O`ZBEK TILI KORPUSI MATNLARINI QAYTA ISHLASH USULLARI

Botir Elov[1], Shahlo Hamroyeva[2], Ruhillo Alayev[3], Zilola Xusainova[4], Umidjon Yodgorov[5]

[1]Alisher Navoiy nomidagi Toshkent davlat o'zbek tili va adabiyoti universiteti, Kompyuter lingvistikasi va raqamli texnologiyalar kafedrasi mudiri, t.f.f.d. dotsent, email: elov@navoiy-uni.uz
[2]Alisher Navoiy nomidagi Toshkent davlat o'zbek tili va adabiyoti universiteti, Kompyuter lingvistikasi va raqamli texnologiyalar kafedrasi f.f.d., dotsent, email: shaxlo.xamrayeva@navoiy-uni.uz
[3]Mirzo Ulug'bek nomidagi O'zbekiston Milliy universiteti, t.f.f.d, email: mr.ruhillo@gmail.com
[4]Alisher Navoiy nomidagi Toshkent davlat o'zbek tili va adabiyoti universiteti, Kompyuter lingvistikasi va raqamli texnologiyalar kafedrasi tayanch doktoranti, email: xusainovazilola@navoiy-uni.uz
[5]Alisher Navoiy nomidagi Toshkent davlat o'zbek tili va adabiyoti universiteti, Kompyuter lingvistikasi va raqamli texnologiyalar kafedrasi o'qituvchisi, email: yodgorov@navoiy-uni.uz

K E Y W O R D S

O`zbek tili korpusi, matnlarini qayta ishlash, Word2Vec, CBOW, Skip-Gram, GloVe, ELMO, BERT.

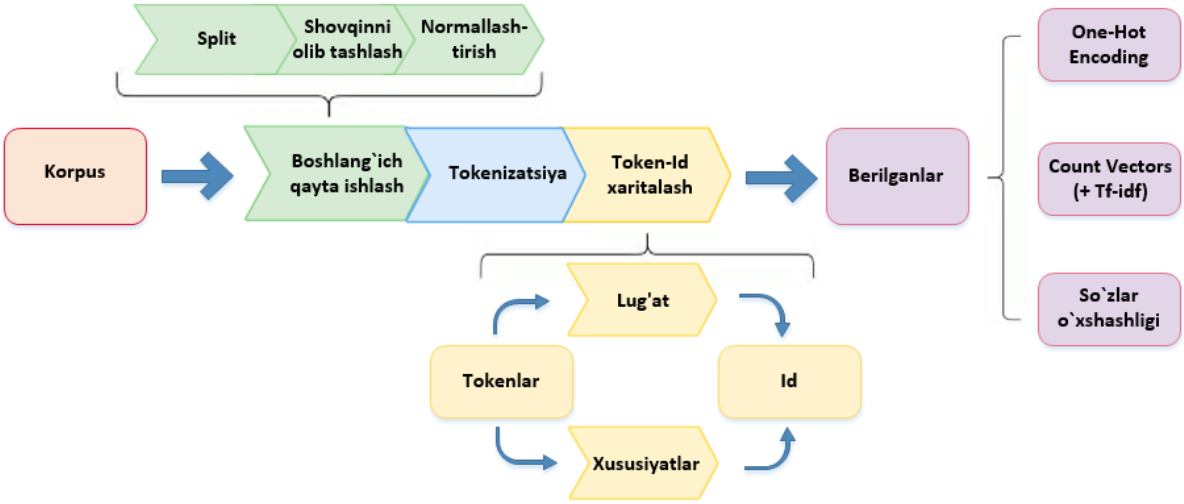
A B S T R A C T

Kompyuterlar raqamli yoki sonli ko`rinishdagi ma`lumotlarni qayta ishlashga mo`ljallangan. Ammo ma'lumotlar har doim ham sonli shaklda bo'lmaydi. Ma'lumotlar belgilar, so'zlar va matnli shaklda bo'lsa ularni qanday qayta ishlash lozim? Kompyuterlarni bizning tabiiy tilimizni qayta ishlashga qanday o`rgatish mumkin? Bugungi kunda Alexa, Google Home va boshqa ko'plab "aqlli" yordamchilar nutqimizni qanday tushunadi va javob beradi? Ushbu maqolada tabiiy tilni qayta ishlash deb nomlangan sun'iy intellekt sohasidagi Bag-of-words (BOW), CountVectorizer, TF-IDF, Co-Occurrence matrix, Word2Vec, CBOW, Skip-Gram, GloVe, ELMO va BERT kabi matnlarni qayta ishlash usullari orqali o`zbek tili korpusi matnlarini qayta ishlash usullari keltiriladi.

Tabiiy tilni qayta ishlash – bu sun'iy intellektning kichik sohasi bo'lib, u mashinalarga inson tilini tushunish va qayta ishlashga yordam beradi. Tabiiy tilni qayta ishlash (natural language processing, NLP) vazifalarining aksariyati uchun eng asosiy qadam tabiiy tildagi shablon (qolip)larni tushunish va dekodlash uchun so'zlarni raqamlarga aylantirishdir. NLPda bu bosqich **matnli ko'rinish** (**text representation**) deb yuritiladi [1,2,3].

Til korpusidagi qayta ishlanmagan matn **boshlang`ich qayta ishlanadi** va mashinali o'rganish modeli uchun mos formatda ko'rinishlariga keltiriladi. Ma'lumotlar *tokenizatsiya, nomuhim so'zlarini olib tashlash, tinish belgilarini olib tashlash, stemming, lemmatizatsiya* va boshqa bir qator boshlang`ich ishlov berish NLP vazifalari orqali qayta ishlanadi (1-rasm). Ushbu jarayonda ma'lumotlardagi mavjud "shovqin"lar tozalaanadi [4,5,6]. Ushbu tozalangan ma'lumotlar NLP ilovasiga va mashinali o'rganish modelining kiritish talablariga muvofiq turli shakl (shablon)larda taqdim etiladi. NLPda matnni qayta ishlash ishlatiladigan umumiy atamalar quyidagilar:

- **Korpus (Corpus, C):** ma'lumotlar to'plami yoki bir nechta matnli ma'lumotlar birgalikda korpus sifatida talqin qilinadi.
- **Lug'at (Vocabulary, V):** korpusdagi mavjud bo'lgan barcha unikal so'zlar to`plami.
- **Hujjat (Document, D):** ma'lumotlar to'plamining bitta matnli yozuvi.
- **Word(Word, W):** lug'atda mavjud so'zlar.



1-rasm. *Til korpusi matnlarini boshlang`ich qayta ishlash bosqichlari*[1]

Yuqoridagi 1-rasmda korpus matarini ML modeli uchun turli kiritish formatlariga aylantirish jarayoni keltirilgan. Chapdan boshlab, korpus tokenlarni, matn qurilish bloklari to‘plamini, ya’ni so‘zlar, belgilar va boshqalarni olishdan oldin bir necha bosqichlardan o‘tadi. ML modellari faqat sonli qiymatlarni qayta ishlashga asoslanganligi sababli, gapdagi tokenlar mos sonli qiymatlar bilan almashtiriladi. Keyingi qadamda, ular o'ng tomonda ko'rsatilgan turli xil kiritishli formatlarga aylantiriladi. Ushbu formatlarning har biri o'zining ijobiy va salbiy tomonlariga ega bo`lib, berilgan NLP vazifasining xususiyatlariga ko'ra strategik ravishda tanlanishi kerak.

Matnlarini qayta ishlash turlari

Matnlarini qayta ishlash jarayoni garchi iterativ bo'lsa ham, mashinali o'rganish modeli/algoritmi uchun muhim rol o'ynaydi. Matnli ko'rinishlarini ikki qismga ajratish mumkin [7,8]:

- *Matnni diskret ko'rinishlari (Discrete text representations);*
- *Taqsimlangan/uzluksiz matn ko'rinishlari (Distributed/Continuous text representations).*

Ushbu maqola diskret matn ko'rinishlariga etibor qaratiladi va Python tilidagi **Sklearn** paketi orqali matnlarni qayta ishlash usullari keltiriladi.

Matnni diskret ko'rinishlari

Korpus matnlarini diskret ko'rinishda tasvirlashda korpusdagi so'zlar bir-biridan mustaqil tarzda ifodalanadi. Ushbu yondashuvda so'zlar korpus(lar) lug'atidagi o'rniga mos keladigan indekslar bilan ifodalanadi. Ushbu toifaga kiruvchi metodlar quyida keltirilgan [1,3,7]:

- *One-Hot encoding;*
- *Bag-of-words (BOW);*
- *CountVectorizer;*
- *TF-IDF*
- *Ngram.*

One-Hot encoding usuli

One-Hot encoding usulida korpusdagi har bir so'zga 0 va 1 dan iborat vektor mos qo`yiladi [9]. Bu usuldagi kodlashtirishda vektorning faqat bitta elementiga – **1**, qolgan barcha elementlariga – **0** belgilanadi. Bu qiymat element toifasini ifodalaydi. Hosil qilingan raqamli vektorlar NLPda **hot vector** deb ataladi va korpusdagi har bir so'zga unikal **hot vector** moslashtiriladi. Usbu amal mashinali o'rganish modeliga har bir so'zni vektori orqali alohida tanib olish imkonini beradi. One-Hot encoding usuli ma'lumotlar to'plamida kategorial xususiyat mavjud bo`lgan hol uchun foydali bo'lishi mumkin. Masalan:

[1] https://towardsdatascience.com/an-overview-for-text-representations-in-nlp-311253730af1

"Men itimni yaxshi ko'raman" degan gapga mos vektor qiymatlari gapdagi har bir so'zga mos tarzda quyidagicha ifodalanadi:

Men → [1 0 0 0], itimni → [0 1 0 0], yaxshi → [0 0 1 0], ko'raman → [0 0 0 1]

yoki,

$$\begin{matrix} \boldsymbol{Men}: \\ \boldsymbol{itimni}: \\ \boldsymbol{yaxshi}: \\ \boldsymbol{ko'raman}: \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ushbu holda gap quyidagicha raqamli shaklda ifodalanadi:

sentence = [[1,0,0,0],[0,1,0,0],

[0,0,1,0],[0,0,0,1]]

One-Hot encoding usulida har bir bit mumkin bo'lgan toifani ifodalaydi va ma'lum bir o'zgaruvchi bir nechta toifalarga kirmasa, uni ifodalash uchun bitta bit yetarli bo`ladi. Ushbu usul orqali "Men" va "men" so'zlariga bir-biridan farq qiluvchi vektorlar mos qo`yiladi. Matnni qayta ishlash jarayonida barcha so'zlarga kichik harflar qo'llash orqali bosma va kichik harflarga bir xil vektorni mos qo`yish mumkin. Ushbu usulda bir o'lchovli vektorning o'lchami lug'at hajmiga teng bo'ladi.

One-Hot encoding usuli orqali korpus kodlashtirilganida lug'atdagi har bir *so'z* yoki *token* **raqamli vektor**ga aylanadi. Demak, korpusdagi gaplar, o'z navbatida, **(p,q)** o'lchamdagi matritsaga aylanadi. Bunda,

- **"p"** - *gapdagi tokenlar soni;*
- **"q"** - *lug'at hajmi.*

One-Hot encoding usulida so`zga mos qo`yilgan raqamli vektorning o'lchami korpusning lug'at hajmiga to'g'ridan-to'g'ri proportsionaldir. Demak, korpus hajmining oshishi bilan vektorning o'lchovi ham ortadi. **100000** tagacha yoki undan ham ko'proq unikal so'zlarni o'z ichiga olishi mumkin bo'lgan katta hajmdagi korpus uchun ushbu usuldan foydalib bo'lmaydi. One-Hot encoding usulini Sklearn paketi yordamida amalga oshiramiz:

```
1. from sklearn.preprocessing import OneHotEncoder
2. import itertools
3. # 4 ta namunaviy hujjat
4. docs = ['Men NLP bilan ishlayman', 'NLP juda ajoyib texnologiya',
5. 'Tabiiy tilni qayta ishlash', 'Zamonaviy texnologiyalar bilan ishlash']
6. # hujjatlarni tokenlarga ajratish
7. tokens_docs = [doc.split(" ") for doc in docs]
8. # tokenlar ro‘yxatini umumlashtirish va so'zni identifikatoriga moslashtiradigan lug'atni yaratish
9. all_tokens = itertools.chain.from_iterable(tokens_docs)
10. word_to_id = {token: idx for idx, token in enumerate(set(all_tokens))}
11. # tokenlar ro‘yxatini token-id ro‘yxatlariga aylantirish
12. token_ids = [[word_to_id[token] for token in tokens_doc] for tokens_doc in tokens_docs]
13. # token-id ro'yxatlarini umumlashtirish
14. vec = OneHotEncoder(categories="auto")
15. X = vec.fit_transform(token_ids)
16. print(X.toarray())
17. _____
```

[[0. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0.]

[0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1.]

[1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0.]

[0. 1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 1. 0. 0.]]

Afzalliklari	Kamchiliklari
Tushunish va amalga oshirish oson	agar toifalar soni juda ko'p bo'lsa, katta hajmli xotira zarur
	so'zlarning vektor ko'rinishi ortogonal bo'lib, turli so'zlar orasidagi munosabatni aniqlab bo`lmaydi
	gapdagi so'zning ahamiyatini aniqlab bo'lmaydi
	yuqori o'lchamli siyrak matritsani ifodalash uchun katta sondagi hisoblashlar talab etiladi

Bag-of-words usuli

Bag-of-words usulida korpusdagi **"so'zlar sumkaga"** joylashtiriladi va har bir so'zning takrorlanish chastotasini hisoblab chiqiladi. Ushbu usulda matnni ifodalash uchun *so'z tartibi* yoki *leksik ma'lumoti* hisobga olinmaydi. BOW usuliga asoslangan algoritmlarda o'xshash so'zlarga ega hujjatlar so'zning joylashuvidan qat'iy nazar o'xshash kabi natija qaytariladi.

BOW usuli orqali matnning fragmenti **fiksirlangan uzunlikdagi vektorlarga** aylantiradi. So'zlarning chastotasi aniqlash *hujjatlarni o`zaro solishtirishga* yordam beradi. BOW usulidan *tematik modellashtirish, hujjatlarni tasniflash* va *elektron pochtada spam xabarlarini aniqlash* kabi turli xil NLP ilovalarida foydalanish mumkin. Quyida o`zbek tilida berilgan 2 ta gapga mos BoW vektori keltirilgan.

1-gap: *"Adirlar ham bahorda lola bilan go'zal, chunki lola – bahorning erka guli".*

2-gap: *"Lola ham shifokorlik kasbini tanladi".*

	Adirlar	bahorda	lola	go'zal	bahorning	erka	guli	shifokorlik	kasbini	tanladi
1-gap	1	1	2	1	1	1	1	0	0	0
2-gap	0	0	1	0	0	0	0	1	1	1

B.Elov, N.Xudayberganov va Z.Xusainovalar tomonidan yozilgan "Tabiiy tilni qayta ishlashda bag of words algoritmidan foydalanish" ilmiy maqolada o`zbek tilidagi matnlarni BoW alrgoritmi vositasida raqamli shaklga o`tkazish usullari keltirilgan [10].

CountVectorizer usuli

CountVectorizer usuli hujjatda so'zning paydo bo'lish chastotasini hisoblab chiqishga asoslangan. Ushbu usul orqali korpusdagi bir nechta gaplar asosida so'zlar matritsasi aniqlanadi va u gapdagi har bir so'zning chastotasi bilan to'ldiriladi [10]. CountVectorizer usulini Sklearn paketi yordamida amalga oshiramiz:

```
1. from sklearn.feature_extraction.text import CountVectorizer
2. text = ["Men NLP bilan ishlayman. NLP juda ajoyib."]
3. vectorizer = CountVectorizer()
4. # Tokenizatsiyalash va lug`atni yaratish
5. vectorizer.fit(text)
6. print(vectorizer.vocabulary_)
7. # hujjatni kodlash
8. vector = vectorizer.transform(text)
9. # kodlangan vektorni umumlashtirish
10. print(vector.shape)
11. print(vector.toarray())
12. _____
{'men': 4, 'nlp': 5, 'bilan': 1, 'ishlayman': 2,
'juda': 3, 'ajoyib': 0}

(1, 6)

[[1 1 1 1 1 2]]
```

Ko'rib turganingizdek, "NLP" so'zi matnda ikki marta uchraydi. Ushbu usulda gapdagi so'zning "og'irligi" uning chastotasiga teng. Ushbu og'irliklar har xil turdagi tahlillar uchun va ML modellarini o'rgatish uchun ishlatilishi mumkin. Zarur natijalarni olish uchun CountVectorizerning o'zgartirilishi mumkin bo'lgan *lowercase, strp_accents, preprocessor* kabi parametrlari mavjud.

Afzalliklari	Kamchiliklari
matndagi so'zlarning chastotasini aniqlash imkonini beradi	so'zning joylashuv ma'lumotlariga e'tibor bermaydi. Natijadan so'zning ma'nosini tushunish mumkin emas.
kodlangan vektorning uzunligi lug'atning uzunligiga teng	Yuqori chastotali so'zlar matn uchun muhimroq ahamiyatga ega ma'lumot beradi degan xato xulosani beradi. "bilan, va, ammo,…" kabi nomuhim so'zlarni bunga misol sifatida keltirish mumkin.
	Bu usul so'zning gapdagi pozitsion ma'lumotlarini yo'qotadi.

TF-IDF usuli

Yuqori chastotali so'zlarni aniqlash va past chastotali so'zlarni e'tiborsiz qoldirish uchun so'zlarning "vaznlarini" mos ravishda normallashtirish kerak. TF-IDF usuli orqali ushbu vazifani amalga oshirish mumkin. TF-IDF qiymatni 2 ta faktor vositasida hisoblash mumkin [11,12]:

$$TF - IDF = TF(w,d) * IDF(w)$$

Bu yerda, $\boldsymbol{TF(w,d)}$ – "**d**" hujjatidagi "**w**" so'zining chastotasi.

$\boldsymbol{IDF(w)}$ qiymatni quyidagicha hisonlash mumkin:

$$IDF(w) = log\ (\frac{N}{df(w)})$$

Bu yerda, **N** – hujjatlarning umumiy soni va $\boldsymbol{df(w)}$ - "**w**" so'zini o'z ichiga olgan hujjatlarning chastotasi.

TF-IDF usuli orqali aniqlangan qiymatlar har bir so'zning og'irligini nafaqat so'zlarning chastotasiga, balki ushbu so'zning butun korpusda qanchalik ko`p uchrashiga ham bog'liq.

TF-IDF qiymatni hisoblash uchun yuqorida muhokama qilingan CountVectorizer qiymatga IDF ballni ko'paytirish lozim. Hosil qilingan natijadan korpusda ko`p uchraydigan so'zlar (masalan, nomuhim so'zlari) uchun qiymatlarni nisbatan kattaroq va juda past chastotali so'zlar ("shovqinli" so`zlar) uchun past ekanligini qayd etish mumkin. TF-IDF usulini Sklearn paketi yordamida amalga oshiramiz:

```
1. from sklearn.feature_extraction.text import TfidfVectorizer
2. text1 = ['Men NLP bilan ishlayman', 'NLP juda ajoyib',
3. 'NLP - bu mashinalarga tabiiy tilni qayta ishlashga imkon berishdir',
4. 'bu misol nlp texnikasiga namuna']
5. tf = TfidfVectorizer()
6. txt_fitted = tf.fit(text1)
7. txt_transformed = txt_fitted.transform(text1)
8. idf = tf.idf_
9. print(dict(zip(txt_fitted.get_feature_names_out(), idf)))
10. _________
```

{'ajoyib': 1.916290731874155, 'berishdir': 1.916290731874155, 'bilan': 1.916290731874155, 'bu': 1.5108256237659907, 'imkon': 1.916290731874155, 'ishlashga': 1.916290731874155, 'ishlayman': 1.916290731874155, 'juda': 1.916290731874155, 'mashinalarga': 1.916290731874155, 'men': 1.916290731874155, 'misol': 1.916290731874155, 'namuna': 1.916290731874155, 'nlp': 1.0, 'qayta': 1.916290731874155, 'tabiiy': 1.916290731874155, 'texnikasiga': 1.916290731874155, 'tilni': 1.916290731874155}

Natijadagi **"NLP"** so'zining vazniga e'tibor bering. U barcha gaplarda mavjud bo'lganligi sababli, unga **1.0** past og'irlik vazni berilgan. Xuddi shunday, **"va"** nomuhim so'ziga ham nisbatan past og'irlik **1.22** berilgan, chunki u berilgan **4** ta sidan **3** tasida mavjud.

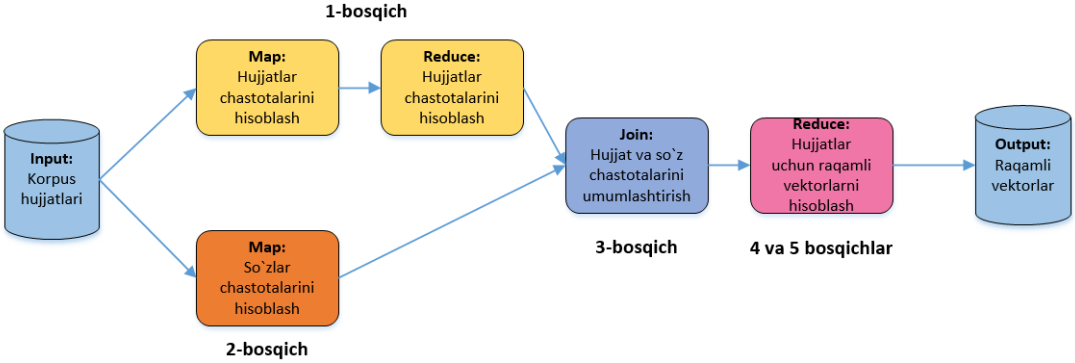
TF-IDF usuli CountVectorizer usuliga o'xshab, zarur natijalarga erishish uchun o'zgartirilishi mumkin bo'lgan turli xil parametrlarga ega. Ba'zi muhim parametrlarga *lowercase*, *strip_accent*, *stop_words*, *max_df*, *min_df*, *norma*, *ngram_range* va *sublinear_tf*[2] kabilarni misol sifatida keltirish mumkin. Ushbu parametrlarning chiqish vazniga ta'siri ushbu maqola doirasida ko'rib chiqilmaydi.

Afzalliklari	Kamchiliklari
oddiy, tushunarli va oson amalga oshirish mumkin	so'zning pozitsion ma'lumotlari saqlanmaydi
Korpusda ko`p uchraydigan so'zlar va past chastotali so`zlarni aniqlash mumkin.	TF-IDF korpusga juda bog'liq. Yuqori sifatli o'quv ma'lumotlariga ega bo'lish talab etiladi.
	So`zlarning semantik xususiyati qayd etilmaydi.

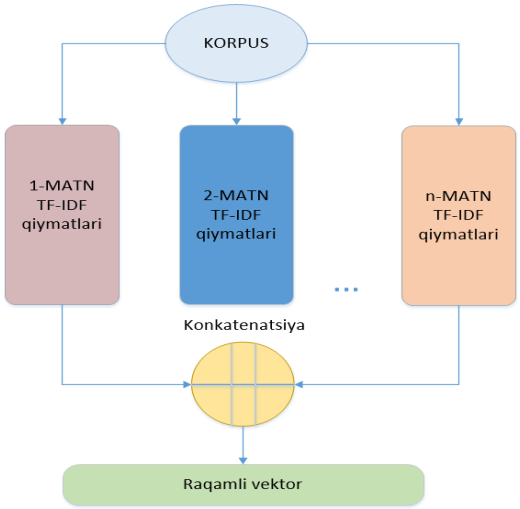
B.Elov, Z.Xusainova va N.Xudayberganovlar tomonidan yozilgan

[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

"O`zbek tili korpusi matnlari uchun TF-IDF statistik ko`rsatkichni hisoblash" ilmiy maqolada o'zbek tili korpusidagi hujjatlarni TF-IDF usulidan foydalanib, kalit so'zga mos tarzda tartiblash jarayonini ko'rib chiqilgan [11] va TF-IDF qiymatni hisoblashning 5 ta bosqichlari keltirilgan:



2-rasm. *TF-IDF qiymatni hisoblash bosqichlari*



3-rasm. *TF-IDF qiymatni hisoblash*

Mualliflar tomonidan amalga oshirilgan hisoblashlar va tahlillar asosida bir qator ilmiy xulosalar va takliflar berilgan. Jumladan, katta hajmdagi til korpuslarida amalga oshiriladigan so'rovlarga mos hujjatlarni aniqlashda TF-IDF usulidan foydalanishlari samarali ekanligi qayd etilgan.

Ngram usuli

Ngram usuli BoW modeliga o'xshash bo`lib, yagona farqi bir so'zning chastotasini hisoblash o'rniga, korpusda birga uchraydigan so'zlar guruhlari (ikki yoki undan ortiq guruhlarda) chastotasi hisoblanadi[13]. Ushbu usul orqali matndagi birlashtirilgan so'zlar soniga qarab, model *bigram* (2 ta so'z), *trigram* (3 ta so'z) kabi ataladi.

```
1. text = ['Men NLP bilan ishlayman', 'NLP juda ajoyib', 'NLP juda qiyin', 'NLP keng ommabop']
2. from sklearn.feature_extraction.text import CountVectorizer
3. cv = CountVectorizer(ngram_range=(2,2))
4. bow = cv.fit_transform(text)
5. print(cv.vocabulary_)
6. print(bow[0].toarray())
7. _____
```

{'men nlp': 4, 'nlp bilan': 5, 'bilan ishlayman': 0, 'nlp juda': 6, 'juda ajoyib': 1, 'juda qiyin': 2, 'nlp keng': 7, 'keng ommabop': 3}

[[1 0 0 0 1 1 0 0]]

Afzalliklari	Kamchiliklari
Ngramlar usuli gapning semantik ma'nosini qamrab oladi va so'zlar orasidagi munosabatni topishga yordam beradi.	Lug'atdan tashqari (OOV) so`zlar qayta ishlanmaydi. Agar so`zlar lug'atda mavjud bo'lmasa, so'zlar yoki ularning semantik ma'nolari o'rtasidagi munosabatlar aniqlanmaydi.
Amalga oshirish oson.	

Taqsimlangan/uzluksiz matn ko'rinishlari

Taqsimlangan matn ko'rinishi – bu so'zning sonli ko'rinishi boshqa so'zdan mustaqil yoki bir-birini istisno qilmasa va ularning konfiguratsiyasi ko'pincha ma'lumotlardagi turli ko'rsatkichlar va tushunchalarni ifodalaydi. Bu

holda so'z haqidagi ma'lumot unga mos vektor bo'ylab taqsimlanadi. Taqsimlangan matn ko'rinishida har bir so'z noyob va bir-biridan mustaqil deb hisoblangan *diskret ko'rinishdan* farq qiladi.

Bugungi kunda ko`p ishlatiladigan taqsimlangan matn ko'rinishlari quyidagilar [14,15,16]:

- *Co-Occurrence matrix;*
- *Word2Vec;*
- *GloVe.*

Co-Occurrence matrix usuli

Co-Occurrence matritsasi usulida bir-biriga yaqin joylashgan ob'ektlarning birgalikda paydo bo'lishini hisobga oladi. Amaldagi ob'ekt **bitta so'z**, **bigramm** (n=2) yoki **ibora** bo'lishi mumkin [15,17]. Berilgan korpusga mos matritsa qiymatlarini hisoblash uchun asosan bitta so'z ishlatiladi. Bu bizga korpusdagi turli so'zlar orasidagi bog'lanishni tushunishga yordam beradi. Yuqorida muhokama qilingan CountVectorizer usulida keltirilgan misolni ko'rib chiqamiz va uni uzluksiz shaklga aylantiramiz:

```
1. from sklearn.feature_extraction.text import CountVectorizer
2. import pandas as pd
3. docs = ['Men NLP bilan ishlayman', 'NLP juda ajoyib',
4. 'NLP - bu mashinalarga tabiiy tilni qayta ishlashga imkon berishdir',
5. 'bu misol nlp texnikasiga namuna']
6. # Nomuhim so'zlarni o'chirish
7. uz_stop_words=open("uz_stop_words.txt",encoding="utf-8").read().split('\n')
8. count_vectorizer = CountVectorizer(stop_words = uz_stop_words, token_pattern='[a-zA-Z0-9’‘\-]{1,}')
9. vectorized_matrix = count_vectorizer.fit_transform(docs)
10. co_occurrence_matrix = (vectorized_matrix.T * vectorized_matrix)
11. print(pd.DataFrame(co_occurrence_matrix.A,
i.  columns=count_vectorizer.get_feature_names_out(),
ii. index=count_vectorizer.get_feature_names_out()))
12. _____
```

	ajoyib	*berishdir*	*imkon*	*ishlashga*	*ishlayman*	*mashinalarga*
ajoyib	1	0	0	0	0	0
berishdir	0	1	1	1	0	1
imkon	0	1	1	1	0	1
ishlashga	0	1	1	1	0	1
ishlayman	0	0	0	0	1	0
mashinalarga	0	1	1	1	0	1
misol	0	0	0	0	0	0
namuna	0	0	0	0	0	0
nlp	1	1	1	1	1	1
tabiiy	0	1	1	1	0	1
texnikasiga	0	0	0	0	0	0
tilni	0	1	1	1	0	1

Har bir so'zning ko'rinishi uning bog`liqlik matritsadagi mos keladigan satri (yoki ustuni) hisoblanadi.

Afzalliklari	**Kamchiliklari**
so'zlarni o`zaro bog`lanishini soddaroq ifodalash	CountVectorizer va TF-IDF matritsalariga o'xshash tarzda matritsa hosil qilinadi
diskret usullardan farqli o'laroq, gapdagi so'zlarning tartibini saqlash	matritsa o'lchami lug'at hajmiga bog`liq
so`zlarni o`zaro bog`lanishini butun korpusdan aniqlash	Ushbu usul yordamida barcha so'z birikmalarini aniqlan bo`lmaydi.

Word2Vec usuli

Word2Vec – bu so'zlarni joylashtirish uchun mashhur algoritm. Ushbu algortim 2013-yilda Tomas Mikalov tomonidan "Vektor fazosida so'zlarning ifodalanishini samarali baholash" tadqiqoti ostida ishlab chiqilgan [18,19]. Metod so'zlarni ifodalashning bashorat qilishga asoslangan.

**So'zlar bog`liqligi (Word embeddings)** – bu so'zning vektor ko'rinishi bo`lib, har bir so'zning boshqa so'zlar bilan semantik va sintaktik aloqasini hisobga olgan holda, belgilangan vektor o'lchami bilan ifodalanadi. Word2vec arxitekturasi bitta yashirin qatlamli tarmoqdir. Yashirin qatlamning og'irligi *so'zning yo'qotish funktsiyasi (normal backprop)* orqali aniqlanadi.

Ushbu arxitektura avtokoderga o'xshab, bu erda sizda kodlovchi qatlami va dekoder qatlami mavjud va o'rta qism o'lchamlarni kamaytirish yoki anomaliyalarni aniqlashda foydalanish uchun ishlatilishi mumkin bo'lgan kirishning siqilgan ko'rinishidir. **Word2vec** usuli orqali korpus tasviri 2 xil usulda amalga oshiriladi[20,21]:

- **CBOW** – atrofdagi so'z konteksti asosida oraliq so'zni taxmin qilishga asoslangan. CBOW usulida kontekst/atrofdagi so'zlarni hisobga olgan holda qaysi so'z ko'proq mos kelishi haqida bo'sh joylarni to'ldirishga harakat qilinadi. Ushbu usul kichikroq ma'lumotlar to'plamlari bilan samarali natijani beradi.
- **Skip-Gram** – maqsadli so'zdan (CBOWga teskari) atrofdagi kontekst so'zlarini taxmin qilishga harakat qiladi. Kattaroq hajmdagi ma'lumotlar to'plamida yaxshiroq natija beradi. Biroq o`quv ma`lumotlar top`lamini qayta ishlashga ko`p vaqt sarflanadi.

Word2vec usuli orqali vektor arifmetikasidan foydalangan holda so'zlar o'rtasidagi o'xshashlik darajasi aniqlanadi. **"Man is to woman as king is to queen"** kabi shablondan arifmetik amallar orqali **"king" - "man" + "woman"** = **"queen"** kabi natijaga ega bo`lish mumkin. Shuningdek, **"queen"** so`zi hozirgi va o'tgan zamon kabi sintaktik va semantik munosabatlarni ifodalaydi. **Gensim** paketi yordamida word2vec usulini ko'rib chiqamiz:

```
from gensim.models import Word2Vec
sentences = ['Men NLP bilan ishlayman', 'NLP juda ajoyib',
'NLP - bu mashinalarga tabiiy tilni qayta ishlashga imkon berishdir',
'bu misol nlp texnikasiga namuna']
# gapni oldindan qayta ishlash Word2Vec uchun zarur formatga aylantirish
sentence_list=[]
for i in sentences:
li = list(i.split(" "))
sentence_list.append(li)
model = Word2Vec(sentence_list, min_count=1,
workers=4, sg=1, window=4)
model.wv['nlp']
model.wv.most_similar(positive=['nlp'])
_________
```

[('imkon', 0.24666069447994232),

('Men', 0.11936754733324051),

('ajoyib', 0.11928389966487885),

('ishlashga', 0.11663015931844711),

('texnikasiga', 0.09614861011505127),

('bu', 0.08543577790260315),

('ishlayman', 0.07172605395317078),

('tilni', 0.05970853567123413),

('mashinalarga', 0.04119439423084259),

('-', 0.012471411377191544)]

Yuqoridagi dastur kodining bir necha satrida biz nafaqat so'zlarni vektor sifatida o'rgatish va ko'rsatish imkoniyatiga egamiz, balki o'xshash va turli so'zlarni aniqlashimiz mumkin. Vektorlar o'rtasidagi o'xshashlikni aniqlashning ikki yo'li mavjud:

- ***Normallashtirilgan:*** *vektorlar orasidagi skalyar ko`paytmani hisoblab, ularni o'xshashligini aniqlash mumkin;*
- ***Normallashtirilmagan:*** *vektorlar orasidagi kosinus o'xshashligini quyidagi formuladan foydalanib hisoblash mumkin:*
- $cosine\ similarity = 1 - cosine\ distance = \frac{u*v}{||u||*||v||}$

Korpus asosida raqamli vektorlarni aniqlash, korpusni mashinali o'qitish va so'zlar o'rtasidagi munosabatlarni aniqlash algoritmlari keying ilmiy nashrlarda ko`rib chiqiladi. Word2Vec usulining afzallikalri va kamchiliklari quyidagi jadvalda keltirilgan:

Afzalliklari	Kamchiliklari
Turli so'zlar o'rtasidagi sintaktik va semantik munosabatlarni aniqlashga imkon beradi	Lug'atdan tashqari so'zlar (OOV)ni qayta ishlab bo`lmaydi.
So`zga mos raqamli vektorining o'lchami kichik va moslashuvchan.	So'zning semantik ifodasi faqat qo'shnilariga asoslanadi.
Korpusni o`qiitsh jarayoni inson omiliga bog`liq emas.	Word2Vec usulini yangi tabiiy tilga qo`llash uchun juda ko`p amallarni bajarish lozim
	Ma`lumotlar aniqligi osjishi uchun kattaroq korpus talab qilinadi.

GloVe usuli

Global Vectors yoki qisqacha GloVe usuli so'zlarni raqamli ifodalashning zamonaviy NLP usulidir. Ushbu usul Jefferi Pennington, Richard Socher va Kristofer Manning tomonidan 2014 yilda ishlab chiqilgan va joriy etilgan [22]. Yuqorida keltirilgan word2vec usulidan farqli ravishda ushbu usulda so'zning lokal va global statistikasi o'rganiladi va so'zlarni ifodalash uchun **gibrid yondashuv** deb ataladi. GloVe usulida quyidagi belgilashlardan foydalaniladi:

$$v_i^T v_j = logP(i|j)$$

yoki,

$$v_i^T v_j = logP(X_{ij}) - logP(X_i)$$

Shunday qilib, **P(i|j)** ga mos tarzda **Vi** va **Vj** so'z vektorlar qiymatlari hisonlanadi. Ushbu vektorlar birgalikda joylashish matritsadagi global statistik ma'lumotni ifodalaydi. GloVe usulidagi maqsad funksiyasi haqida keyingi ilmiy maqolalarda ma`lumot keltiriladi. Kata hajmdagi matnlarni oldindan o'rgatilgan modellardan asosida GloVe vektorlarini shakllantirish va ulardan foydalanish quyida keltirilgan:

```
import gensim.downloader as api
# 2 milliard tvitning 25 o'lchamli GloVe tasvirini yuklab olish
twitter_glove = api.load("glove-twitter-25")
# O'xshash so'zlarni topish
print(twitter_glove.most_similar("book",topn=10))
# 25D vektorlarni olish
print(twitter_glove['book'])
print(twitter_glove.similarity("book", "school"))
```

```
_____
[('books', 0.94181889295578), ('project', 0.9214614033699036), ('review', 0.9140495657920837), ('script', 0.9069417119026184), ('new', 0.9069172143936157), ('feature', 0.8995184302330017), ('guest', 0.897861659526825), ('read', 0.8931056261062622), ('post', 0.8916701674461365), ('art', 0.8880472183227539)]

[ 0.21621   0.056781  0.82955  -0.1424    0.82832
 -0.87341   1.699

 -0.25702   0.65303  -0.82435   0.26496   0.4612
 -4.0463   -0.044556

  0.15648  -0.083655  0.72399   0.20802  -
  0.27561  -0.024987 -0.83992

 -0.92536  -0.95454   0.42348  -0.14709 ]

0.7545484
```

GloVe usulining afzallikalri va kamchiliklari quyidagi jadvalda keltirilgan:

Afzalliklari	Kamchiliklari
Word2vec usuliga qaraganda yaxshiroq ishlaydi	Birgalikda yuzaga keladigan matritsa va global ma'lumotdan foydalanganligi sababli, GloVe usulida word2vec usuliga qaraganda ancha ko`p xotira talab etiladi
Vektorlarni qurishda so'z juftligi va so'z juftligi munosabatini inobatga oladi	Word2vec usuliga o'xshab, u ko'p ma'noli so'zlar muammosini hal qilmaydi
Word2Vec bilan solishtirganda GloVe usulini	

parallellashtirish	
osonroq, shuning	
uchun o`rgatish vaqti	
qisqaroq	

Zamonaviy yondashuvlar

ELMO usuli

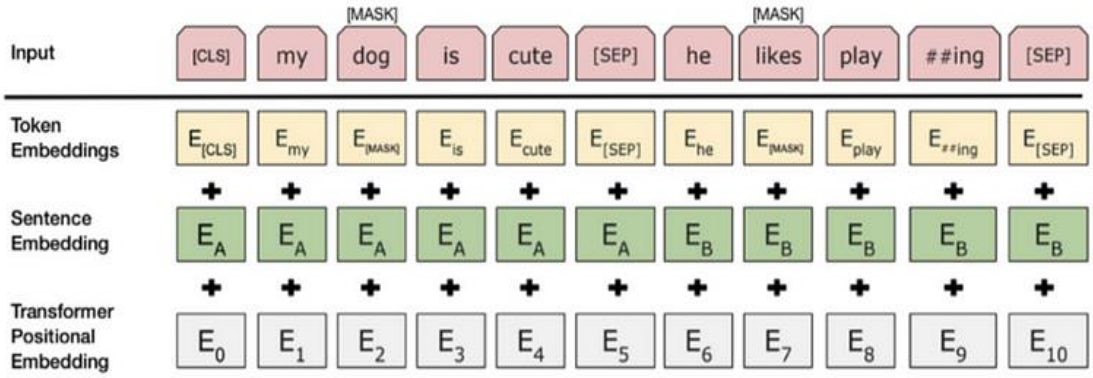
2018 yil mart oyida Metyu Peters va boshqalar chuqur kontekstli so‘z ko‘rinishlari nomidagi maqolasi taqdim etildi [23]. U taklif qilgan usulda word2vec va GloVe usullari kamchiliklarini vektor ko'rinishi va u ifodalovchi so'z o'rtasida ko'pdan-bir munosabatga ega bo'lish orqali hal qilishga harakat qilinadi. ELMO usulida kontekstni inobatga olinib, so'zning vektor ko'rinishini mos ravishda o'zgartiriladi.

ELMO usulida so'zlarni boshlang`ich so'z vektorlariga aylantirish uchun belgilar darajasidagi **CNN**lardan foydalaniladi. O'qitish jarayonida qo'shimcha ravishda ikki tomonlama **LSTM**lardan foydalaniladi. Usul orqali yo'naltirish va orqaga iteratsiya kombinatsiyasi mos ravishda so'zdan oldin va keyingi kontekst ma'lumotlarini ifodalovchi oraliq so'z vektorlarini yaratiladi. Boshlangi`ch so'z vektori va 2 ta oraliq so'z vektorining vaznli yig'indisi yakuniy qiymatni beradi.

BERT usuli

BERT – 2019 yildagi **Google AI** jamoasining “Pre-training of Deep Bidirectional Transformers for Language Understanding” nomli maqolasida tavsiflangan tilni tushunish uchun chuqur ikki yo‘nalishli **transformerlar (Transformers)**ni oldindan o‘rgatish usuli hisoblanadi [24].

Bu usulda transformerlarni oldindan o‘rgatish uchun yangi o‘z-o‘zini nazorat qiladigan mashinali o‘rganish vazifasidir.



4-rasm. *BERT usuli arxitekturasi.*

BERT usulida til modelining ikki tomonlama kontekstidan foydalanadi. Bunda bashorat qilish vazifalari uchun ishlatiladigan oraliq tokenlarni yaratish uchun *chapdan o'ngga* va *o'ngdan chapga* “niqoblash”ga harakat qilinadi.

BERT modeliga kiritilgan ma'lumotlar tokenlarni joylashtirish, segmentatsiyalashdan iborat bo'lib, kontekstda so'zni to'g'ri bashorat qilish uchun model uchun niqoblash strategiyasiga amal qiladi. BERT usuli orqali so'zlar orasidagi kontekstual munosabatni o'rganadigan va NER va savol-javob tizimlari kabi boshqa vazifalarni bajarish uchun sozlangan transformer tarmog'idan foydalanadi.

Matnni sonli ko`rinishi ilovalari

Ushbu maqolda keltirilgan matnning sonli ko'rinishdagi modellarini quyidagi NLP vazifalariga qo`llash mumkin:

- **Matn tasnifi (Text Classification):** Matnni tasniflash vazifasida matnga boshlang`ich ishlov berish uchun matnni vektor ko'rinishida shakllantirish muhimdir.
- **Mavzuni modellashtirish (Topic Modelling):** Mavzuni modellashtirish vazifasida matnni turli mavzularda modellashtirish uchun to'g'ri formatda taqdim etilishi talab qilinadi.
- **Avtomatik tuzatish modeli (Autocorrect Model):** Avtomatik tuzatish modeli orqali matndagi imlo xatolari tuzatiladi. Avtomatik

tuzatish modeli vositasida berilgan matnni talab qilingan sonli formatda taqdim etish kerak.

- **Yangi matn generatsiya qilish (Text Generation):** Matnni generatsiya qilish uchun ehtimollarga asoslangan sonli matn formati talab qilinadi.

Mashinali o'rganish modelini o'rgatishdan oldin matnni ma'lum bir formatda ifodalash juda muhimdir. Format qanchalik murakkab bo'lsa, modelning aniqligi va natijalari shunchalik yaxshi bo'ladi. Matnli ma'lumotlarini o'z ichiga olgan har bir NLP ilovasi yaxshi matn ko'rinishini talab qiladi.

Xulosa

Matnni diskret ko'rinishlari usullari orqali korpusdagi har bir so'z unikal deb hisoblanadi va yuqorida muhokama qilgan turli usullarga asoslanib, sonli shaklga aylantiriladi. Maqolada keltirilgan turli xil usullarning bir-biriga o'xshash bir nechta afzalliklari va kamchiliklari keltirildi. Ularni bir butun sifatida umumlashtiramiz. Matnni diskret ko'rinishdagi sonli qiymatlarini hosil qiluvchi usullarni *tushunish*, *amalga oshirish* va *talqin qilish* oson. TF-IDF kabi algoritmlar oddiy va nomuhim so'zlarni filtrlash uchun ishlatilishi mumkin. Bu esa modelni o'rgatish va tezroq umumlashtirishga yordam beradi. Usullarni kamchiliklari sifatida hosil qilinadigan lug'atning korpus hajmiga to'g'ridan-to'g'ri proportsionalligini keltirish mumkin. Katta hajmli lug'at turli xildagi xotira cheklovlariga olib kelishi mumkin. Barcha usullarda korpusdagi so'zlarni bir-biridan mustaqil deb hisoblanadi. Bu esa nol bo'lmagan qiymatlarga ega bo'lgan juda siyrak vektorlarni hosil qilinishiga olib keladi. Hosil qilingan vektorlar so'zning kontekstini yoki semantikasini ifodalamaydi. Matnning diskret ko'rinishlari mashinali o'rganishning klassik usullari va chuqur o'rganish dasturlarida *hujjatlar o'xshashligi, hissiyotlarni tasniflash, spam tasnifi* va *mavzularni modellashtirish* kabi NLP vazifalarini hal qilish uchun keng qo'llaniladi.

NLPdagi murakkab vazifalarnini *taqsimlangan matn ko'rinishlari algoritmlari* vositasida hal qilish mumkin. Taqsimlangan matn ko'rinishlaridan til korpusni tushunish va o'rganishda foydalanish mumkin. Korpus ichidagi so'zlarni va ularning bir-biri bilan qanday bog'lanishini o'rganishni bunga misol sifatida keltirish mumkin. Bugungi kunda *Savol-javob tizimlari, hujjatlar tasnifi, chatbot, NER ob'ektni tanib olish* kabi murakkab NLP vazifalarini hal qilish uchun nazorat ostidagi o'rganish modellarini ishlab chiqishda taqsimlangan matn ko'rinishlaridan keng miqyosida foydalanilmoqda. Hozirgi vaqtda ushbu usullar CNN va LSTMlarga asoslangan zamonaviy NLP ilovalarini ishlab chiqishda qo`llanilmoqda.

Foydalanilgan adabiyotlar ro`yxati

1. Naseem, U., Razzak, I., Khan, S. K., & Prasad, M. (2021). A Comprehensive Survey on Word Representation Models: From Classical to State-of-the-Art Word Representation Language Models. *ACM Transactions on Asian and Low-Resource Language Information Processing*, *20*(5). https://doi.org/10.1145/3434237
1. Chai, C. P. (2023). Comparison of text preprocessing methods. *Natural Language Engineering*, *29*(3). https://doi.org/10.1017/S1351324922000213
2. Probierz, B., Hrabia, A., & Kozak, J. (2023). A New Method for Graph-Based Representation of Text in Natural Language Processing. *Electronics*, *12*(13). https://doi.org/10.3390/electronics12132846
3. B.ELov, E.Adalı, Sh.Khamroeva, O.Abdullayeva, Z.Xusainova, N.Xudayberganov (2023). The Problem of Pos Tagging and Stemming for Agglutinative Languages. *8 th International Conference on Computer Science and Engineering UBMK 2023, Mehmet Akif Ersoy University, Burdur – Turkey.*
4. B.ELov, Sh.Khamroeva, Z.Xusainova (2023). The pipeline processing of NLP. *E3S Web of Conferences 413, 03011, INTERAGROMASH 2023.* https://doi.org/10.1051/e3sconf/202341303011
5. B.Elov, Sh.Hamroyeva, X.Axmedova. Methods for creating a morphological analyzer. *14th International Conference on Intellegent Human Computer Interaction, IHCI 2022, 19-23 October 2022, Tashkent.* https://dx.doi.org/10.1007/978-3-031-27199-1_4

6. Siebers, P., Janiesch, C., & Zschech, P. (2022). A Survey of Text Representation Methods and Their Genealogy. *IEEE Access*, *10*. https://doi.org/10.1109/ACCESS.2022.3205719
7. Jiang, Z., Gao, S., & Chen, L. (2020). Study on text representation method based on deep learning and topic information. *Computing*, *102*(3). https://doi.org/10.1007/s00607-019-00755-y
8. Rodríguez, P., Bautista, M. A., Gonzàlez, J., & Escalera, S. (2018). Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing*, *75*. https://doi.org/10.1016/j.imavis.2018.04.004
9. B.Elov, Z.Xusainova, N.Xudayberganov. Tabiiy tilni qayta ishlashda Bag of Words algoritmidan foydalanish. *O`zbekiston: til va madaniyat (Amaliy filologiya), 2022, 5(4).* http://aphil.tsuull.uz/index.php/language-and-culture/article/download/32/29
10. B.Elov, Z.Xusainova, N.Xudayberganov. O`zbek tili korpusi matnlari uchun TF-IDF statistik ko`rsatkichni hisoblash. *SCIENCE AND INNOVATION INTERNATIONAL SCIENTIFIC JOURNAL VOLUME 1 ISSUE 8 UIF-2022: 8.2 | ISSN: 2181-3337*
11. *https://www.academia.edu/105829396/OZBEK_TILI_KORPUSI_MATNLARI_UCHUN_TF_IDF_STATISTIK_KORSATKICHNI_HISOBLASH*
12. Fu, Y., & Yu, Y. (2020). Research on text representation method based on improved TF-IDF. *Journal of Physics: Conference Series*, *1486*(7). https://doi.org/10.1088/1742-6596/1486/7/072032
13. Maharjan, S., Mave, D., Shrestha, P., Montes-Y-Gómez, M., González, F. A., & Solorio, T. (2019). Jointly learning author and annotated character N-gram embeddings: A case study in literary text. *International Conference Recent Advances in Natural Language Processing, RANLP*, *2019-September*. https://doi.org/10.26615/978-954-452-056-4_080
14. Wawrzyński, A., & Szymański, J. (2021). Study of statistical text representation methods for performance improvement of a hierarchical attention network. *Applied Sciences (Switzerland)*, *11*(13). https://doi.org/10.3390/app11136113
15. Zhao, J. S., Song, M. X., Gao, X., & Zhu, Q. M. (2022). Research on Text Representation in Natural Language Processing. *Ruan Jian Xue Bao/Journal of Software*, *33*(1). https://doi.org/10.13328/j.cnki.jos.006304
16. Babić, K., Martinčić-Ipšić, S., & Meštrović, A. (2020). Survey of neural text representation models. In *Information (Switzerland)* (Vol. 11, Issue 11). https://doi.org/10.3390/info11110511
17. Eleyan, A., & Demirel, H. (2011). Co-occurrence matrix and its statistical features as a new approach for face recognition. *Turkish Journal of Electrical Engineering and Computer Sciences*, *19*(1). https://doi.org/10.3906/elk-0906-27
18. Cahyani, D. E., & Patasik, I. (2021). Performance comparison of tf-idf and word2vec models for emotion text classification. *Bulletin of Electrical Engineering and Informatics*, *10*(5). https://doi.org/10.11591/eei.v10i5.3157
19. Method, N. W., Goldberg, Y., Levy, O., Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2014). word2vec Explained : Deriving Mikolov et al. *ArXiv:1402.3722 [Cs, Stat]*, *2*.
20. Xiong, Z., Shen, Q., Xiong, Y., Wang, Y., & Li, W. (2019). New generation model of word vector representation based on CBOW or skip-gram. *Computers, Materials and Continua*, *60*(1). https://doi.org/10.32604/cmc.2019.05155
21. Jang, B., Kim, I., & Kim, J. W. (2019). Word2vec convolutional neural networks for classification of news articles and tweets. *PLoS ONE*, *14*(8). https://doi.org/10.1371/journal.pone.0220976
22. Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*. https://doi.org/10.3115/v1/d14-1162
23. Kutuzov, A., & Kuzmenko, E. (2021). Representing ELMo embeddings as two-

dimensional text online. *EACL 2021 - 16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the System Demonstrations*. https://doi.org/10.18653/v1/2021.eacl-demos.18

24. Joshi, M., Levy, O., Weld, D. S., & Zettlemoyer, L. (2019). BERT for coreference resolution: Baselines and analysis. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*. https://doi.org/10.18653/v1/d19-1588