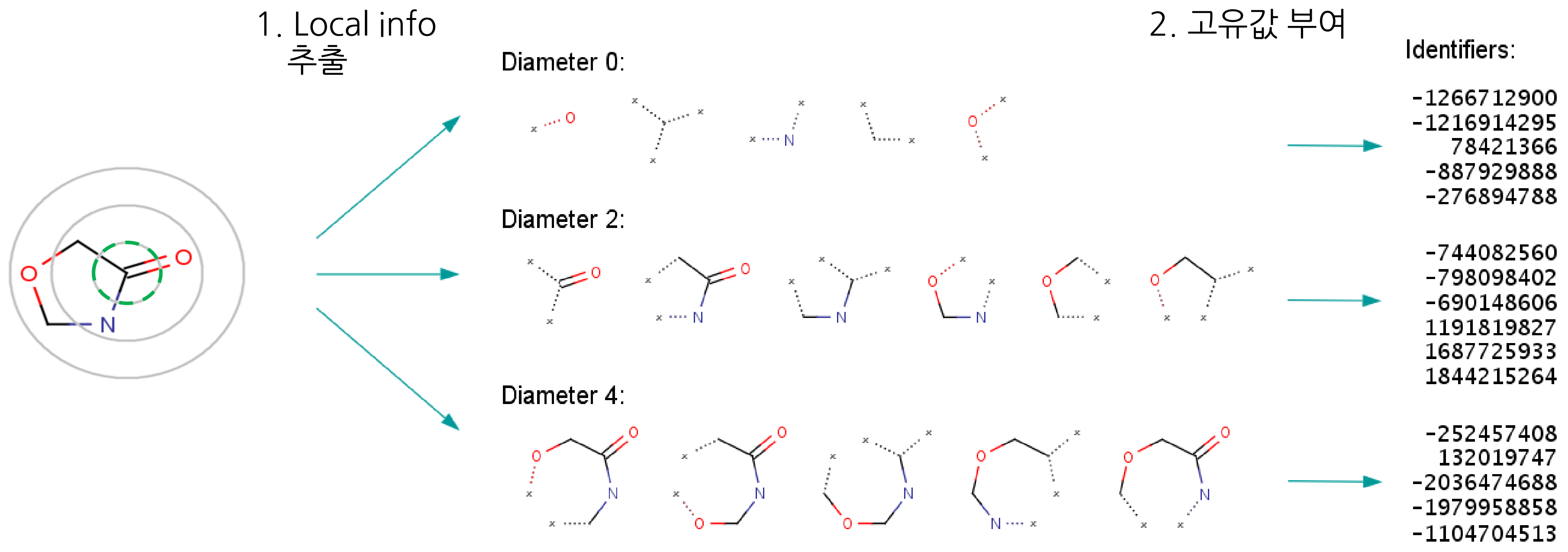
Identifiers:

```
-1266712900
-1216914295
   78421366
 -887929888
 -276894788
```

https://docs.chemaxon.com/display/docs/extended-connectivity-fingerprint-ecfp.md

Previous work: ECFPs

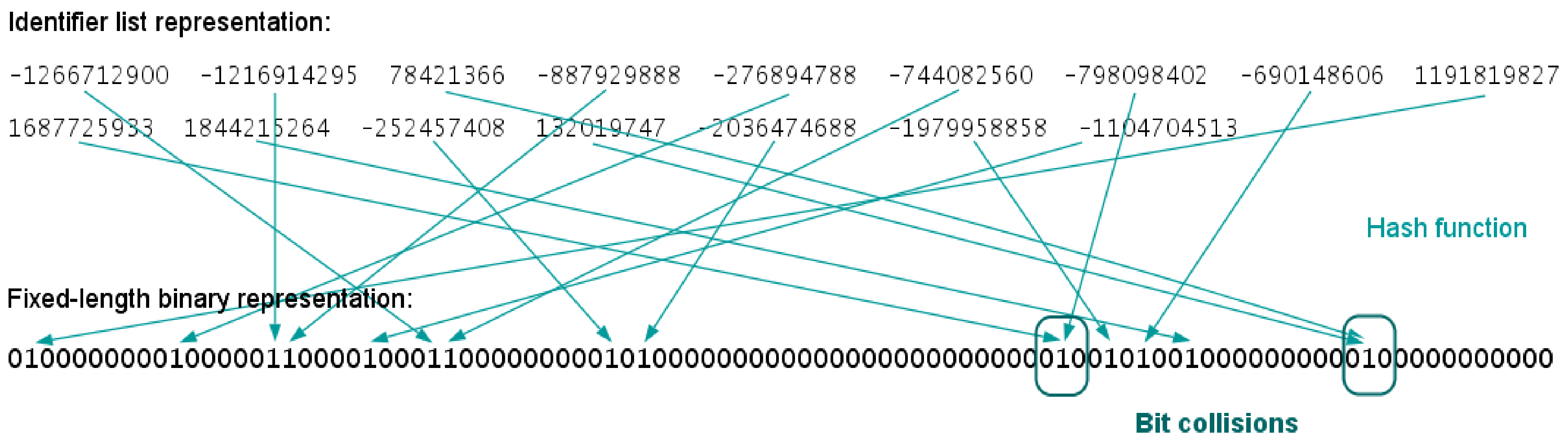
- **Generation Process**
  - **1. This identifier captures some local information.** **(e.g., atomic number, connection count, etc.)**
  - **2. local information are packed into a single integer value using hash function**
  - 3. Iterative updating of identifiers



Previous work: ECFPs

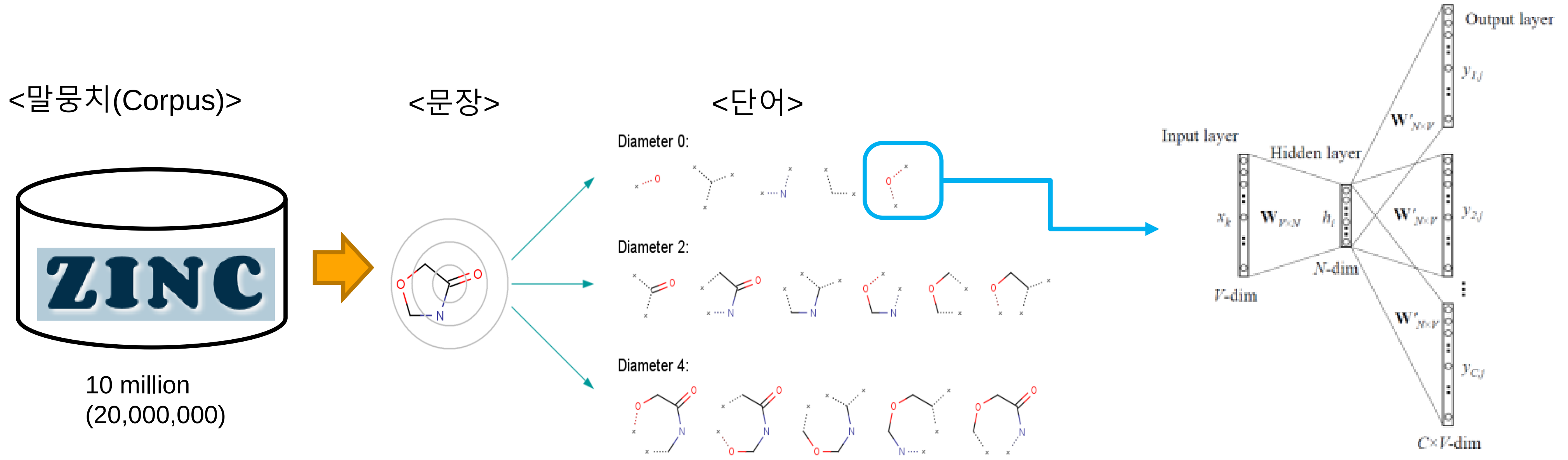
- **Generation Process**
  - 1. This identifier captures some local information. (e.g., atomic number, connection count, etc.)
  - 2. local information are packed into a single integer value using hash function
  - **3. Iterative updating of identifiers**

- **Limitation: Bit collisions**
  - Information loss: Bit collisions in fixed vector
  - Too long fixed vector length
  - Sparse Vector



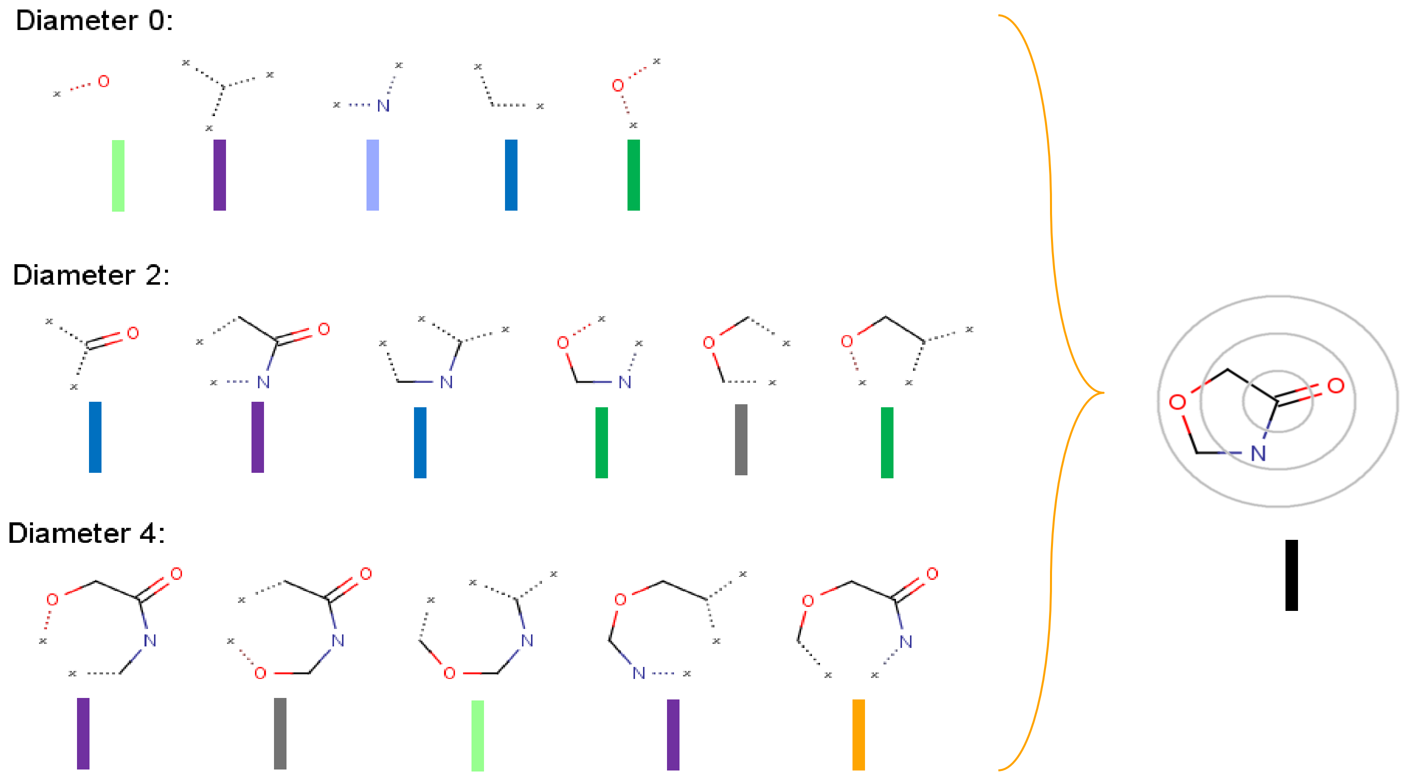
Mol2vec – 문장 및 단어화

- **약물 후보물질의 2차원 그래프 표현형에 word2vec 모델을 적용하여 특질을 추출함**
  - 문장: 약물 후보물질의 2차원 그래프
    - ZINC DB에는 약 55만개의 단백질이 존재함
  - 단어(word): Circular morgan 알고리즘으로 단어를 정의함
    - 약물의 각 Atom에서 특정 radius 범위 내의 node와 edge로 subgraph를 구성함

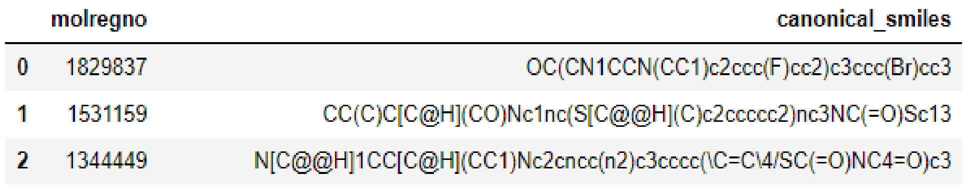


Mol2vec – 문장 및 단어화

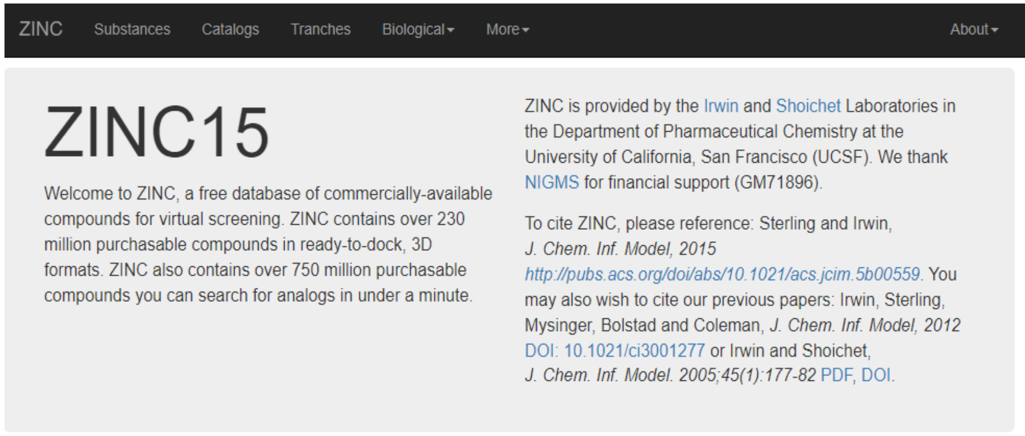
- 단어 벡터를 합하여 문장 벡터를 구축함



1. Get DB



	molregno	canonical_smiles
0	1829837	OC(CN1CCN(CC1)c2ccc(F)cc2)c3ccc(Br)cc3
1	1531159	CC(C)C[C@H](CO)Nc1nc(S[C@@H](C)c2ccccc2)nc3NC(=O)Sc13
2	1344449	N[C@@H]1CC[C@H](CC1)Nc2cncc(n2)c3cccc(\C=C\4/SC(=O)NC4=O)c3

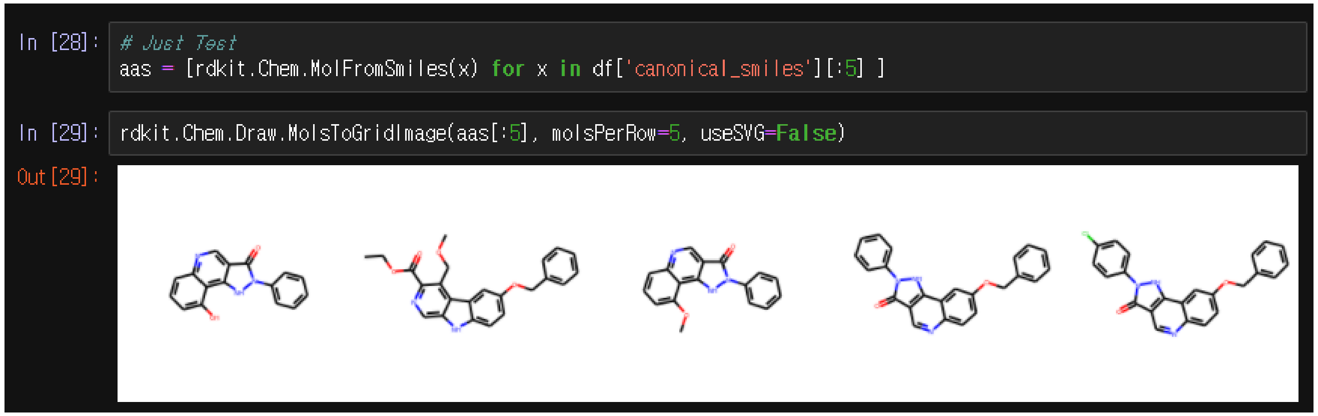


ZINC15

Welcome to ZINC, a free database of commercially-available compounds for virtual screening. ZINC contains over 230 million purchasable compounds in ready-to-dock, 3D formats. ZINC also contains over 750 million purchasable compounds you can search for analogs in under a minute.

ZINC is provided by the Irwin and Shoichet Laboratories in the Department of Pharmaceutical Chemistry at the University of California, San Francisco (UCSF). We thank NIGMS for financial support (GM71896).

To cite ZINC, please reference: Sterling and Irwin, *J. Chem. Inf. Model, 2015* *http://pubs.acs.org/doi/abs/10.1021/acs.jcim.5b00559*. You may also wish to cite our previous papers: Irwin, Sterling, Mysinger, Bolstad and Coleman, *J. Chem. Inf. Model, 2012* DOI: 10.1021/ci3001277 or Irwin and Shoichet, *J. Chem. Inf. Model. 2005;45(1):177-82* PDF, DOI.



```
In [28]: # Just Test
         aas = [rdkit.Chem.MolFromSmiles(x) for x in df['canonical_smiles'][:5] ]

In [29]: rdkit.Chem.Draw.MolsToGridImage(aas[:5], molsPerRow=5, useSVG=False)
Out[29]:
```

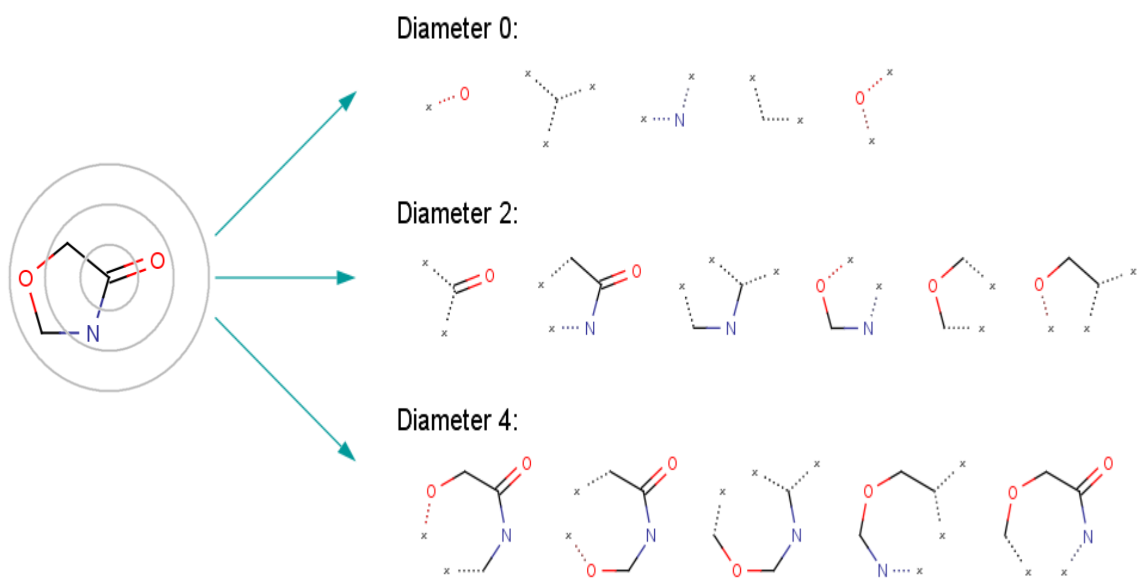
RDKit

Open-Source Cheminformatics and Machine Learning

1.1 Build corpus

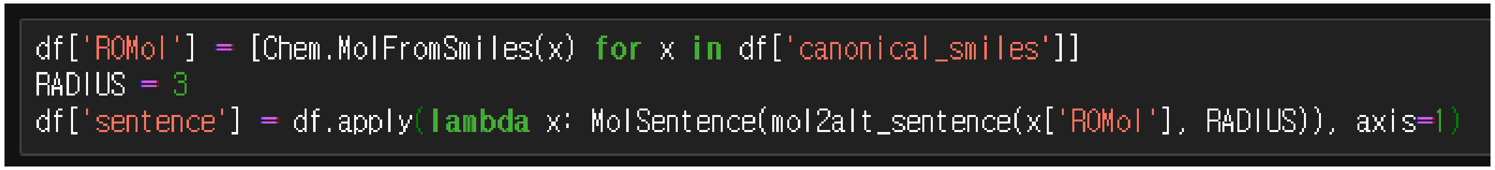
```python
df['ROMol'] = [Chem.MolFromSmiles(x) for x in df['canonical_smiles']]
RADIUS = 3
df['sentence'] = df.apply(lambda x: MolSentence(mol2alt_sentence(x['ROMol'], RADIUS)), axis=1)
```

Open-Source Cheminformatics
and Machine Learning

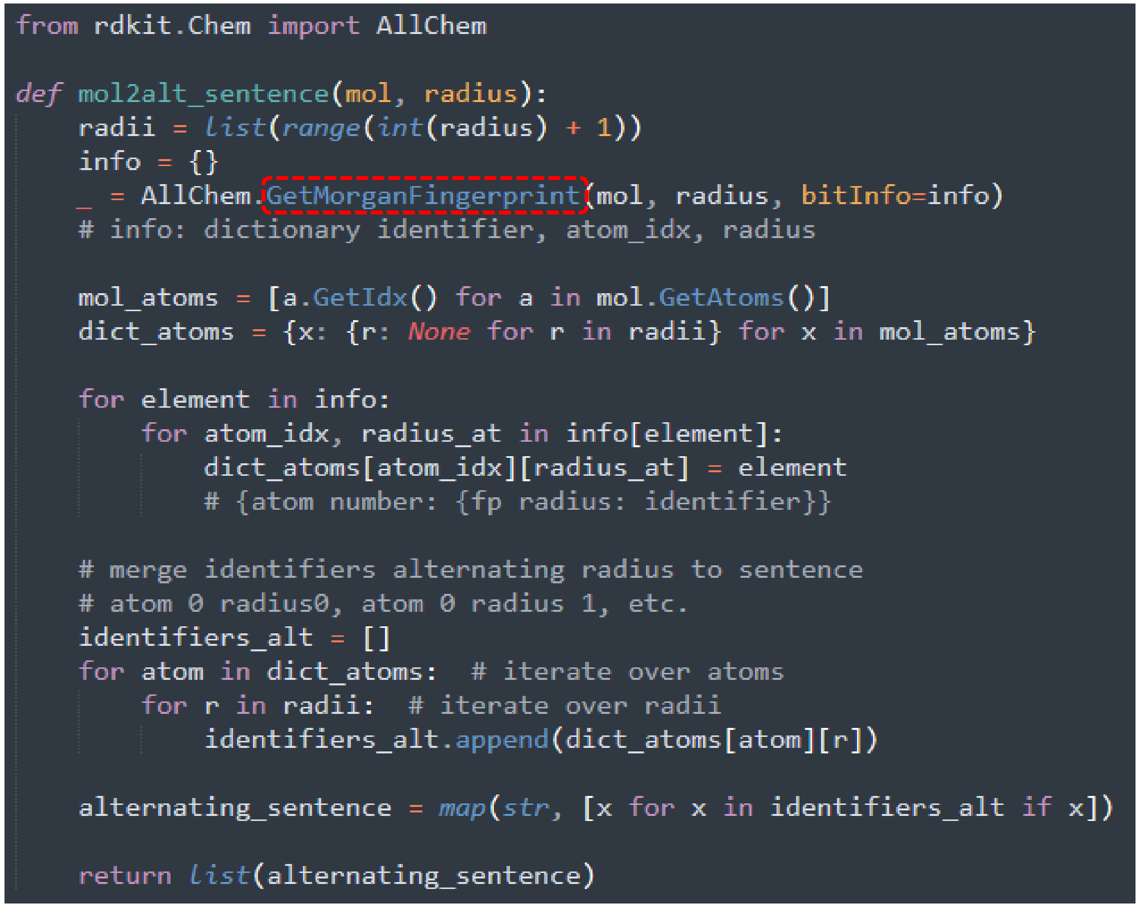


```
(864662311, 1542633699, 2245273601,
2782530898, 2245384272, 2258843522,
2092489639, 1634606847, 2968968094,
2803848648, 2968968094, 2803848648,
2092489639, 963029399, 2968968094,
2803848648, 2968968094, 2803848648,
3217380708, 2473389857, 3218693969,
951226070, 3218693969, 951226070,
3217380708, 1637836422, 882399112,
3337745083, 3218693969, 951226070,
3218693969, 951226070, 3217380708,
3579962709, 3218693969, 951226070,
3218693969, 951226070, 3217380708,
2646219661, 3612926680, 3632350815,
3218693969, 951226070, 3218693969,
951226070)
```

1.1 Build corpus



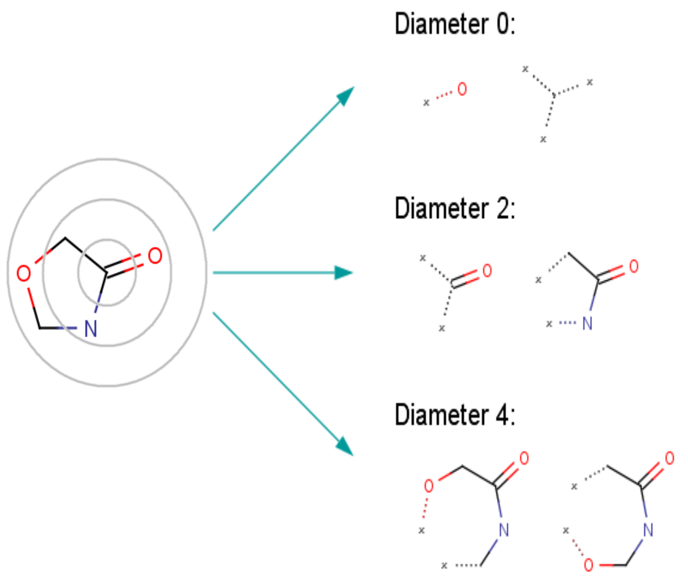
```
df['ROMol'] = [Chem.MolFromSmiles(x) for x in df['canonical_smiles']]
RADIUS = 3
df['sentence'] = df.apply(lambda x: MolSentence(mol2alt_sentence(x['ROMol'], RADIUS)), axis=1)
```



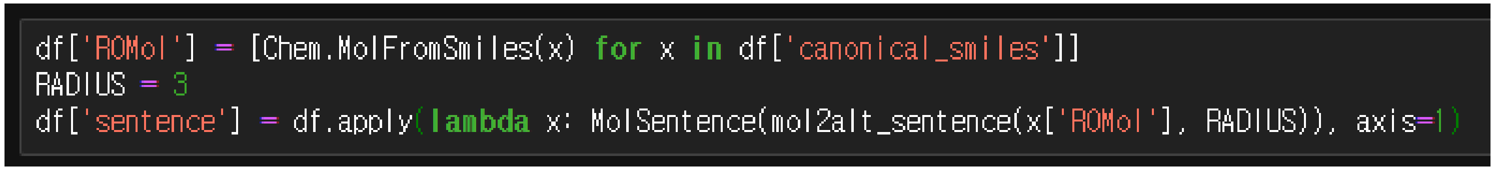
```
from rdkit.Chem import AllChem

def mol2alt_sentence(mol, radius):
    radii = list(range(int(radius) + 1))
    info = {}
    _ = AllChem.GetMorganFingerprint(mol, radius, bitInfo=info)
    # info: dictionary identifier, atom_idx, radius

    mol_atoms = [a.GetIdx() for a in mol.GetAtoms()]
    dict_atoms = {x: {r: None for r in radii} for x in mol_atoms}

    for element in info:
        for atom_idx, radius_at in info[element]:
            dict_atoms[atom_idx][radius_at] = element
            # {atom number: {fp radius: identifier}}

    # merge identifiers alternating radius to sentence
    # atom 0 radius0, atom 0 radius 1, etc.
    identifiers_alt = []
    for atom in dict_atoms:  # iterate over atoms
        for r in radii:  # iterate over radii
            identifiers_alt.append(dict_atoms[atom][r])

    alternating_sentence = map(str, [x for x in identifiers_alt if x])

    return list(alternating_sentence)
```

RDKit

Open-Source Cheminformatics and Machine Learning

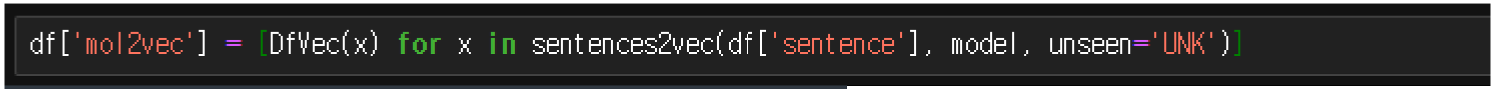


1.1 Build corpus

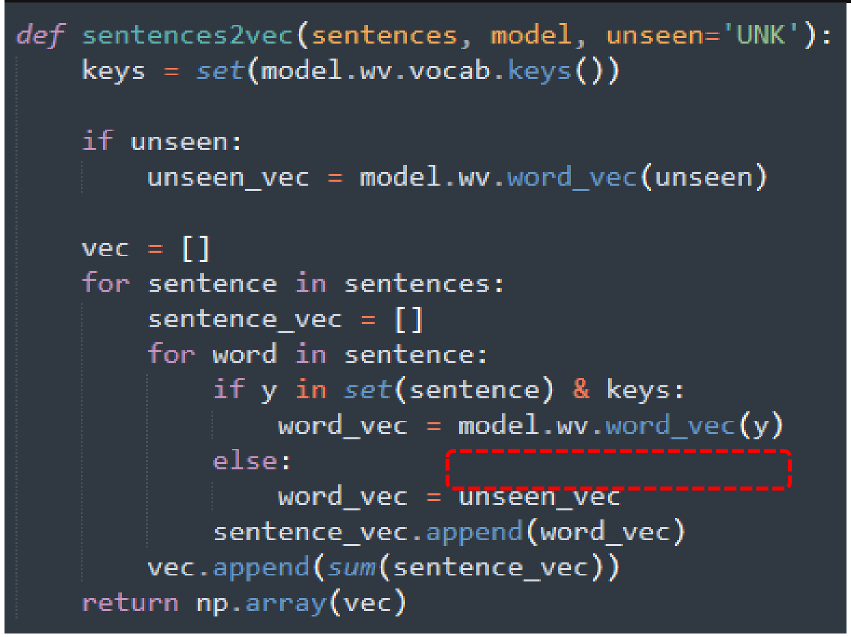


```python
df['ROMol'] = [Chem.MolFromSmiles(x) for x in df['canonical_smiles']]
RADIUS = 3
df['sentence'] = df.apply(lambda x: MolSentence(mol2alt_sentence(x['ROMol'], RADIUS)), axis=1)
```

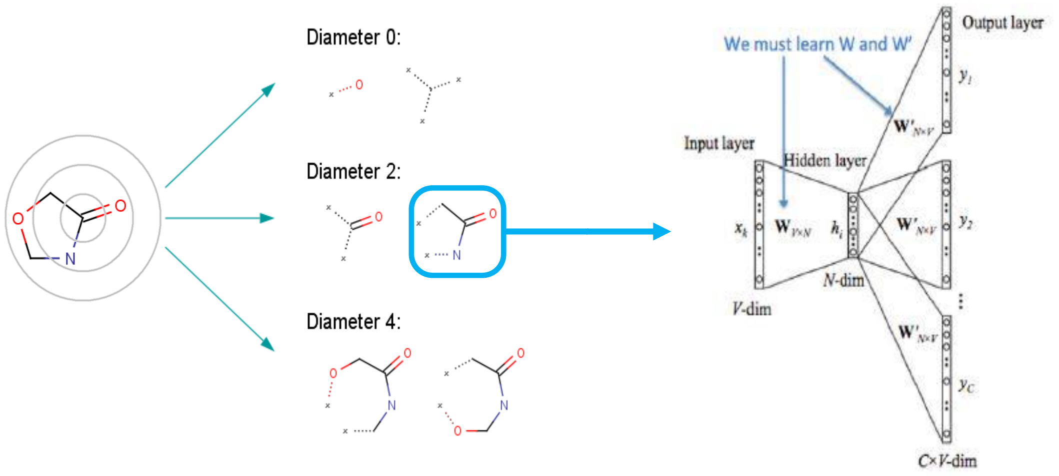
1.2 Extract vector



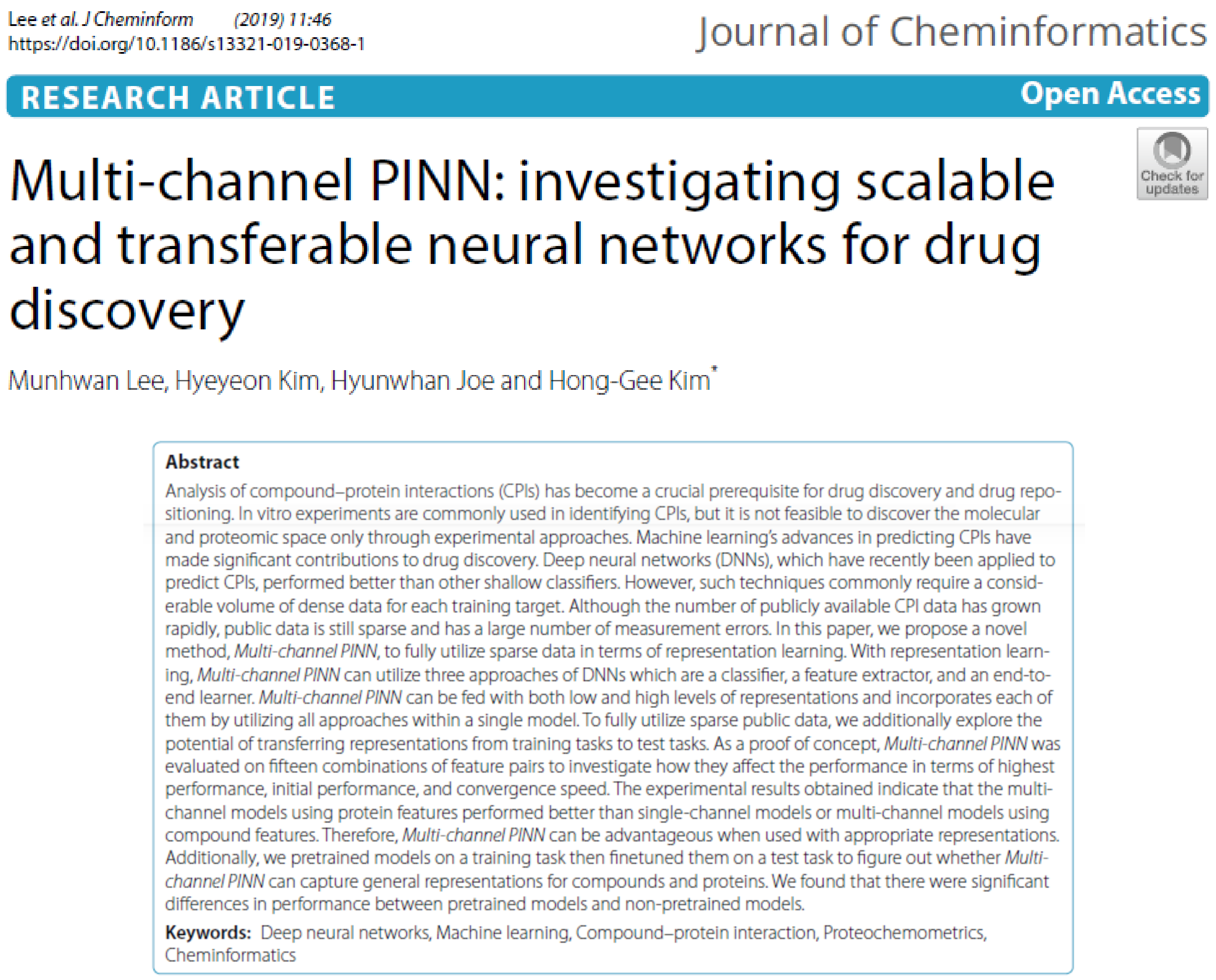
```python
df['mol2vec'] = [DfVec(x) for x in sentences2vec(df['sentence'], model, unseen='UNK')]
```



```python
def sentences2vec(sentences, model, unseen='UNK'):
    keys = set(model.wv.vocab.keys())

    if unseen:
        unseen_vec = model.wv.word_vec(unseen)

    vec = []
    for sentence in sentences:
        sentence_vec = []
        for word in sentence:
            if y in set(sentence) & keys:
                word_vec = model.wv.word_vec(y)
            else:
                word_vec = unseen_vec
            sentence_vec.append(word_vec)
        vec.append(sum(sentence_vec))
    return np.array(vec)
```



문장 벡터 재구성



Lee *et al. J Cheminform* (2019) 11:46
https://doi.org/10.1186/s13321-019-0368-1

Journal of Cheminformatics

RESEARCH ARTICLE **Open Access**

Multi-channel PINN: investigating scalable and transferable neural networks for drug discovery

Munhwan Lee, Hyeyeon Kim, Hyunwhan Joe and Hong-Gee Kim*

Abstract
Analysis of compound–protein interactions (CPIs) has become a crucial prerequisite for drug discovery and drug repositioning. In vitro experiments are commonly used in identifying CPIs, but it is not feasible to discover the molecular and proteomic space only through experimental approaches. Machine learning's advances in predicting CPIs have made significant contributions to drug discovery. Deep neural networks (DNNs), which have recently been applied to predict CPIs, performed better than other shallow classifiers. However, such techniques commonly require a considerable volume of dense data for each training target. Although the number of publicly available CPI data has grown rapidly, public data is still sparse and has a large number of measurement errors. In this paper, we propose a novel method, *Multi-channel PINN*, to fully utilize sparse data in terms of representation learning. With representation learning, *Multi-channel PINN* can utilize three approaches of DNNs which are a classifier, a feature extractor, and an end-to-end learner. *Multi-channel PINN* can be fed with both low and high levels of representations and incorporates each of them by utilizing all approaches within a single model. To fully utilize sparse public data, we additionally explore the potential of transferring representations from training tasks to test tasks. As a proof of concept, *Multi-channel PINN* was evaluated on fifteen combinations of feature pairs to investigate how they affect the performance in terms of highest performance, initial performance, and convergence speed. The experimental results obtained indicate that the multi-channel models using protein features performed better than single-channel models or multi-channel models using compound features. Therefore, *Multi-channel PINN* can be advantageous when used with appropriate representations. Additionally, we pretrained models on a training task then finetuned them on a test task to figure out whether *Multi-channel PINN* can capture general representations for compounds and proteins. We found that there were significant differences in performance between pretrained models and non-pretrained models.

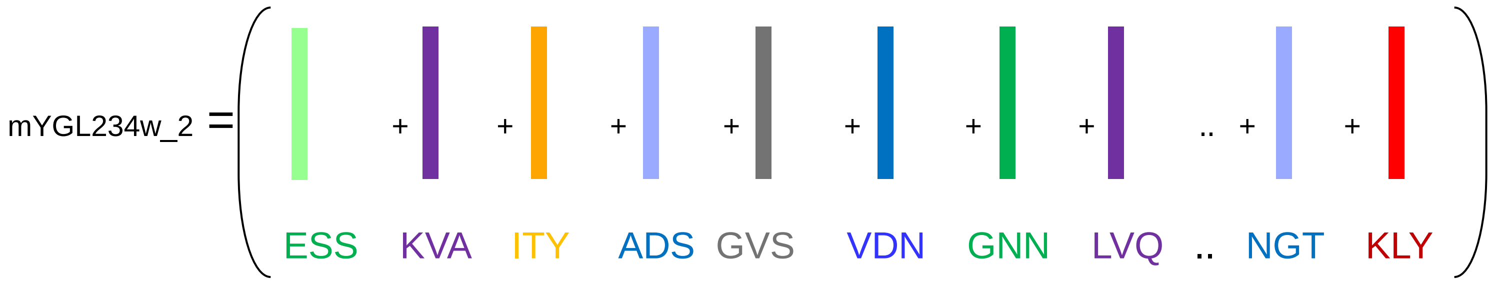
Keywords: Deep neural networks, Machine learning, Compound–protein interaction, Proteochemometrics, Cheminformatics

문장 벡터 재구성: 단순합

- protein total sequence= sentence

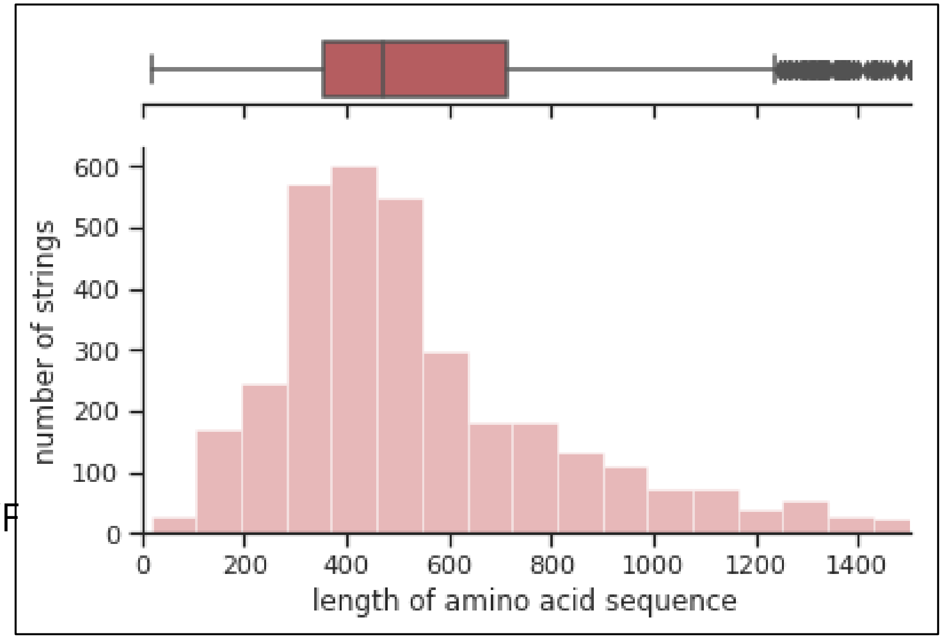
Example : COG Family150 - YGL234w_2

ESSKVAITYADSGVSVDNGNNLVQTIKEMVRSTRRPGADSDIGGFGGLFDLAQAGFRQNEDTLLVGATDGVGTKLIIAQETGIH
NTVGIDLVAMNVNDLVVQGAEPLFFLDYFATGALDIQVASDFVSGVANGCIQSGCALVGGETSEMPGMYPPGHYDTNGTAV
GAVLRQDILPKINEMAAGDVLLGLASSGVHSNGFSLVRKIIQHVALPWDAPCPWDESKTLGEGILEPTKIYVKQLLPSIRQRLLLG
LAHITGGGLVENIPRAIPDHLQARVDMSTWEVPRVFKWFGQAGNVPHDDILRTFNMGVGMVLIVKRENVKAVCDSLTEEGEII
WELGSLQERPKDAPGCVIENGTKLY



문장 벡터 재구성: 단순합

- 인간 문장 내의 평균 단어 수: 15-20 단어
- 짧은 아미노산 서열 사례: 10 chars
  - Erythrocyte membrane glycopeptide
  - “CEGHSHDHGA”



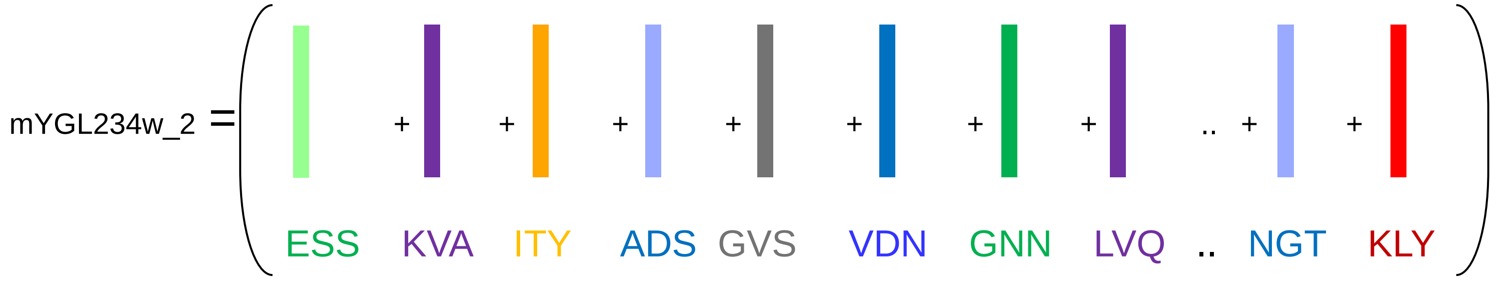
- 긴 아미노산 서열 사례: 34,350 chars
  - Titan
  - “MTTQAPTFTQ PLQSVVVLEG STATFEAHIS GFPVPEVSWF RDGQVISTST LPGVQISFSD GRAKLTIPAV TKANSGRYSL KATNGSGQAT STAELLVKAE …”

문장 벡터 재구성: 산술평균

- protein total sequence= sentence

Example : COG Family150 - YGL234w_2

ESSKVAITYADSGVSVDNGNNLVQTIKEMVRSTRRPGADSDIGGFGGLFDLAQAGFRQNEDTLLVGATDGVGTKLIIAQETGIH NTVGIDLEMPGMYPPQDILPKINKIYVKQLLPSIRQRLLLGLGEIIWELGSLQERPKDAPGCVIE .... NGTKLY

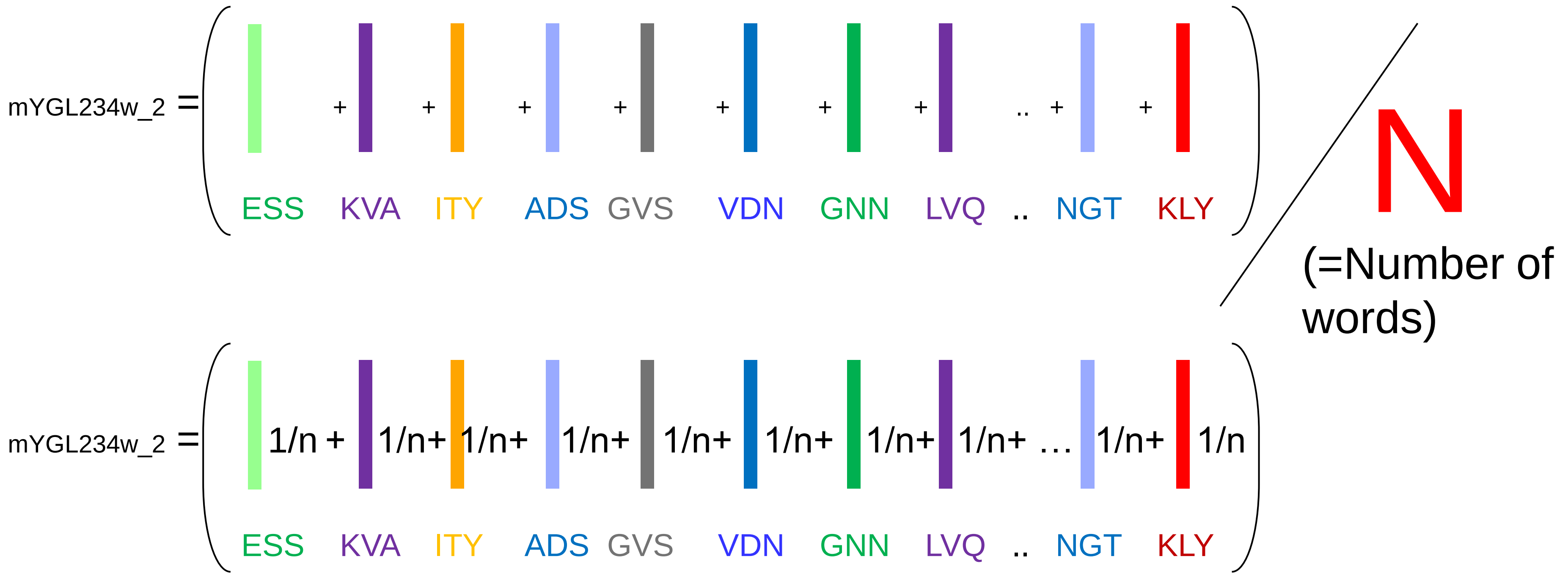


문장 벡터 재구성: 산술평균

- protein total sequence= sentence

Example : COG Family150 - YGL234w_2

ESSKVAITYADSGVSVDNGNNLVQTIKEMVRSTRRPGADSDIGGFGGLFDLAQAGFRQNEDTLLVGATDGVGTKLIIAQETGIH NTVGIDLEMPGMYPPQDILPKINKIYVKQLLPSIRQRLLLGLGEIIWELGSLQERPKDAPGCVIE .... NGTKLY



mYGL234w_2 =
+ + + + + + + .. + +
ESS KVA ITY ADS GVS VDN GNN LVQ .. NGT KLY
N
(=Number of words)
mYGL234w_2 =
1/n + 1/n+ 1/n+ 1/n+ 1/n+ 1/n+ 1/n+ 1/n+ … 1/n+ 1/n
ESS KVA ITY ADS GVS VDN GNN LVQ .. NGT KLY

문장 벡터 재구성: TF-IDF

- **단어 고유의 중요성을 부여**
  - 단어 고유의 중요성은 아래의 기준에서 더 높아짐
    - 특정 단어가 현재 문서(문장)에서 많이 사용됨 (TF)
    - 특정 단어가 다른 문서(문장)에서 많이 사용되지 않음 (IDF)

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

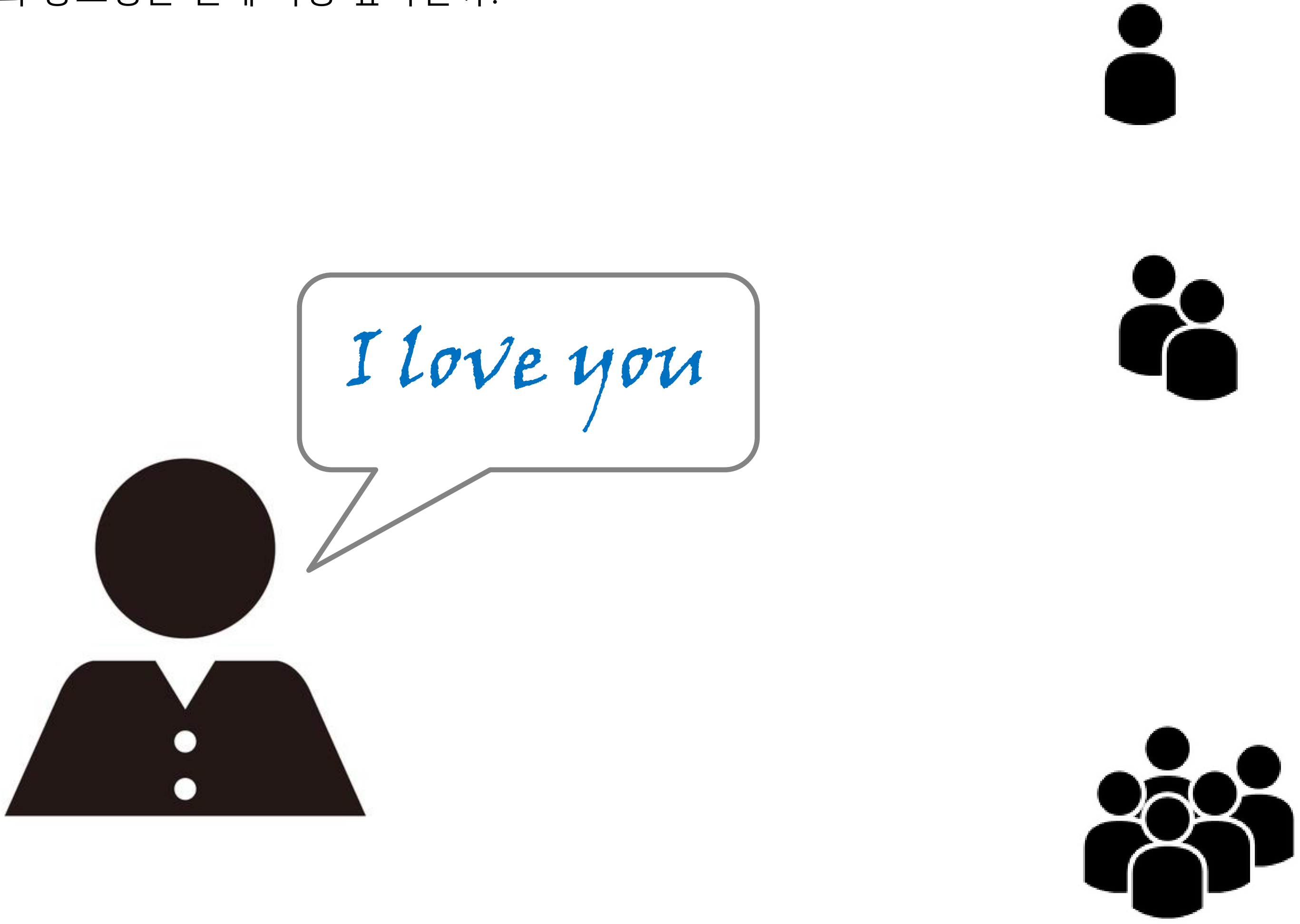
$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

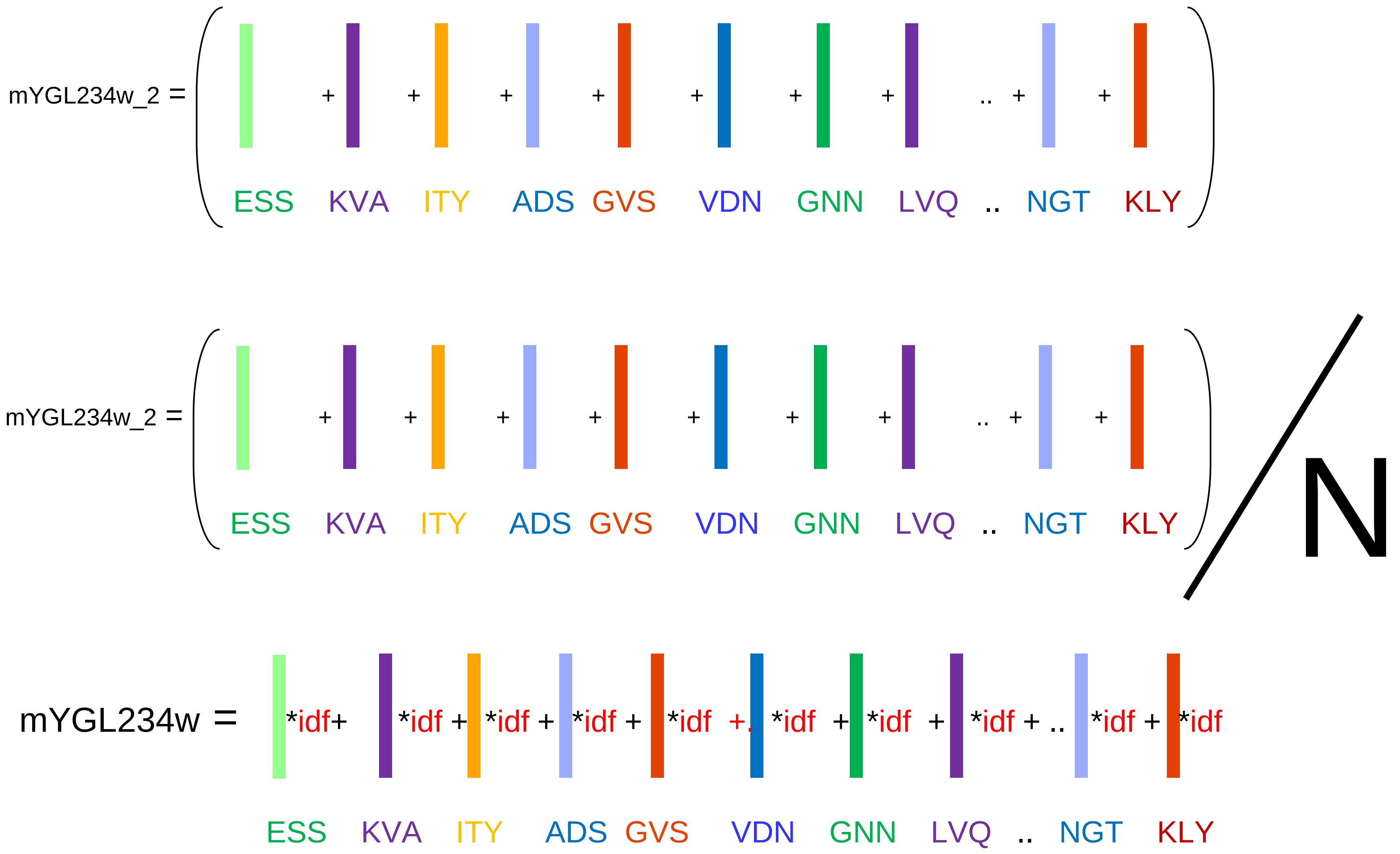
N = total number of documents

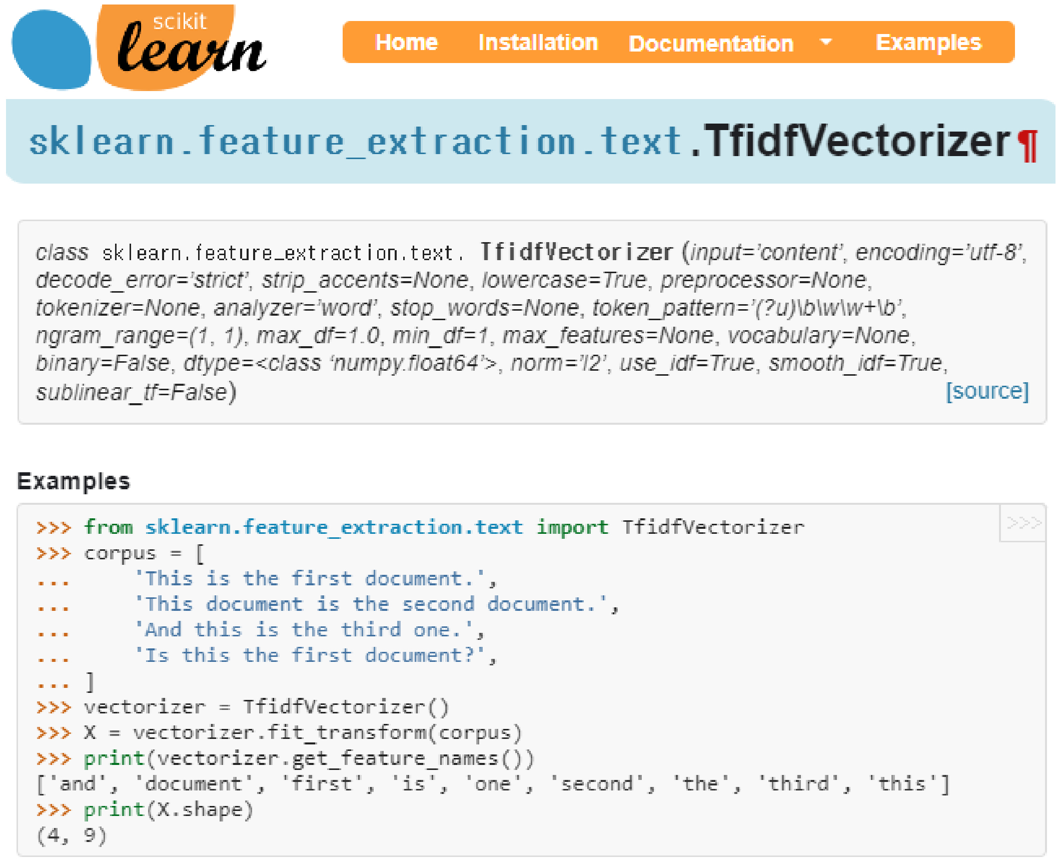
문장 벡터 재구성: TF-IDF

- **단어 고유의 중요성을 부여**
  - LOVE 의 중요성은 언제 가장 높아질까?



문장 벡터 재구성: 산술평균과 TF-IDF


mYGL234w_2 =
+ + + + + + + .. + +
ESS KVA ITY ADS GVS VDN GNN LVQ .. NGT KLY
mYGL234w_2 =
+ + + + + + + .. + +
ESS KVA ITY ADS GVS VDN GNN LVQ .. NGT KLY
N
mYGL234w =
*idf+ *idf + *idf + *idf + *idf +. *idf + *idf + *idf + .. *idf + *idf
ESS KVA ITY ADS GVS VDN GNN LVQ .. NGT KLY

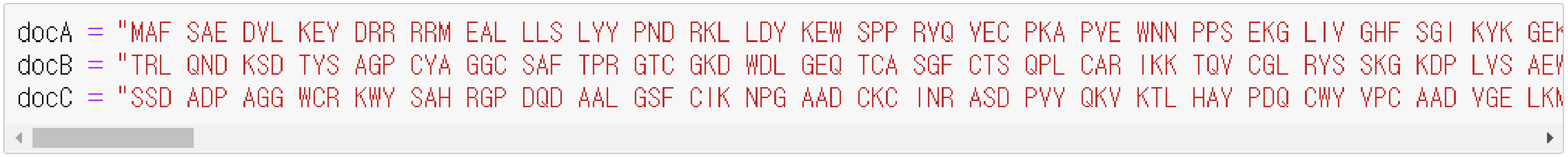


sklearn.feature_extraction.text.TfidfVectorizer¶

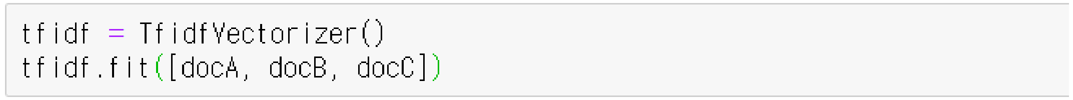
class sklearn.feature_extraction.text.**TfidfVectorizer**(*input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, analyzer='word', stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.float64'>, norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=False*) [source]

Examples

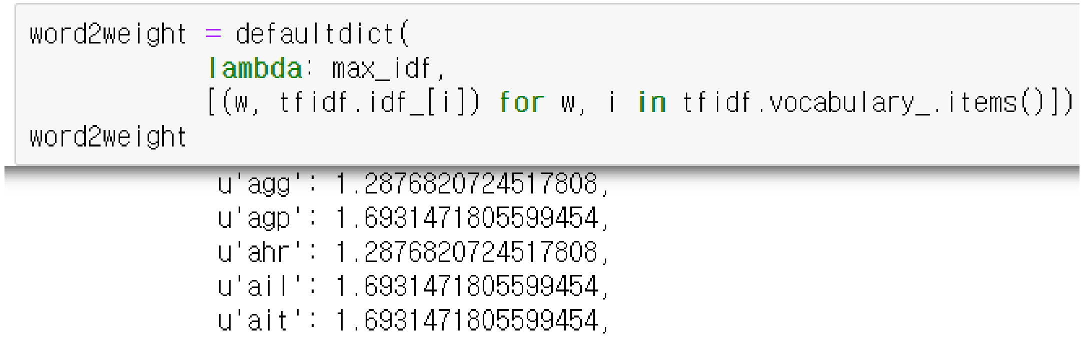
```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> corpus = [
...     'This is the first document.',
...     'This document is the second document.',
...     'And this is the third one.',
...     'Is this the first document?',
... ]
>>> vectorizer = TfidfVectorizer()
>>> X = vectorizer.fit_transform(corpus)
>>> print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
>>> print(X.shape)
(4, 9)
```



```
docA = "MAF SAE DVL KEY DRR RRM EAL LLS LYY PND RKL LDY KEW SPP RVQ YEC PKA PVE WNN PPS EKG LIV GHF SGI KYK GE
docB = "TRL QND KSD TYS AGP CYA GGC SAF TPR GTC GKD WDL GEQ TCA SGF CTS QPL CAR IKK TQV CGL RYS SKG KDP LVS AE
docC = "SSD ADP AGG WCR KWY SAH RGP DQD AAL GSF CIK NPG AAD CKC INR ASD PVY QKV KTL HAY PDQ CWY VPC AAD VGE LK
```



```
tfidf = TfidfVectorizer()
tfidf.fit([docA, docB, docC])
```



```
word2weight = defaultdict(
            lambda: max_idf,
            [(w, tfidf.idf_[i]) for w, i in tfidf.vocabulary_.items()])
word2weight
```

```
             u'agg': 1.2876820724517808,
             u'agp': 1.6931471805599454,
             u'ahr': 1.2876820724517808,
             u'ail': 1.6931471805599454,
             u'ait': 1.6931471805599454,
```

THANK YOU FOR LISTENING

ANY QUESTIONS?