Жинақталған мәліметтерді тазалау әдістерімен танысу

Болатбек М.А.

Деректер қоймаларын құру кезінде оған кіретін ақпаратты тазартуға әлі де жеткілікті көңіл бөлінбейді. Шамасы, сақтау мөлшері неғұрлым үлкен болса, соғұрлым жақсы болады деп саналады. Бұл деректер қоймасын қоқыс полигонына айналдырудың сенімді әдісі.

Деректерді тазалау қажет! Өйткені, ақпарат гетерогенді және әрдайым көптеген көздерден жиналады. Бұл тазарту процесін соншалықты күрделі және өзекті ететін әртүрлі деректерді жинау нүктелерінің болуымен түсіндіріледі.

Қателіктер әрдайым болады және олардан толықтай құтылу мүмкін емес. Мүмкін кейде оларды түзетуге ресурстар жұмсамай, оларды бір ретке келтірген дұрыс болар. Бірақ, жалпы алғанда, сіз кез-келген жолмен қателіктер санын қолайлы деңгейге дейін азайтуға тырысуыңыз керек. Сонымен қатар, мәселенің психологиялық аспектісін ескеру қажет. Егер талдаушы немесе шешім қабылдаушы деректер қоймасынан алатын сандарға сенімді болмаса, онда ол басқа көздерден алынған ақпаратқа сүйенеді. Бұл сақтау құнын айтарлықтай төмендетеді.

Қателіктердің түрлері

Біз типтік сәйкессіздік, енгізу форматтарындағы және кодтаудағы айырмашылықтар сияқты қарапайым қателерді қарастырмаймыз. Яғни, ақпарат бірнеше көздерден алынған жағдайлар, онда әр түрлі келісімдер бірдей фактіні көрсету үшін қабылданады.Мұндай қате адамның жынысын белгілеу есептерінде жиі кездеседі. Бір жерде ол Е/Ә, бір жерде 1/0, бір жерде шын/жалған деп белгіленеді. Мұндай қателіктермен оларды кодтау және типтерді енгізу ережелерін белгілеу арқылы күреседі, сондықтан олар салыстырмалы түрде оңай шешіледі. Бізді қарапайым тәсілдермен шешілмейтін күрделі мәселелер қызықтырады.Күрделі қателіктердің түрлері өте көп. Әмбебаптықтан басқа, белгілі бір пәндік аймаққа немесе тапсырмаға ғана тән қателер бар. Бірақ тапсырмаға тәуелді емес мысалдарды қарастырайық:

1. Ақпараттың сәйкессіздігі;
2. Мәліметтердің қалып кетуі (пропуски);
3. Қалыпты емес мәндер;
4. Шу;
5. Деректерді енгізу қателері.

Осы мәселелердің әрқайсысын шешу үшін дәлелденген әдістер бар. Әрине, қателерді қолмен басқаруға болады, бірақ деректердің үлкен көлемімен бұл қиынға соғады. Сондықтан, біз осы мәселелерді автоматты режимде адамның минималды қатысуымен шешудің нұсқаларын қарастырамыз.

Ақпараттың сәйкессіздігі

Алдымен қарама-қайшылық деп нені қарастыру керектігін шешу керек. Бір қызығы, бұл тривиалды емес міндет. Мысалы, зейнетақы картасын Тегі, Аты, Әкесінің аты және жынысы өзгерген жағдайда өзгерту керек. Демек, адам әйел болып туылып, ер адам зейнетке шықты, ешқандай қайшылық жоқ!Қарама-қайшылықты не деп санайтынымызды шешіп, осындай жазбаларды тапқаннан кейін бірнеше әрекет нұсқалары бар:

1. Егер бірнеше қарама-қайшы жазбалар табылса, олардың барлығын жойыңыз немесе кез-келген қарапайым ереже бойынша таңдалған нұсқалардың бірін қалдырыңыз. Мысалы, соңғы жазба. Бұл әдіс тривиалды, сондықтан оңай жүзеге асырылады. Кейде бұл жеткілікті.
2. Статистиканы ескере отырып, қарама-қайшы деректерді түзетіңіз. Мысалы, қарама-қайшылықты мәндердің әрқайсысының пайда болу ықтималдығын есептеп, ең ықтималдысын таңдауға болады. Көбінесе бұл әдіс дұрыс нәтиже береді.

Мәліметтердің қалып кетуі (пропуски);

Бұл мәселе көптеген деректер қоймаларының қиындығы болып табылады. Болжау әдістерінің көпшілігі деректер біркелкі тұрақты ағынмен келеді деген болжамнан туындайды. Іс жүзінде бұл сирек кездеседі. Сондықтан, деректер қоймаларын қолданудың ең танымал бағыттарының бірі — болжау-нашар немесе айтарлықтай шектеулермен жүзеге асырылады. Бұл құбылыспен күресу үшін келесі әдістерді қолдануға болады:
1. Жуықтау. Яғни, егер қандай-да бір нүктеде деректер болмаса, біз оның айналасын алып, белгілі формулаларға сәйкес осы нүктедегі мәнді есептейміз, қоймаға тиісті жазба қосамыз. Бұл реттелген деректер үшін жақсы жұмыс істейді. Мысалы, күнделікті өнімді сату туралы ақпарат.

2. Ең орынды мағынаны анықтау. Ол үшін нүктенің айналасы емес, барлық деректер алынады. Бұл әдіс реттелмеген ақпарат үшін қолданылады, яғни зерттелетін нүктенің айналасы не екенін анықтай алмайтын жағдайлар.

Қалыпты емес мәндер

Кейде жалпы көріністен қатты шығып кететін деректер кездеседі. Мысалы, өнімнің бағасы орташа деңгейден 10 есе жоғары. Мұндай мәндер жақсы реттеледі. Талдау алгоритмдері процестердің табиғаты туралы ештеңе білмейді. Сондықтан кез-келген аномалия толығымен қалыпты мән ретінде қабылданады. Осыған байланысты модель қатты бұрмаланады кездейсоқ сәтсіздік немесе сәттілік үлгі болып саналады.Бұл проблемамен күресудің әдісі бар - қатаң бағалау. Мысал ретінде медианалық сүзгі бола алады. Біз қолда бар деректерді бағалаймыз және рұқсат етілген шекарадан асатын кез келген нәрсеге келесі әрекеттердің бірін қолданамыз:

1. Қалыпты емес мәндер алынып тасталады;
2. Аномалды деректер жақын шекаралық мәндерге ауыстырылады.

Шу

Талдау кезінде әрдайым дерлік біз шуылға кезігеміз. Көбінесе шу пайдалы ақпарат бермейді, тек суретті анық көруге кедергі келтіреді. Бұл құбылыспен күресудің бірнеше әдістері бар:

1. Спектрлік талдау. Оның көмегімен біз деректердің жоғары жиілікті компоненттерін кесіп тастай аламыз, яғни шу-бұл негізгі сигналға жақын жиі және шамалы тербелістер.
2. Авторегрессиялық әдістер. Бұл өте кең таралған әдіс уақыт қатарларын талдауда белсенді қолданылады. Ол процесті сигнал және шу ретінде сипаттайтын функцияны табуға дейін азаяды. Шын мәнінде, осыдан кейін шуды алып тастауға және негізгі сигналды қалдыруға болады.

Мәліметтерді енгізудегі қателіктер

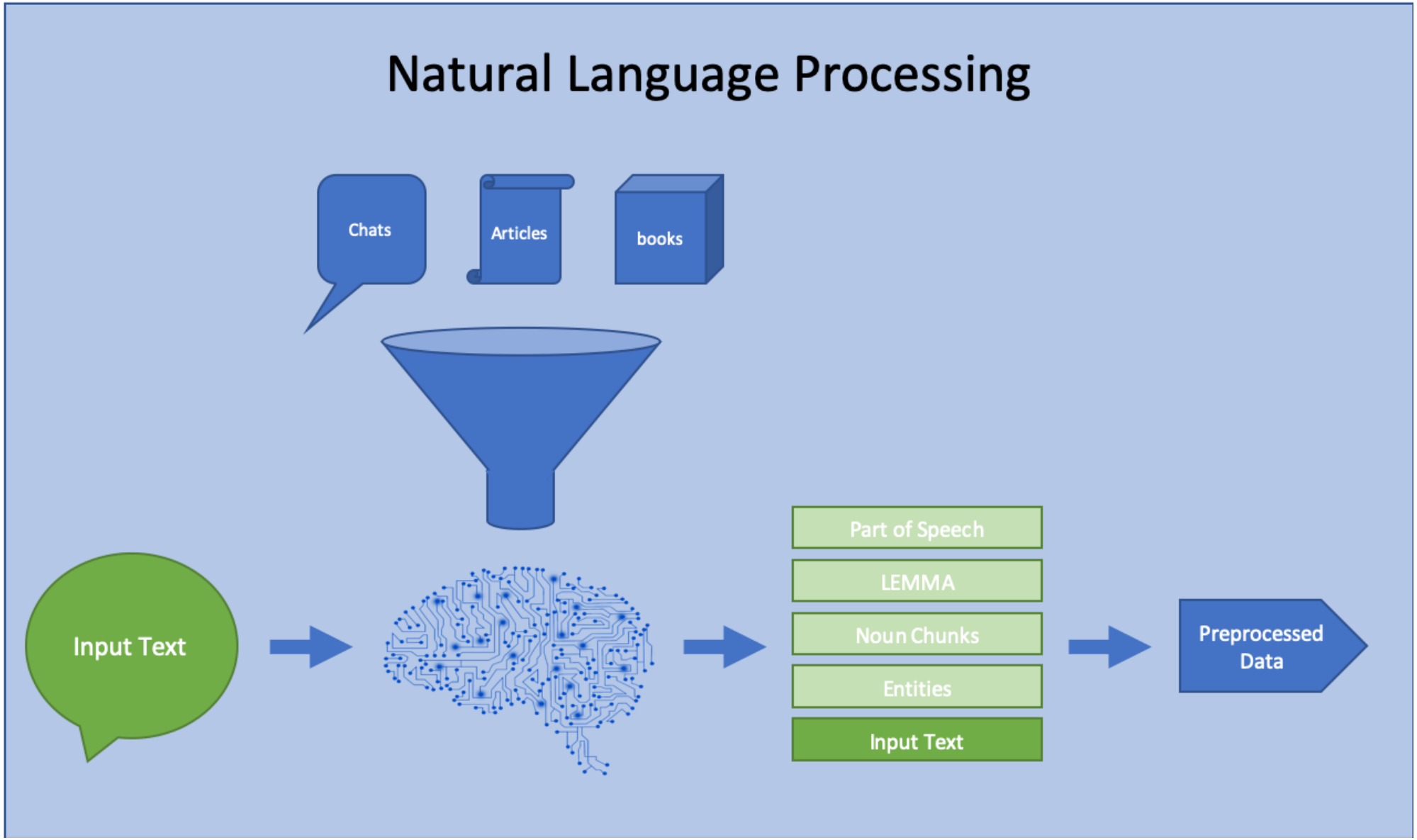
Бұл жеке тақырып, өйткені мұндай қателіктердің саны тым үлкен, мысалы, теру, деректерді саналы түрде бұрмалау, форматтардың сәйкес келмеуі және деректерді енгізу қосымшасының ерекшеліктеріне байланысты қателерді есептемеу. Олардың көпшілігімен күресу үшін дәлелденген әдістер бар. Кейбір нәрселер айқын, мысалы, деректерді қоймаға енгізбес бұрын форматтарды тексеруге болады. Кейбіреулері күрделі. Мысалы, әр түрлі тезаурус негізінде терулерді түзетуге болады. Бірақ, кез-келген жағдайда, сіз осындай қателіктерден тазартуыңыз керек.

Деректерді алдын-ала өңдеу машинаны оқыту моделін құрудағы маңызды қадам болып табылады және деректердің қаншалықты жақсы өңделгеніне байланысты нәтижелер көрінеді. NLP-да мәтінді алдын-ала өңдеу модель құру процесінің алғашқы қадамы болып табылады.

Мәтінді алдын-ала өңдеудің/тазартудың кейбір жалпы кезеңдері:

- Төменгі әріптер
- Тыныс белгілерін жою
- Тоқтату сөздерін жою
- Жиі кездесетін сөздерді жою
- Сирек сөздерді жою
- Стемминг
- Лемматизация
- Эмотикондарды жою
- Эмодзилерді жою
- Смайликтерді сөзге түрлендіру
- URL мекен-жайларын жою
- HTML тегтерін жою
- Чаттағы сөздерді түрлендіру
- Емлені түзету

Natural Language Processing



Сонымен, бұл мәтіндік деректермен жасай алатын мәтінді алдын-ала өңдеудің әртүрлі түрлері. Бірақ мұның бәрін үнемі жасаудың қажеті жоқ. Пайдалану жағдайына байланысты алдын-ала емдеу кезеңдерін мұқият таңдау керек, өйткені бұл да маңызды рөл атқарады.Мысалы, көңіл-күйді талдау кезінде эмодзилерді жоюдың қажеті жоқ, өйткені бұл көңіл-күй туралы маңызды ақпаратты жеткізеді.

Төменгі регистр

Төменгі регистр - бұл мәтінді алдын-ала өңдеудің кең таралған әдісі. Идея - «МӘТІН" және "мәтін" бірдей өңделуі үшін кіріс мәтінін бірдей форматқа түрлендіру.Бұл мәтінді сипаттау әдістері үшін жиілік, TF idf сияқты пайдалы, өйткені ол бірдей сөздерді біріктіруге көмектеседі, осылайша қайталануды азайтады және дұрыс/tfidf мәндерін алады.Сөйлеудің бір бөлігін белгілеу (дұрыс корпус зат есімдер туралы кейбір ақпарат беретін жерде және т. б.) және көңіл-күйді талдау (жоғарғы корпус ашуды білдіретін жерде және т. б.) сияқты тапсырмаларды орындау кезінде бұл пайдасыз болуы мүмкін.Әдепкі бойынша, Төменгі корпус sklearn TfidfVectorizer және Keras Tokenizer сияқты көптеген заманауи векторизаторлар мен токенизаторларда жасалады. Сондықтан біз олар үшін пайдалану жағдайына байланысты қажет болған жағдайда жалған мәнді орнатуымыз керек.

In [2]:

```
df["text_lower"] = df["text"].str.lower()
df.head()
```

Out[2]:

	text	text_lower
0	@115712 I understand. I would like to assist y...	@115712 i understand. i would like to assist y...
1	@sprintcare and how do you propose we do that	@sprintcare and how do you propose we do that
2	@sprintcare I have sent several private messag...	@sprintcare i have sent several private messag...
3	@115712 Please send us a Private Message so th...	@115712 please send us a private message so th...
4	@sprintcare I did.	@sprintcare i did.

Тыныс белгілерін жою

Мәтінді алдын-ала өңдеудің тағы бір кең таралған әдісі-мәтіндік деректерден тыныс белгілерін жою. Бұл тағы да мәтінді стандарттау процесі, ол " ура "және" ура!«тіркестерін бір тркес ретінде тану үшін қажет.Сондай-ақ, пайдалану жағдайына байланысты алып тастау үшін тыныс белгілерінің тізімін мұқият таңдау керек. Мысалы, python-да string.punctuation келесі тыныс белгілерін қамтиды

!"#$%&\'()*+,-./:;<=>?@[\\]^_{|}~`

Біздің қажеттіліктерімізге сәйкес көбірек тыныс белгілерін қосуға немесе жоюға болады.

In [3]:

```
# drop the new column created in last cell
df.drop(["text_lower"], axis=1, inplace=True)

PUNCT_TO_REMOVE = string.punctuation
def remove_punctuation(text):
    """custom function to remove the punctuation"""
    return text.translate(str.maketrans('', '', PUNCT_TO_REMOVE))

df["text_wo_punct"] = df["text"].apply(lambda text: remove_punctuation(text))
df.head()
```

Out[3]:

	text	text_wo_punct
0	@115712 I understand. I would like to assist y...	115712 I understand I would like to assist you...
1	@sprintcare and how do you propose we do that	sprintcare and how do you propose we do that
2	@sprintcare I have sent several private messag...	sprintcare I have sent several private message...
3	@115712 Please send us a Private Message so th...	115712 Please send us a Private Message so tha...
4	@sprintcare I did.	sprintcare I did

Тоқтату сөздерін жою

Тоқтату сөздері-бұл "the", "a" және т.б. сияқты тілде жиі кездесетін сөздер. Көп жағдайда оларды мәтіннен алып тастауға болады, өйткені олар кейінгі талдау үшін құнды ақпарат бермейді. Сөйлеудің бір бөлігін белгілеу сияқты жағдайларда біз оларды алып тастамауымыз керек, өйткені олар мүмкіндік туралы өте құнды ақпарат береді.Бұл тоқтату сөздерінің тізімдері әртүрлі тілдерге арналған және біз оларды қауіпсіз қолдана аламыз. Мысалы, nltk пакетінен ағылшын тіліне арналған сөздерді тоқтату тізімін төменде көруге болады.

```
from nltk.corpus import stopwords
", ".join(stopwords.words('english'))
```

```
"i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your,
yours, yourself, yourselves, he, him, his, himself, she, she's, her, hers, herself, it,
it's, its, itself, they, them, their, theirs, themselves, what, which, who, whom, this,
that, that'll, these, those, am, is, are, was, were, be, been, being, have, has, had, ha
ving, do, does, did, doing, a, an, the, and, but, if, or, because, as, until, while, of,
at, by, for, with, about, against, between, into, through, during, before, after, above,
below, to, from, up, down, in, out, on, off, over, under, again, further, then, once, he
re, there, when, where, why, how, all, any, both, each, few, more, most, other, some, su
ch, no, nor, not, only, own, same, so, than, too, very, s, t, can, will, just, don, do
n't, should, should've, now, d, ll, m, o, re, ve, y, ain, aren, aren't, couldn, could
n't, didn, didn't, doesn, doesn't, hadn, hadn't, hasn, hasn't, haven, haven't, isn, is
n't, ma, mightn, mightn't, mustn, mustn't, needn, needn't, shan, shan't, shouldn, should
n't, wasn, wasn't, weren, weren't, won, won't, wouldn, wouldn't"
```

Сол сияқты, біз басқа тілдер үшін де тізімді алып, оларды қолдана аламыз.

In [5]:

```
STOPWORDS = set(stopwords.words('english'))
def remove_stopwords(text):
    """custom function to remove the stopwords"""
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])

df["text_wo_stop"] = df["text_wo_punct"].apply(lambda text: remove_stopwords(text))
df.head()
```

Out[5]:

	text	text_wo_punct	text_wo_stop
0	@115712 I understand. I would like to assist y...	115712 I understand I would like to assist you...	115712 I understand I would like assist We wou...
1	@sprintcare and how do you propose we do that	sprintcare and how do you propose we do that	sprintcare propose
2	@sprintcare I have sent several private messag...	sprintcare I have sent several private message...	sprintcare I sent several private messages one...
3	@115712 Please send us a Private Message so th...	115712 Please send us a Private Message so tha...	115712 Please send us Private Message assist J...
4	@sprintcare I did.	sprintcare I did	sprintcare I

Жиі кездесетін сөздерді жою

Алдын ала өңдеудің алдыңғы кезеңінде біз тілдік ақпаратқа негізделген сөздерді жойдық. Бірақ, айталық, егер бізде белгілі бір аймаққа тән корпус болса, онда біз үшін онша маңызды емес жиі кездесетін сөздер болуы мүмкін.Осылайша, бұл қадам берілген корпустағы жиі кездесетін сөздерді жою болып табылады. Егер біз tf idf сияқты нәрсені қолданатын болсақ, ол автоматты түрде көмекке келеді.Ең көп кездесетін сөздерді алайық, содан кейін оларды келесі қадамда алып тастайық.

In [6]:

```
from collections import Counter
cnt = Counter()
for text in df["text_wo_stop"].values:
    for word in text.split():
        cnt[word] += 1

cnt.most_common(10)
```

Out[6]:

```
[('I', 1437),
 ('us', 752),
 ('DM', 514),
 ('help', 479),
 ('Please', 376),
 ('We', 338),
 ('Hi', 293),
 ('Thanks', 287),
 ('get', 279),
 ('please', 247)]
```

In [7]:

```
FREQWORDS = set([w for (w, wc) in cnt.most_common(10)])
def remove_freqwords(text):
    """custom function to remove the frequent words"""
    return " ".join([word for word in str(text).split() if word not in FREQWORDS])

df["text_wo_stopfreq"] = df["text_wo_stop"].apply(lambda text: remove_freqwords(text))
df.head()
```

Out[7]:

	text	text_wo_punct	text_wo_stop	text_wo_stopfreq
0	@115712 I understand. I would like to assist y...	115712 I understand I would like to assist you...	115712 I understand I would like assist We wou...	115712 understand would like assist would need...
1	@sprintcare and how do you propose we do that	sprintcare and how do you propose we do that	sprintcare propose	sprintcare propose
2	@sprintcare I have sent several private messag...	sprintcare I have sent several private message...	sprintcare I sent several private messages one...	sprintcare sent several private messages one r...
3	@115712 Please send us a Private Message so th...	115712 Please send us a Private Message so tha...	115712 Please send us Private Message assist J...	115712 send Private Message assist Just click ...
4	@sprintcare I did.	sprintcare I did	sprintcare I	sprintcare

Сирек сөздерді жою

Бұл алдын-ала өңдеудің алдыңғы кезеңіне өте ұқсас, бірақ біз сирек кездесетін сөздерді корпустан алып тастаймыз.

In [8]:

```
# Drop the two columns which are no more needed
df.drop(["text_wo_punct", "text_wo_stop"], axis=1, inplace=True)

n_rare_words = 10
RAREWORDS = set([w for (w, wc) in cnt.most_common()[:-n_rare_words-1:-1]])
def remove_rarewords(text):
    """custom function to remove the rare words"""
    return " ".join([word for word in str(text).split() if word not in RAREWORDS])

df["text_wo_stopfreqrare"] = df["text_wo_stopfreq"].apply(lambda text: remove_rarewords(tex
t))
df.head()
```

Out[8]:

	text	text_wo_stopfreq	text_wo_stopfreqrare
0	@115712 I understand. I would like to assist y...	115712 understand would like assist would need...	115712 understand would like assist would need...
1	@sprintcare and how do you propose we do that	sprintcare propose	sprintcare propose
2	@sprintcare I have sent several private messag...	sprintcare sent several private messages one r...	sprintcare sent several private messages one r...

Біз сөздердің бүкіл тізімін (сөздерді тоқтату, жиі кездесетін сөздер және сирек кездесетін сөздер) біріктіріп, оларды бірден жою үшін бірыңғай тізімді жасай аламыз.

Стемминг

Стемминг-бұл флексивті (немесе кейде туынды) сөздерді олардың сөзжасамдық, негізгі немесе түбірлік формасына келтіру процесі. Мысалы, егер корпуста "жүргізді" және "жүру" деген екі сөз болса, онда оларды жүруге мәжбүр ететін жұрнақ негіз болады. Бірақ тағы бір мысалда бізде "жұбаныш" және "жұбанышты" деген екі сөз бар, стеммер жұрнақты алып тастап, оларды түбір етеді.Қол жетімді стемминг алгоритмдерінің бірнеше түрлері бар және олардың бірі-Портердің стеммері, ол кеңінен қолданылады. Біз nltk пакетін сол үшін қолдана аламыз.

In [9]:

```
from nltk.stem.porter import PorterStemmer

# Drop the two columns
df.drop(["text_wo_stopfreq", "text_wo_stopfreqrare"], axis=1, inplace=True)

stemmer = PorterStemmer()
def stem_words(text):
    return " ".join([stemmer.stem(word) for word in text.split()])

df["text_stemmed"] = df["text"].apply(lambda text: stem_words(text))
df.head()
```

Out[9]:

	text	text_stemmed
0	@115712 I understand. I would like to assist y...	@115712 I understand. I would like to assist y...
1	@sprintcare and how do you propose we do that	@sprintcar and how do you propos we do that
2	@sprintcare I have sent several private messag...	@sprintcar I have sent sever privat messag and...
3	@115712 Please send us a Private Message so th...	@115712 pleas send us a privat messag so that ...
4	@sprintcare I did.	@sprintcar I did.

Біз private және propose сияқты сөздердің шығу тегіне байланысты соңында E әрпі бар екенін көреміз. Бұл оған арналмаған. Ол үшін не істей аламыз? Мұндай жағдайларда біз лемматизацияны қолдана аламыз.Сондай-ақ, бұл porter стеммері ағылшын тіліне арналған. Егер біз басқа тілдермен жұмыс жасасақ, snowball stemmer қолдана аламыз. Snowballstemmer үшін Қолдау көрсетілетін тілдер:

In [10]:

```
from nltk.stem.snowball import SnowballStemmer
SnowballStemmer.languages
```

Out[10]:

```
('danish',
 'dutch',
 'english',
 'finnish',
 'french',
 'german',
 'hungarian',
 'italian',
 'norwegian',
 'porter',
 'portuguese',
 'romanian',
 'russian',
 'spanish',
 'swedish')
```

Лемматизация

Лемматизация өзгертілген сөздерді олардың негізіне дейін қалыптастыру сияқты, бірақ ол түбір сөздің (лемма деп те аталады) тілге тиесілі болуын қамтамасыз ететіндігімен ерекшеленеді.Нәтижесінде бұл процесс стемминг процесіне қарағанда баяу жүреді. Сондықтан жылдамдық талаптарына байланысты біз стеммингті де, лемматизацияны да қолдана аламыз.Біздің ұсыныстарымызды лемматизациялау үшін nltk-де WordNetLemmatizer бағдарламасын қолданайық.

In [11]:

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
def lemmatize_words(text):
    return " ".join([lemmatizer.lemmatize(word) for word in text.split()])

df["text_lemmatized"] = df["text"].apply(lambda text: lemmatize_words(text))
df.head()
```

Out[11]:

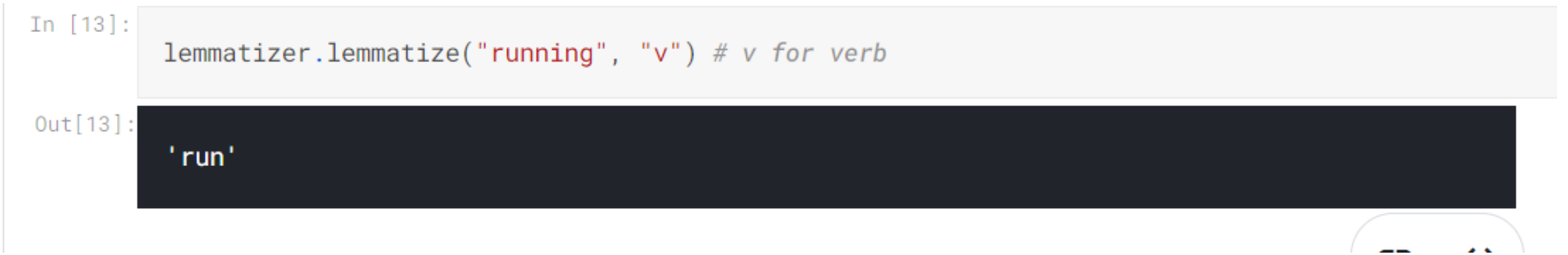
	text	text_stemmed	text_lemmatized
0	@115712 I understand. I would like to assist y...	@115712 I understand. I would like to assist y...	@115712 I understand. I would like to assist y...
1	@sprintcare and how do you propose we do that	@sprintcar and how do you propos we do that	@sprintcare and how do you propose we do that
2	@sprintcare I have sent several private messag...	@sprintcar I have sent sever privat messag and...	@sprintcare I have sent several private messag...
3	@115712 Please send us a Private Message so th...	@115712 pleas send us a privat messag so that ...	@115712 Please send u a Private Message so tha...

Біз сөйлемдегі және жеке сөйлемдегі соңғы E лемматизацияны қолданған кезде, стеммингтен айырмашылығы сақталатынын көреміз. Лемматизацияда тағы бір нәрсе бар. Енді «жүгіруді» лемматизациялауға тырысайық.

```
In [12]: lemmatizer.lemmatize("running")

Out[12]: 'running'
```

Ол оны түбірлік іске қосу формасына өзгертпестен қайта іске қосылды. Себебі, лемматизация процесі дұрыс лемманы алу үшін POS тегіне байланысты. Енді осы сөз үшін POS тегін көрсетіп, қайтадан лемматизациялайық.



Енді біз сөздің түбірін алдық. Сонымен, біз NLTK-тегі лемматизаторға арналған сөзбен бірге POS тегінімізді ұсынуымыз керек. POS-қа байланысты лемматизатор әртүрлі нәтижелерді қайтара алады.Мысалы, stripes деген сөзді алып, оның түбірін етістік және зат есім ретінде іздеп көрейік.

In [14]:

```
print("Word is : stripes")
print("Lemma result for verb : ",lemmatizer.lemmatize("stripes", 'v'))
print("Lemma result for noun : ",lemmatizer.lemmatize("stripes", 'n'))
```

```
Word is : stripes
Lemma result for verb :  strip
Lemma result for noun :  stripe
```

Эмодзилерді жою

Әлеуметтік медиа платформаларын көбірек қолдана отырып, эмодзилерді қолдануда және біздің күнделікті өмірімізде жиі кездеседі. Мәтіндік талдау үшін осы эмодзилерді жою қажет болуы мүмкін.Осы кодтың арқасында мәтіннен эмодзилерді жою үшін төмендегі көмекші функцияны іздеңіз.

In [16]:

```
# Reference : https://gist.github.com/slowkow/7a7f61f495e3dbb7e3d767f97bd7304b
def remove_emoji(string):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', string)

remove_emoji("game is on 🔥🔥")
```

Out[16]:

```
'game is on '
```

Эмотикондарды жою

Бұл біз соңғы қадамда жасадық, иә? Жоқ. Соңғы қадамда біз эмодзилерді жойдық, бірақ эмотикондарды емес. Эмодзи мен эмотикон арасында аздап айырмашылық бар. Grammarist.com сайтында айтылғандай, эмотикон белгілі бір жолмен жиналған кезде бет әлпетін білдіретін пернетақта таңбаларынан тұрады, Эмодзи - бұл нақты сурет.

:-)- бұл Эмотикон

😀 – эмодзи

NeelShah эмодзилердің керемет жиынтығы үшін рахмет, біз оларды эмодзилерді жою үшін қолданамыз.Тағы да назар аударыңыз, эмотикондарды/эмодзилерді жою әрдайым қолайлы емес және шешім нақты пайдалану жағдайына байланысты қабылдануы керек.

```
EMOTICONS = { u":-\)":"Happy face or smiley",
 u":\)":"Happy face or smiley",
u":-\]":"Happy face or smiley",
 u":\]":"Happy face or smiley",
 u":-3":"Happy face smiley",
 u":3":"Happy face smiley",
 u":->":"Happy face smiley",
u":>":"Happy face smiley",
u"8-\)":"Happy face smiley",
 u":o\)":"Happy face smiley",
 u":-\}":"Happy face smiley",
u":\}":"Happy face smiley",
 u":-\)":"Happy face smiley",
 u":c\)":"Happy face smiley«
X,
```

In [19]:

```
def remove_emoticons(text):
    emoticon_pattern = re.compile(u'(' + u'|'.join(k for k in EMOTICONS) + u')')
    return emoticon_pattern.sub(r'', text)

remove_emoticons("Hello :-)")
```

Out[19]:

```
'Hello '
```

In [20]:

```
remove_emoticons("I am sad :(")
```

Out[20]:

```
'I am sad '
```

Эмоцияны сөзге айналдыру

Алдыңғы қадамда біз эмодзилерді жойдық. Көңіл-күйді талдау сияқты пайдалану жағдайларында эмотикондар кейбір құнды ақпарат береді, сондықтан оларды жою жақсы шешім болмауы мүмкін. Мұндай жағдайларда не істей аламыз?Бір жолы-эмодзилерді келесі модельдеу процестерінде қолдануға болатындай етіп word форматына түрлендіру. Біз мұны эмодзилерді сөзге айналдыру үшін қайтадан қолданамыз.

In [21]:

```
def convert_emoticons(text):
    for emot in EMOTICONS:
        text = re.sub(u'('+emot+')', "_".join(EMOTICONS[emot].replace(",","").split()), tex
t)
    return text

text = "Hello :-) :-)"
convert_emoticons(text)
```

Out[21]:

```
'Hello Happy_face_smiley Happy_face_smiley'
```

In [22]:

```
text = "I am sad :()"
convert_emoticons(text)
```

Out[22]:

```
'I am sad Frown_sad_andry_or_poutingConfusion'
```

Смайликтерді сөзге түрлендіру

Енді Эмодзи үшін де солай істейік. Біз бұл сөздікті эмодзилерді тиісті сөздерге айналдыру үшін қолданамыз.Тағы да, бұл түрлендіру белгілі бір пайдалану жағдайлары үшін эмодзилерді алып тастағаннан гөрі жақсы болуы мүмкін. Осы пайдалану жағдайына сәйкес келетінін пайдаланыңыз.

```
EMO_UNICODE = {
    u':1st_place_medal:': u'\U0001F947',
    u':2nd_place_medal:': u'\U0001F948',
    u':3rd_place_medal:': u'\U0001F949',
    u':AB_button_(blood_type):': u'\U0001F18E',
    u':ATM_sign:': u'\U0001F3E7',
    u':A_button_(blood_type):': u'\U0001F170',
    u':Afghanistan:': u'\U0001F1E6 \U0001F1EB',
    u':Albania:': u'\U0001F1E6 \U0001F1F1',
    u':Algeria:': u'\U0001F1E9 \U0001F1FF',
    u':American_Samoa:': u'\U0001F1E6 \U0001F1F8',
    u':Andorra:': u'\U0001F1E6 \U0001F1E9',
    u':Angola:': u'\U0001F1E6 \U0001F1F4',
    u':Anguilla:': u'\U0001F1E6 \U0001F1EE',
    u':Antarctica:': u'\U0001F1E6 \U0001F1F6',
    u':Antigua_&_Barbuda:': u'\U0001F1E6 \U0001F1EC',
    u':Aquarius:': u'\U00002652',
    u':Argentina:': u'\U0001F1E6 \U0001F1F7',
    u':Aries:': u'\U00002648',
```

```
}

UNICODE_EMO = {v: k for k, v in EMO_UNICODE.items()}
```

In [24]:

```
def convert_emojis(text):
    for emot in UNICODE_EMO:
        text = re.sub(r'('+emot+')', "_".join(UNICODE_EMO[emot].replace(",","").replace
(":","").split()), text)
    return text

text = "game is on 🔥"
convert_emojis(text)
```

Out[24]:

```
'game is on fire'
```

In [25]:

```
text = "Hilarious 😂"
convert_emojis(text)
```

URL мекен-жайларын жою

Алдын ала өңдеудің келесі қадамы-деректердегі барлық URL мекен-жайларын жою. Мысалы, егер біз twitter-ді талдайтын болсақ, онда Твиттерде URL мекен-жайы болуы мүмкін. Әрі қарай талдау үшін оларды алып тастау қажет болуы мүмкін.Ол үшін төмендегі код үзіндісін қолдана аламыз.

In [26]:

```
def remove_urls(text):
    url_pattern = re.compile(r'https?://\S+|www\.\S+')
    return url_pattern.sub(r'', text)
```

In [27]:

```
text = "Driverless AI NLP blog post on https://www.h2o.ai/blog/detecting-sarcasm-is-difficul
t-but-ai-may-have-an-answer/"
remove_urls(text)
```

Out[27]:

```
'Driverless AI NLP blog post on '
```

Now let us take a `http` url and check the code

In [28]:

```
text = "Please refer to link http://lnkd.in/ecnt5yC for the paper"
remove_urls(text)
```

Out[28]:

```
'Please refer to link  for the paper'
```

HTML тегтерін жою

Көптеген жерлерде пайдалы болатын алдын - ала өңдеудің тағы бір кең таралған әдісі-html тегтерін жою. Бұл әсіресе әртүрлі веб-сайттардан деректерді жойсақ пайдалы. Сайып келгенде, біз мәтіннің бөлігі ретінде html жолдарын ала аламыз.Алдымен HTML тегтерін тұрақты өрнектермен жоюға тырысайық.

In [30]:

```
def remove_html(text):
    html_pattern = re.compile('<.*?>')
    return html_pattern.sub(r'', text)

text = """<div>
<h1> H2O</h1>
<p> AutoML</p>
<a href="https://www.h2o.ai/products/h2o-driverless-ai/"> Driverless AI</a>
</div>"""

print(remove_html(text))
```

```
H2O
AutoML
Driverless AI
```

HTML құжатынан мәтінді алу үшін біз BeautifulSoup пакетін қолдана аламыз.

In [31]:

```
from bs4 import BeautifulSoup

def remove_html(text):
    return BeautifulSoup(text, "lxml").text

text = """<div>
<h1> H2O</h1>
<p> AutoML</p>
<a href="https://www.h2o.ai/products/h2o-driverless-ai/"> Driverless AI</a>
</div>
"""

print(remove_html(text))
```

```
H2O
AutoML
Driverless AI
```

Чаттағы сөздерді түрлендіру

Егер біз чат деректерімен айналысатын болсақ, бұл мәтінді алдын-ала өңдеудің маңызды кезеңі. Адамдар чатта көптеген қысқартылған сөздерді қолданады, сондықтан бұл сөздерді біздің талдау мақсаттарымызға кеңейту пайдалы болар еді. Чаттағы жаргон сөздерінің жақсы тізімін алды. Мұны біз мұнда түрлендіру үшін қолдана аламыз. Бұл тізімге қосымша сөздер қосуға болады.

In [32]:

```
chat_words_str = """
AFAIK=As Far As I Know
AFK=Away From Keyboard
ASAP=As Soon As Possible
ATK=At The Keyboard
ATM=At The Moment
A3=Anytime, Anywhere, Anyplace
BAK=Back At Keyboard
BBL=Be Back Later
BBS=Be Back Soon
BFN=Bye For Now
B4N=Bye For Now
BRB=Be Right Back
BRT=Be Right There
BTW=By The Way
B4=Before
B4N=Bye For Now
CU=See You
CUL8R=See You Later
CYA=See You
FAQ=Frequently Asked Questions
```

In [33]:

```
chat_words_map_dict = {}
chat_words_list = []
for line in chat_words_str.split("\n"):
    if line != "":
        cw = line.split("=")[0]
        cw_expanded = line.split("=")[1]
        chat_words_list.append(cw)
        chat_words_map_dict[cw] = cw_expanded
chat_words_list = set(chat_words_list)

def chat_words_conversion(text):
    new_text = []
    for w in text.split():
        if w.upper() in chat_words_list:
            new_text.append(chat_words_map_dict[w.upper()])
        else:
            new_text.append(w)
    return " ".join(new_text)

chat_words_conversion("one minute BRB")
```

```
            new_text.append(w)
    return " ".join(new_text)

chat_words_conversion("one minute BRB")
```

Out[33]:

```
'one minute Be Right Back'
```

In [34]:

```
chat_words_conversion("imo this is awesome")
```

Out[34]:

```
'In My Opinion this is awesome'
```

Емлені түзету

Мәтінді алдын-ала өңдеудің тағы бір маңызды кезеңі-емлені түзету. Терулер мәтіндік деректерде жиі кездеседі және біз талдау жасамас бұрын емле қателерін түзеткіміз келуі мүмкін.Егер біз өзіміздің емлені түзеткішті жазғымыз келсе, Питер Норвигтің әйгілі кодынан бастауға болады.Бұл дәптерде емлені түзету үшін python package ру емлесін тексеру құралын қолданайық.

In [35]:

```
!pip install pyspellchecker
```

```
Requirement already satisfied: pyspellchecker in /opt/conda/lib/python3.6/site-packages (0.
5.0)
```

In [36]:

```
from spellchecker import SpellChecker

spell = SpellChecker()
def correct_spellings(text):
    corrected_text = []
    misspelled_words = spell.unknown(text.split())
    for word in text.split():
        if word in misspelled_words:
            corrected_text.append(spell.correction(word))
        else:
            corrected_text.append(word)
    return " ".join(corrected_text)

text = "speling correctin"
correct_spellings(text)
```

Out[36]:

```
'spelling correction'
```

In [37]:

```
text = "thnks for readin the notebook"
correct_spellings(text)
```

Out[37]:

```
'thanks for reading the notebook'
```

Назарларыңызға рақмет!