ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ БІЛІМ МИНИСТРЛІГІ

Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық университеті

Автоматика және ақпараттық технологиялар институты

Бағдарламалық инженерия кафедрасы

Жеңісов Айдос Жеңісұлы

МАГИСТРЛІК ДИССЕРТАЦИЯ

Магистр акедемиялық дәрежесін алу үшін

Диссертация тақырыбы	Қазақ тіліндегі деректерді сентименталды талдау моделі мен әдістерін зерттеу
Мамандық бағыты	7M06101 – Software Engineering

Ғылыми жетекшісі,
PhD, қауымдастырылған
профессор,
_____Н.К. Мукажанов
«__» _____ 2023 г.

Пікір беруші,
PhD, «Компьютерлік
инженерия» кафедрасының
меңгерушісі
_____ Т.Т. Чинибаева
«__» _____ 2023 г.

Норма бақылаушы,
PhD, қауымдастырылған
профессор,
_____ А.Т. Ахмедиярова
«__» _____ 2023 г.

ҚОРҒАУҒА ЖІБЕРІЛДІ
Кафедра меңгерушісі
физ-мат. ғыл кандидаты,
профессор
_____ А. Н. Молдагулова
«__»_____ 2023 г.

Алматы 2023

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ БІЛІМ МИНИСТРЛІГІ

Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық университеті

Автоматика және ақпараттық технологиялар институты

Бағдарламалық инженерия кафедрасы

Мамандық: 7М06101– Software Engineering

БЕКІТЕМІН
БИ Кафедра меңгерушісі
физ-мат. ғыл кандидаты,
профессор
_____ А. Н. Молдагулова
«__» _____ 2023 г.

Магистрлік диссертацияны орындауға арналған
ТАПСЫРМА

магистрант Жеңісов Айдос Жеңісұлына
Диссертация тақырыбы: «Қазақ тіліндегі деректерді сентименталды талдау моделі мен әдістерін зерттеу»
Аяқталған диссертация тапсыру уақыты «__» _____

Магистрлік диссертацияның бастапқы деректері: Қазақ тіліндегі сентименталды деректерді талдаудың моделдерін мен әдістері ұсынылып, толығырақ сипатталды. Зерттеу жұмысының қойылған мақсаттардың ішінде: сентименталды талдау әдістерін зерттей отырып, моделдер әзірленді; Зерттелген жұмысты талдап, соған байланысты олардың тиімділігі бағаланды

Магистрлік диссертацияда әзірленген сұрақтар тізімі немесе магистрлік жұмыстың қысқаша мазмұны: а) зерттеу алды жоба жұмысымен танысу; ә) веб-ресурстардан өзекті деректерді жинау; б) сентименталды талдаудың моделдері мен алгоритмдері; в) зерттеу нәтижелерін талдап соған байланысты тиімділігін анықтау.

Ұсынылатын негізгі қолданылған әдебиеттер: M. Sokolova, G. Lapalme, A systematic analysis of performance measures for classification tasks, Information Processing & Management 45 (2009) 427–437; Web-sentiment Analysis Of Public Comments (Public Reviews) For Languages With Limited Resources Such As The Kazakh Language, Dinara Gimadi, Richard Evans, Kiril Simov; Y. Chandra, A. Jana, Sentiment analysis using machine learning and deep learning, in: 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom), 2020

Магистрлік диссертацияны дайындауға арналған

ГРАФИК

Бөлімдердің атауы, әзірленетін сұрақтар тізімі	Ғылыми жетекшіге ұсыну мерзімі	Ескертулер
Бөлім 1. Сентименталды деректерді талдауда қолданылатын негізгі ұғымдар және әдістерге шолу	30.10.2022	Орындалды
Бөлім 2. Веб-ресурстардан өзекті деректерді жинау, олардың негізінде машина алдын ала өңдеудң жүзеге асыру	03.03.2023	Орындалды
Бөлім 3. Сентименталды талдау әдістерін қолдану	24.04.2023	Орындалды

Көрсетілген жоба бойынша бөлімдерге байланысты консультациялар

Бөлім	Консультант, (ғылыми дәрежесі)	Уақыт	Қолы
Стандартизация Норма бақылаушы	PhD, қауымдастырылған профессор, А.Т. Ахмедиярова		
Бағдарламалық қамтамасыз ету	PhD, қауымдастырылған профессор, Н.К. Мукажанов		
Антиплагиат	Ассистент, Б.С. Әубәкіров		

Тапсырманың берілген күні "____"_____ 2023 г.

Кафедра меңгерушісі _____ А.Н. Молдагулова

Ғылыми жетекшісі _____ Н.К. Мукажанов

Тапсырманы орындауға магистрант қабылдады _____ А.Ж. Жеңісов

Күні "_____"_____ 2023 ж.

АҢДАТПА

Қазіргі уақытта әлеуметтік желідегі жазбалар, жаңалықтар және тұтынушылардың пікірлері сияқты қазақ тіліндегі цифрлық контентті пайдаланудың артуымен үлкен көлемдегі деректерді өңдей алатын және адамдардың пікірі мен көзқарасы туралы құнды ақпарат беретін автоматтандырылған сентименталды талдау әдістеріне қажеттілік артып отыр. Бұл мақалада қазақ тіліндегі деректердің сентименталды талдау моделдері мен әдістері қарастылады. Сентименталды талдау деректер жинағы ретінде веб-ресурстардан пікір деректер жиналып, бұл жаңа деректерге машиналық және терең оқыту әдістерін қолдана отырып олардың тиімділігін бағаланды. Мақала деректердің қалай жиналғаны, сапаны оңтайландыру үшін оның алдын ала өңделу жолы, сондай-ақ бағалау процесіне арналған тәжірбиелер туралы ақпарат қамтиды.

Түйін сөздер: сентименталды талдау, қазақ тілі, деректер жинағы, сөз, көңіл-күй

АННОТАЦИЯ

В настоящее время с ростом использования цифрового контента на казахском языке, такого как сообщения в социальных сетях, новости и отзывы клиентов, растет потребность в автоматизированных методах анализа тональности, которые могут обрабатывать большие объемы данных и предоставлять ценную информацию о мнениях людей и отношения. В данной статье рассматриваются модели и методы сентиментального анализа данных в казахском языке. В качестве набора данных анализа настроений были собраны данные мнений из веб-ресурсов, и их эффективность была оценена с использованием методов машинного и глубокого обучения на этих новых данных. Статья включает информацию о том, как данные собирались, как они были предварительно обработаны для оптимизации качества, а также о методах процесса оценки.

Ключевые слова: сентиментальный анализ, казахский язык, набор данных, слово, настроение

ANNOTATION

Nowadays, with the increasing use of digital content in Kazakh, such as social media posts, news, and customer testimonials, there is a growing need for automated sentiment analysis methods that can process large amounts of data and provide valuable information about people's opinions and attitudes. This article discusses the models and methods of sentimental data analysis in the Kazakh language. Opinion data from web resources was collected as a sentiment analysis dataset and its performance was evaluated using machine and deep learning methods on this new data. The article includes information on how the data was collected, how it was pre-processed to optimize quality, and methods for the evaluation process.

Key words: sentimental analysis, kazakh language, data set, word, sentiment

МАЗМҰНЫ

КІРІСПЕ

Жұмыстың өзектілігі: Қазіргі уақытта әлеуметтік желідегі жазбалар, жаңалықтар және тұтынушылардың пікірлері сияқты қазақ тіліндегі цифрлық контентті пайдаланудың артуымен үлкен көлемдегі деректерді өңдей алатын және адамдардың пікірі мен көзқарасы туралы құнды ақпарат беретін автоматтандырылған сентименталды талдау әдістеріне қажеттілік артып отыр. Қазақ тіліндегі сентименталды деректерді талдаудың моделі мен әдістерін зерттеу бірқатар себептерге байланысты өте маңызды.

Біріншіден, сентименталды талдау табиғи тілді өңдеу саласында барған сайын маңызды бола түсуде. Себебі сентименталды талдау Қазақстанда жұмыс істейтін немесе қазақстандық нарыққа шығуға ұмтылатын кәсіпорындар мен ұйымдар үшін маңызды практикалық қолданбаларға ие болуы мүмкін. Интернетте қолжетімді деректер көлемінің ұлғаюымен көптеген кәсіпорындар мен ұйымдар үлкен көлемді деректерден құнды ақпараттарды ала отырып, тұтынушылардың қалауларын, нарықтық үрдістерді түсіну және тиімді маркетингтік стратегияларды әзірлеу сияқты маңызды құралдарға ие бола алады. [1]

Екіншіден, сентименталды талдау әлеуметтану, психология және саясаттану сияқты салаларда зерттеу мақсатында пайдаланылуы мүмкін. Әлеуметтік желідегі жазбалардағы, жаңалықтардағы, мақалалардағы немесе басқа да мәтіндік деректерде айтылған деректерді талдау арқылы зерттеушілер қазақ қоғамындағы әртүрлі топтардың немесе жеке адамдардың көзқарастары, эмоциялары мен сенімдері туралы түсінікке ие бола алады.

Үшіншіден, сентименталды деректерді талдау денсаулық сақтау, қаржы және білім беруді қоса алғанда, әртүрлі салаларда да қолданылуы мүмкін. Мысалы, оны емделушілердің пікірлерін талдау және денсаулық сақтау қызметтерін жақсарту, жалған қаржылық транзакцияларды анықтау және білім беру бағдарламаларының тиімділігін бағалау үшін пайдалануға болады.

Жалпы, қазақ тіліндегі сентименталды деректерді талдаудың моделі мен әдістерін зерттеу теориялық тұрғыдан да, практикалық тұрғыдан да өзекті. Ол Қазақстанда бар деректердің орасан зор көлемін жақсырақ түсіну және пайдалану, сондай-ақ еліміздің технологиялық және ғылыми әлеуетін дамытуға ықпал ету үшін қажет.

Жұмыстың мақсаты: Сентименталды талдау әдістерін зерттей отырып, әртүрлі тәсілдерді салыстырып қазақ тіліне арналған сентименталды талдаудың барынша нақты үлгісін беру болып табылады.

Зерттеудің негізгі міндеттері:

- сентименталды деректерді талдауда қолданылатын негізгі ұғымдар мен әдістерге шолу жасау;
- веб-ресурстардан өзекті деректерді жинау, олардың негізінде машина алдын ала өңдеуді жүзеге асыру;

- жасалған үлгілер мен әдістерді нақты деректер жиынына қолдану және олардың тиімділігін бағалау;

- ұсынылған үлгілер мен әдістер бойынша озық тәжірибелерді қалай тиімді қолдану керектігі туралы ақпарат пен нұсқаулар беру.

Зерттеу пәні: Мәтінді деректер таңдау

Зерттеу нысаны: Веб-ресурстардан жиналған қазақ тілдіндегі пікірлер

Зерттеу әдістері: Аталған міндеттерді шешу үшін машиналық оқыту, терең оқыту және деректерді визуализациялау әдісі пайдаланылады

Зерттеудің ғылыми маңыздылығы:

1. Зерттеу қазақ тіліндегі сентименталды талдауды зерттеуге арналған деректер жиынтығы ретінде пайдаланылуы мүмкін қазақ тіліндегі мәтіндердің үлкен корпусын жинау және оларды алдын ала өңдеуді қамтиды.

2. Зерттеу қазақ тілі үшін арнайы қолдануға болатын тілді өңдеу құралдары мен машиналық оқыту жүйесін дамытуға үлес қоса алады. Бұл қазақ тілін пайдаланушылар үшін қолжетімді тілді өңдеу құралдарының сапасын жақсартуға көмектесуі мүмкін.

3. Жұмыс тіл зерттеулерінің маңыздылығын көрсете отырып, қазақ тілінің әртүрлі салаларда, соның ішінде академиялық ортада, бизнесте және үкіметте қолданылуына ықпал ете алады.

Жұмыстың ғылыми мәні қазақ тіліндегі табиғи тілді өңдеу саласын ілгерілету және қазақ тілін пайдаланушылар үшін тілді өңдеу құралдарының машиналық оқыту жүйесін дамытуға ықпал ету әлеуетінде жатыр.

Жұмыс апробация. Диссертациялық зерттеу жұмысы «Студенческий вестник» № 19(258) атты ғылыми журналға «Қазақ тіліндегі сентименталды деректерді талдаудың моделдері мен әдістерін зерттеу» тақырыбымен жарияланды

Жұмыстың көлемі мен құрылымы. Диссертация кіріспеден, үш бөлімнен және қорытындыдан, 63 беттік мәтін, 30 сурет, 4 кесте және пайдаланылған 9 дереккөздің тізімінен тұрады.

1 Сентименталды деректерді талдауда қолданылатын негізгі ұғымдар мен әдістерге шолу

1.1 Сентименталды талдау жайлы жалпы мәліметтер

Сентименталды талдау, сондай-ақ пікірді анықтау деп те белгілі, берілген мәтінде айтылған сезімдерді анықтауға және талдауға бағытталған табиғи тілді өңдеудің (NLP) ішкі саласы болып табылады [1]. Бұл адамдардың белгілі бір тақырыпқа, брендке, өнімге немесе қызметке қатысты пікірлерін, көзқарастарын және эмоцияларын түсіну үшін пайдалануға болатын қуатты құрал. Сентименталды талдауды нарықты зерттеу, әлеуметтік медиа мониторингі, тұтынушыларға қызмет көрсету және қоғамдық пікірді талдау сияқты неше түрлі бағыттарға қолдануға болады [2].

Сезімдерді талдау бірнеше себептерге байланысты маңызды:

- тұтынушылардың пікірлері – Цифрлық дәуірде тұтынушылар әлеуметтік медиа платформаларынан онлайн нарықтарға дейін барлық жерде өнімдер мен қызметтер туралы пікірлері мен пікірлерін қалдырады. Сентименталды талдау компанияларға тұтынушыларының көңіл-күйлері мен пікірлерін түсінуге және тиісінше өз өнімдері мен қызметтерін жақсартуға көмектеседі;

- бренд беделін басқару – Сентименталды талдауы компанияларға өз бренді туралы онлайн сұхбаттарды талдау арқылы олардың бренд беделін бақылауға мүмкіндік береді. Бұл олардың брендіне қатысты ықтимал проблемаларды және теріс көзқарастарды анықтауға және оларды шешу үшін тиісті шаралар қабылдауға көмектеседі;

- нарықты зерттеу – Сентименталды талдау нарықты зерттеудің пайдалы құралы болып табылады, өйткені ол тұтынушылардың қалауы, мінез-құлқы және пікірлері туралы түсінік береді. Бұл компанияларға өздерінің мақсатты аудиториясын түсінуге және маркетингтік стратегияларын жақсарту үшін деректерге негізделген шешімдер қабылдауға көмектеседі;

- акция бағасын болжау – Сентименталды талдау белгілі бір компанияларға немесе салаларға қатысты қоғамдық пікірді талдау арқылы акциялардың бағасын болжау үшін пайдаланылуы мүмкін. Бұл ақпаратты инвесторлар акцияларды сатып алу және сату туралы негізделген шешім қабылдау үшін пайдалана алады.

- бәсекелестік талдау – Талдауды тұтынушылардың көңіл-күйі мен бәсекелес брендтерге қатысты пікірлерін талдау арқылы бәсекелестік талдау үшін де пайдалануға болады. Ол компанияларға бәсекелестерінен ерекшеленетін салаларды анықтауға және өнімдерін немесе қызметтерін жақсартуға көмектеседі.

- саясатта қолдану – Талдау саяси кандидаттарға, партияларға және мәселелерге қатысты қоғамдық пікірді бақылау үшін пайдаланылуы мүмкін. Бұл ақпаратты саяси науқандар өз хабарламаларын нақтылау және бәсекелестік артықшылыққа қол жеткізу үшін пайдалана алады;

- тәуекелдерді басқару – Сентименталды талдау компанияларға белгілі бір тақырыпқа немесе өнімге теріс қатынасты талдау арқылы ықтимал тәуекелдер мен мәселелерді анықтауға көмектеседі. Бұл ақпаратты ықтимал мәселелерді белсенді түрде шешу және тәуекелдерді азайту үшін пайдалануға болады.

Сентименталды талдау компаниялар үшін тұтынушылардың көңіл-күйі мен пікірлері туралы түсінік алу, олардың бренд беделін қадағалау, нарықтық зерттеулер жүргізу, бәсекелестік талдау жүргізу және тәуекелді басқару үшін маңызды. Бұл компаниялар үшін деректерге негізделген шешімдер қабылдауға және өнімдерін, қызметтерін және тұтынушылармен өзара әрекеттесуді жақсартуға арналған қуатты құрал.

Сезімдерді талдаудың негізгі алғышарттары мәтіндерді олар білдіретін сезімдерге байланысты оң, теріс немесе бейтарап деп жіктеуге болады. Сезімдерді сөздер, сөз тіркестері, эмотикондар, тіпті дауыс ырғағы сияқты түрлі тілдік құралдар арқылы білдіруге болады. Мысалы, «Мен бұл өнімді жақсы көремін» сөйлемі жағымды көзқарасты білдірсе, «Маған бұл өнім ұнамады» жағымсыз сезімді білдіреді. Дегенмен, мәтінде айтылған көңіл-күй әрқашан қарапайым емес және контекст, сарказм, ирония және мәдени нюанстар сияқты әртүрлі факторларға байланысты болуы мүмкін.

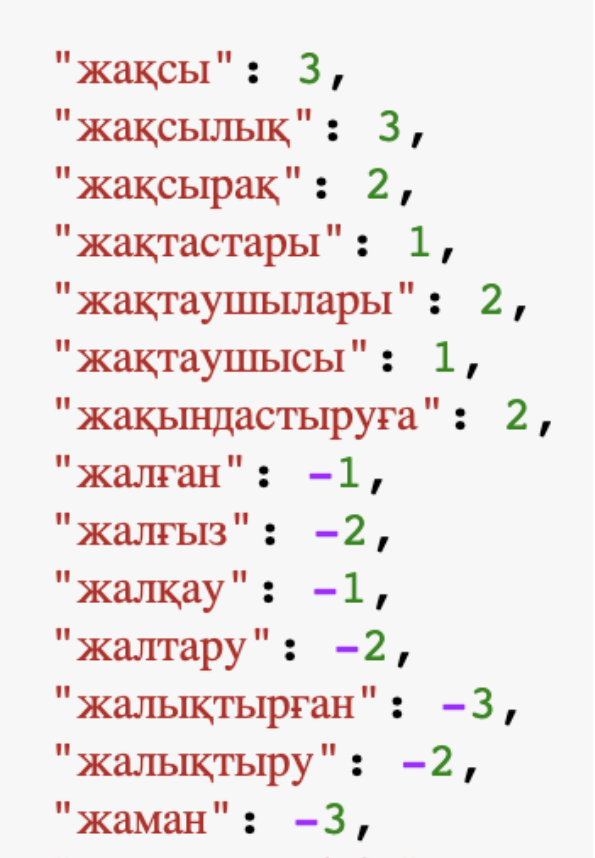
1.2 Лексиконға негізделген әдістер

Бұл мәселелерді шешу үшін сезімді талдау әртүрлі әдістерге, соның ішінде сөздікке негізделген әдістерге, машиналық оқытуға және терең оқытуға сүйенеді. Лексиконға негізделген әдістер алдын ала жасалған сөздіктерді немесе сөздерді және оларға сәйкес сезім ұпайларын қамтитын лексикондарды пайдаланады. Мысалы, «махаббат» сөзі оң мәнге ие, ал «жек көру» сөзі теріс мәнге ие. Содан кейін мәтіннің көңіл-күй бағасы оның құрамдас сөздерінің ұпайларының қосындысы немесе орташа мәні негізінде есептеледі. Лексикаға негізделген әдістер салыстырмалы түрде қарапайым және жылдам, бірақ табиғи тілдің нюанстары мен күрделілігін түсіруге жарамсыз болуы мүмкін.

Сөздікке негізделген әдістер көңіл-күйді талдаудың танымал тәсілі және жиі қолданылатын лексиконның бірі AFINN (Ағылшын сөздеріне арналған аффективті нормалар). AFINN - -5 (теріс) мен +5 (оң) аралығындағы ағылшын сөздері үшін алдын ала есептелген көңіл-күй ұпайларының тізімі. [3]

AFINN-де әрбір сөзге жағымды немесе жағымсыз көңіл-күймен байланысы негізінде көңіл-күй бағасы беріледі. Ұпайларды контексте сөздердің сезімін бағалайтын адам комментаторлары қолмен анықтайды. Мысалы, «махаббат» және «бақытты» сияқты сөздер жоғары оң бағаға ие болса, «жек көру» және «қайғылы» сияқты сөздер теріс бағаға ие болады. Бейтарап сөздер әдетте нөлге жақын ұпайға ие. AFINN көмегімен көңіл-күйді талдауды орындау үшін берілген мәтіндегі әрбір сөз үшін сезім баллы есептеледі. Одан кейін мәтін үшін жалпы көңіл-күй ұпайын алу үшін жеке ұпайлар жинақталады. Оң

ұпайлар жағымды көңіл-күйді, теріс ұпайлар жағымсыз көңіл-күйді, ал нөлге жақын ұпайлар бейтарап көңіл-күйді көрсетеді (сурет 1.2.1).



Сурет 1.2.1. Қазақшаланған AFINN-165 сөздігінен үзінді

AFINN қарапайым және оңай қол жетімді сөздік болғанымен, оның белгілі бір шектеулері бар. Көңіл-күй ұпайлары сөз деңгейінде тағайындалғандықтан, олар мәтінмен көрсетілген көңіл-күйді толық көрсетпеуі мүмкін. Мәтінмәндік нюанстар, сарказм және бейнелі тіл AFINN сияқты сөздікке негізделген әдістерге қиындық тудыруы мүмкін. Сонымен қатар, AFINN сезімді түсіндіруге әсер етуі мүмкін сөз тәртібін, сөйлем құрылымын немесе терістеуді ескермейді.

1.3 Машиналық оқытуға негізделген әдістер

Екінші жағынан, машиналық оқытуға негізделген әдістер аннотацияланған мәтіндердің үлкен деректер жиынтығынан көңіл-күй үлгілерін анықтауды үйренетін алгоритмдерге сүйенеді. Алгоритмдер таңбаланған деректер жиынында оқытылады, мұнда әрбір мәтін сәйкес сезім белгісімен (оң, теріс немесе бейтарап) аннотацияланады. Алгоритм деректердегі үлгілерді тануды үйренеді және оларды жаңа мәтіндерді жіктеу үшін пайдаланады.

Машиналық оқытуға негізделген әдістер сөздікке негізделген әдістерге қарағанда икемді және дәлірек, бірақ олар көп таңбаланған деректерді қажет етеді және көрінбейтін деректерге жақсы жалпыланбауы мүмкін [4].

Мысал ретінде логистикалық регрессияны қарастырсақ болады:

Логистикалық регрессия көңіл-күйді талдауда жиі қолданылатын алгоритм болып табылады. Бұл нәтижені анықтайтын бір немесе бірнеше тәуелсіз айнымалылар бар деректер жиынтығын талдаудың статистикалық әдісі. Сезімдерді талдау контекстінде тәуелсіз айнымалылар мәтіннен алынған мүмкіндіктер болып табылады, ал нәтиже - көңіл-күй белгісі (оң, теріс немесе бейтарап).

Логистикалық регрессия алгоритмі мәтіннен алынған мүмкіндіктер жиынтығын ескере отырып, белгілі бір сезім белгісінің ықтималдығын бағалау арқылы жұмыс істейді. Алгоритм логистикалық функцияны (сонымен қатар сигмоидтық функция деп те аталады) сызықтық теңдеудің шығысын оң көңіл-күй белгісінің ықтималдығын білдіретін 0 мен 1 арасындағы мәнге салыстыру үшін пайдаланады. Теріс көңіл-күй белгісінің ықтималдығы оң көңіл-күй белгісінің ықтималдығынан 1-ге тең болады [5].

Логистикалық регрессия үлгісін үйрету үшін алгоритм мәтін мен сезім белгілерінің белгіленген деректер жиынын пайдаланады. Мүмкіндіктер мәтіннен жоғарыда атап өтілген TF-IDF векторизациясы әдісін пайдалана отырып шығарылады және сезім белгілері екілік (оң немесе теріс) болып табылады [6]. Содан кейін алгоритм мәтінде байқалған сезім белгілерінің ықтималдығын барынша арттыратын мүмкіндіктердің салмақтарын үйренеді. Модель үйретілгеннен кейін оны жаңа мәтіннің сезім белгісін болжау үшін пайдалануға болады.

Логистикалық регрессияның бір артықшылығы оның түсіндіруге оңай қарапайым алгоритм болып табылады. Модел үйренген салмақтарды қандай мүмкіндіктер көңіл-күй белгісінің ең болжамды екенін анықтау үшін тексеруге болады. Логистикалық регрессия сонымен қатар салыстырмалы түрде шағын деректер жиынында жақсы жұмыс істейді, бұл оны белгіленген деректер шектелген сезімдерді талдау тапсырмалары үшін жақсы таңдау жасайды [7].

Multinomial Naive Bayes - мәтінді жіктеу үшін сезімді талдауда қолданылатын танымал машиналық оқыту алгоритмі. Бұл мәтінді алдын ала анықталған категорияларға жіктеу үшін Байес теоремасын қолданатын ықтималдық алгоритмі.

Бұл алгоритмде мәтіндегі әрбір сөз ерекшелік ретінде қарастырылып, оның жиілігі есептеледі. Содан кейін әрбір сыныпқа жататын әрбір сөздің ықтималдығы (оң немесе теріс сезім) Байес теоремасы арқылы есептеледі. Соңғы болжам әр сыныптың ықтималдығын салыстыру және ықтималдығы жоғарысын таңдау арқылы жасалады [8].

Multinomial Naive Bayes құжаттағы сөздердің жиілігі көпмүшелік таралудан кейін келеді деп болжайды. Сондай-ақ, әрбір сөздің кездесуі

құжаттағы басқа сөздердің кездесуіне тәуелсіз деп есептейді. Бұл болжам нақты әлемдегі мәтіндік деректерде әрқашан дұрыс бола бермеуі мүмкін, бірақ Multinomial Naive Bayes әлі де тәжірибеде жақсы жұмыс істейді және сезімді талдауда кеңінен қолданылады.

Бұл алгоритмның бір артықшылығы оның үлкен деректер жиынын өңдеудегі қарапайымдылығы мен тиімділігі болып табылады. Дәл болжамдар жасау үшін жаттығу деректерінің аз ғана көлемін қажет етеді және оның орындалу уақыты өте үлкен деректер жиындары үшін де жылдам.

Жұмыста бұл алгоритмді Python арқылы жүзеге асыру үшін біз scikit-learn кітапханасын қолданамыз.

Дегенмен, Multinomial Naive Bayes сирек сөздермен жұмыс істегенде немесе жаттығу деректері өте теңгерімсіз болған кезде жақсы жұмыс істемеуі мүмкін. Мұндай жағдайларда логистикалық регрессия немесе қолдау векторлық машиналары сияқты басқа алгоритмдер қолайлырақ болуы мүмкін. Тұтастай алғанда, Multinomial Naive Bayes көңіл-күйді талдауда пайдалы және кеңінен қолданылатын алгоритм болып табылады, бірақ оны тәжірибеде пайдалану кезінде оның шектеулері мен ықтимал кемшіліктерін ескеру маңызды [9].

LinearSVC – классификациялық есептерді шығару үшін қолданылатын машиналық оқыту алгоритмі. Бұл сызықтық бөлінетін жіктеу мәселелерін шешуге арналған Support Vector Machine (SVM) алгоритмінің нұсқасы. LinearSVC табиғи тілді өңдеу тапсырмаларында, соның ішінде сезімді талдауда кеңінен қолданылады.

LinearSVC – кіріс деректерді әртүрлі класстарға бөлу үшін сызықтық гипержазықтықты пайдаланатын екілік жіктеу алгоритмі. Гипержазықтық - бұл деректер нүктелерін ерекшеліктеріне қарай тиісті сыныптарға бөлетін шешім шекарасы. Алгоритм оқу деректерінен классификация қатесін азайтатын оңтайлы гипержазықтықты табу арқылы үйренеді.

LinearSVC екі класс арасындағы маржаны барынша арттыратын оңтайлы гипержазықтықты табу арқылы жұмыс істейді [10]. Маржа гипержазықтық пен әрбір сыныптағы ең жақын деректер нүктелерінің арасындағы қашықтық ретінде анықталады. Содан кейін алгоритм кіріс деректер мен гипержазықтық арасындағы нүкте туындысын есептеу арқылы болжам жасайды. Нәтиже оң болса, деректер нүктесі бір класс ретінде жіктеледі, ал теріс болса, ол басқа класс ретінде жіктеледі.

Жалпы алғанда, LinearSVC моделі көңіл-күйді талдау тапсырмалары үшін қуатты алгоритм болып табылады және оның әртүрлі деректер жиындарында жоғары дәлдікке қол жеткізуі көрсетілген. Дегенмен, кез келген машиналық оқыту алгоритмі сияқты, оның өнімділігі оқу деректерінің сапасы мен өлшеміне, сондай-ақ гиперпараметрлерді таңдауға байланысты.

XGBoost (Extreme Gradient Boosting) - әртүрлі салаларда, соның ішінде көңіл-күйді талдауда қолданылатын танымал машиналық оқыту алгоритмі. Бұл болжам жасау үшін шешім ағаштарының ансамблін жасайтын күшейту алгоритмінің бір түрі [11].

XGBoost градиентті күшейту алгоритміне негізделген және өнімділік пен дәлдікті жақсарту үшін бірнеше негізгі мүмкіндіктерді қосады. Негізгі мүмкіндіктердің бірі - жаттығу кезінде модельді оңтайландыру үшін жетілдірілген жоғалту функциясын пайдалану. Сонымен қатар, XGBoost машинаны оқытуда жиі кездесетін мәселе болып табылатын шамадан тыс орнатудың алдын алу үшін регуляризация деп аталатын әдісті пайдаланады.

Сезімдерді талдау контекстінде XGBoost мәтінді оң, теріс немесе бейтарап деп жіктеу үшін пайдаланылуы мүмкін. Сезімдерді талдау үшін XGBoost пайдалану үшін алдымен мәтіндік деректерді алдын ала өңдеп, оны сандық мүмкіндіктерге түрлендіру керек. Мұны TF-IDF Vectorizer көмегімен токенизация, сөзді жоюды тоқтату және мүмкіндіктерді шығару сияқты әдістерді қолдану арқылы жасауға болады.

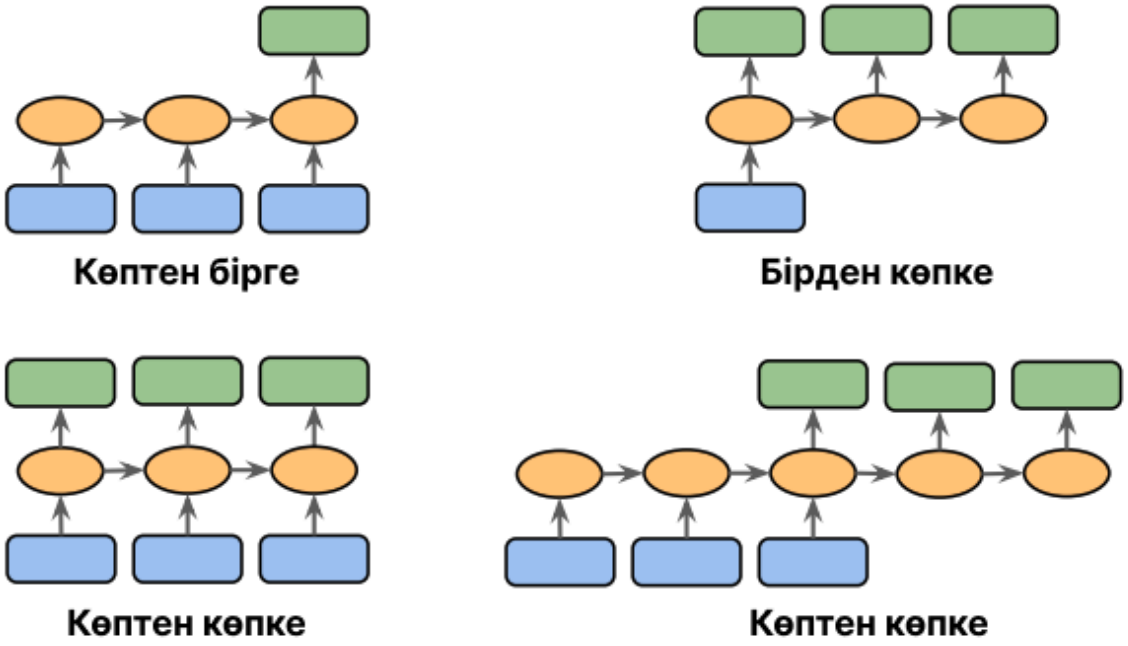
Деректерді алдын ала өңдегеннен кейін біз XGBoost үлгісін мәтіннің белгіленген деректер жиынына және олардың сәйкес сезім белгілеріне үйрете аламыз. Жаттығу кезінде XGBoost шешім ағаштарының ансамблін жасайды, мұнда әрбір ағаш деректердің басқа жиынында оқытылады. Соңғы болжам барлық жеке шешім ағаштарының болжамдарын біріктіру арқылы жасалады.

XGBoost басқа машиналық оқыту алгоритмдерімен салыстырғанда сезімді талдауда бірнеше артықшылықтарға ие. Ол өзінің жоғары дәлдігі мен жылдамдығымен танымал және миллиондаған жолдар мен мыңдаған бағандары бар үлкен деректер жиынын өңдей алады. Оған қоса, XGBoost-та модель үшін ең маңызды мүмкіндіктерді автоматты түрде таңдай алатын, шамадан тыс орнату қаупін азайтатын кірістірілген мүмкіндікті таңдау әдісі бар.

1.4 Терең оқытуға негізделген әдістер

Нейрондық желілер сияқты терең оқыту әдістері көңіл-күйді талдауда перспективалы нәтижелер көрсетті. Олар мәтіндердің күрделі көріністерін зерттей алады және көңіл-күйдің нюанстары мен контекстке сезімтал табиғатын қабылдай алады. Терең оқытуға негізделген әдістер машиналық оқытуға негізделген әдістерге қарағанда көбірек деректерді қажет етеді, бірақ олар сезімді талдау тапсырмаларында заманауи нәтижелерге қол жеткізе алады. [12]

Тізбекті модельдеу сентименталды талдауда шешуші рөл атқарады, өйткені ол мәтіндегі, сөз тәртібі мен контекстті ескере отырып, көңіл-күйді талдауға мүмкіндік береді [13]. Сентименталды талдау сөйлем, шолу немесе әлеуметтік желідегі жазба сияқты мәтін бөлігінде көрсетілген негізгі сезімді немесе эмоцияны анықтауға бағытталған. Тізбекті модельдеу әдістерін қолдана отырып, көңіл-күйді талдау сөздер арасындағы күрделі қарым-қатынастарды түсіріп, мәтін арқылы жеткізілетін сезімді дәл түсіндіре алады (сурет 1.4.1).



Сурет 1.4.1. Тізбекті модельдеудің әртүрлі категориялары

Жоғарыдағы суретте бейнеленген кірістер мен шығыстар арасындағы қатынастардың әртүрлі категорияларын толығырақ қарастырайық.

- Көптен бірге (many-to-one): Кіріс реттілік болып табылады, бірақ шығыс реттік емес, бекітілген өлшемді вектор немесе скаляр. Мысалы, сезімді талдауда кіріс мәтіндік деректер (мысалы, фильмді шолу) және шығыс - сынып белгісі (мысалы, рецензентке фильм ұнаған-ұнамағанын көрсететін белгі);

- Бірден көпке (one-to-many): Кіріс стандартты пішімде, реттілік емес, бірақ шығыс реттілік болып табылады. Бұл санаттың мысалы ретінде суреттің субтитрлері болып табылады - кіріс сурет болып табылады және шығыс сол кескіннің мазмұнын қорытындылайтын сөйлем.;

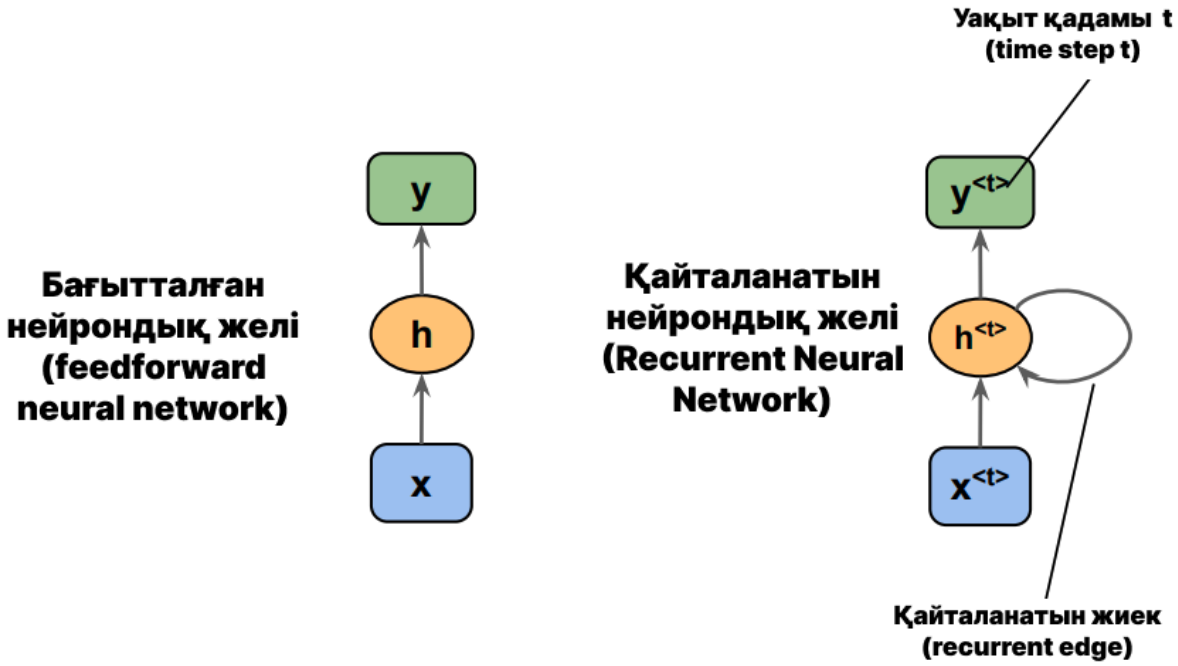
- Көптен көпке (many-to-many): Кіріс және шығыс массивтерінің екеуі де реттілік болып табылады. Бұл санатты кіріс пен шығыс синхрондалғанына қарай одан әрі бөлуге болады. Синхрондалған көптен көпке үлгілеу мәселесінің мысалы ретінде бейненің әрбір кадры белгіленетін бейне классификациясы болып табылады. Кейінге қалдырылған көптен көпке үлгілеу тапсырмасының мысалы бір тілді басқа тілге аудару болады. Мысалы, орыс тіліндегі сөйлемді қазақ тіліне аудару үшін оны машина оқуы және өңдеуі керек [15].

Қайталанатын нейрондық желі (Recurrent Neural Network, RNN) – жүйелі деректерді өңдеу үшін арнайы әзірленген нейрондық желі архитектурасының түрі. Бұл уақыт қатарларын талдау, табиғи тілді өңдеу, сөйлеуді тану және қолжазбаны тану сияқты ретті өңдеу тапсырмаларында әсіресе тиімді [16].

Деректерді кірістен шығысқа бір өтуде өңдейтін алға бағытталған нейрондық желілерден айырмашылығы, RNNs алдыңғы кірістер туралы ақпаратты сақтауға және оны болашақ кірістерді өңдеуге әсер ету үшін

пайдалануға мүмкіндік беретін кері байланыс механизміне ие. Бұл кері байланыс механизмі RNN-ге сериялық деректерде бар уақытша тәуелділіктер мен үлгілерді түсіруге мүмкіндік береді [17].

RNN (сурет 1.4.2) негізгі құрамдас бөлігі ақпаратты бір қадамнан келесі қадамға ретімен беруге мүмкіндік беретін қайталанатын байланыс болып табылады. Әрбір уақыт қадамында RNN кірісті қабылдайды және оның жады ретінде әрекет ететін жасырын күйді сақтай отырып, шығыс шығарады. Жасырын күй желі көрген және әр қадам сайын жаңартылатын өткен ақпараттың қысқаша мазмұны ретінде қызмет етеді.



Сурет 1.4.2. Қайталанатын нейрондық желі

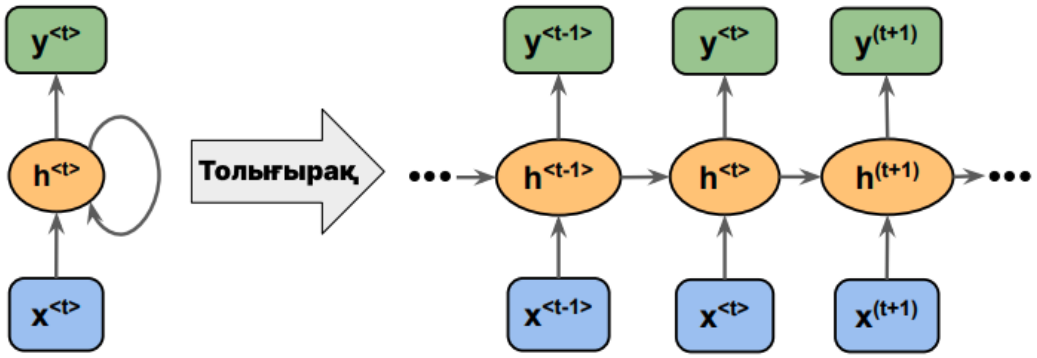
Кіріс қабаты (x), жасырын қабат (h) және шығыс қабаты (o) бар векторлар деп есептейік.

Стандартты алға жіберу желісінде ақпарат кірістен жасырын деңгейге, содан кейін жасырын қабаттан шығыс деңгейге өтеді. Екінші жағынан, RNN-де жасырын қабат ағымдағы уақыт қадамының кіріс деңгейінен де, алдыңғы уақыт қадамының жасырын қабатынан да кірісті алады.

Жасырын қабаттағы іргелес уақыт қадамдарындағы ақпарат ағыны желіде өткен оқиғалардың жадына ие болуына мүмкіндік береді. Бұл ақпарат ағыны әдетте цикл ретінде көрсетіледі, сонымен қатар графикалық белгілердің қайталанатын жиегі ретінде белгілі, бұл жалпы RNN архитектурасы өз атын осында алады.

Көп қабатты перцептрондар сияқты, RNN бірнеше жасырын қабаттардан тұруы мүмкін. Бір жасырын қабаты бар RNN-ге бір қабатты RNN ретінде

сілтеме жасау әдеттегі жағдай екенін ескеріңіз, оны Adaline немесе логистикалық регрессия сияқты жасырын қабаты жоқ бір қабатты NN-мен шатастырмау керек. Төмендегі суретте (сурет 1.4.3) бір жасырын қабаты бар RNN көрсетілген:



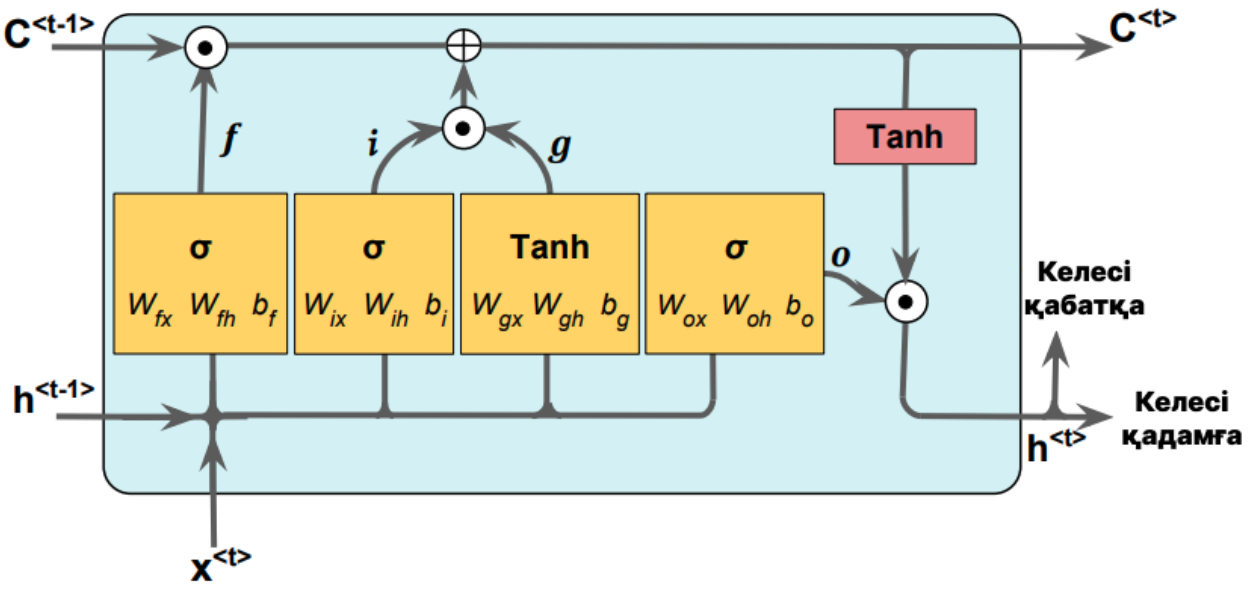
Сурет 1.4.3. Қайталанатын нейрондық желі

RNN архитектурасын және ақпарат ағынын тексеру үшін алдыңғы суретте көруге болатын қайталанатын жиегі бар ықшам көрініс ашуға болады.

Белгілі болғандай, стандартты NN-дегі әрбір жасырын блок тек бір кіріс алады - кіріс қабатымен байланысты таза алдын ала белсендіру. Керісінше, RNN-дегі әрбір жасырын бірлік кірістің екі нақты жиынын алады — кіріс деңгейінен алдын ала белсендіру және алдыңғы уақыт қадамынан бірдей жасырын қабатты белсендіру, $t - 1$.

Бірінші рет қадамында, $t = 0$, жасырын бірліктер нөлдерге немесе шағын кездейсоқ мәндерге инициализацияланады. Содан кейін, $t > 0$ болатын уақыт қадамында жасырын бірліктер ағымдағы уақытта деректер нүктесінен өз енгізуін алады, $x^{(t)}$ және $t - 1$ жасырын бірліктердің алдыңғы мәндері $x^{(t-1)}$ ретінде көрсетілген [18].

Long Short-Term Memory (LSTM) табиғи тілді өңдеу тапсырмаларында, соның ішінде сентименталды талдауда жиі қолданылатын нейрондық желі архитектурасының түрі болып табылады. Дәстүрлі нейрондық желілерден айырмашылығы, LSTM желілері ұзақ мерзімді тәуелділіктерді өңдеуге арналған және дәйекті деректерді тиімді модельдей алады, бұл оларды мәтіндік деректерді талдау үшін қолайлы етеді. Ол жад блоктарын қамти отырып, олардың әрқайсысында кіріс және шығыс қақпасы бар. Олар ұяшықтардың (сурет 1.4.4) кірісі мен белсендірілуін өлшеу арқылы желінің кірісі мен шығысын басқарады.



Сурет 1.4.4. LSTM ұяшығы

LSTM ұяшығындағы қақпалардың үш түрі бар, олар ұмытылатын қақпалар, кіру қақпалары және шығу қақпалары ретінде белгілі [19].

Ұмыту қақпасы (f_t) жад ұяшығына шексіз өсусіз ұяшық күйін қалпына келтіруге мүмкіндік береді. Ұмыту қақпасы (сурет 1.4.5) қандай ақпарат арқылы өтуге болатынын және қандай ақпаратты жасыруға болатынын шешеді.

$$f_t = \sigma(W_{xf}x^{(t)} + W_{hf}h^{(t-1)} + b_f) \quad (1.1)$$

f_t – ұмыту қақпасының мәні;

σ – 0 мен 1 арасындағы кіріс мәндерін кішірейтетін сигма тәрізді белсендіру функциясы;

W_{xf} – ұмыту қақпасымен және $x^{(t)}$ кірісімен байланысты салмақ матрицасы;

$x^{(t)}$ – ағымдағы кіріс;

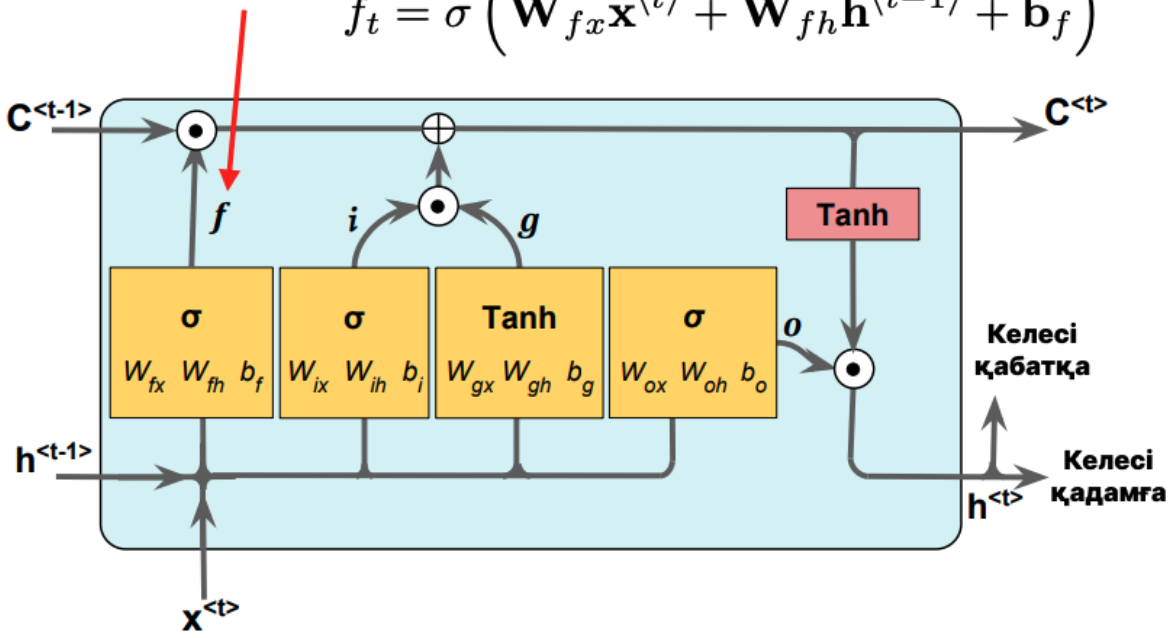
W_{hf} – ұмыту қақпасымен және алдыңғы жасырын күймен байланысты салмақ матрицасы;

$h^{(t-1)}$ – алдыңғы жасырын күй;

b_f – ұмыту қақпасымен байланысты бейтарап термині;

Ұмыту қақпасы (forget gate) қандай ақпараттың есте қалғанын, қайсысы ұмытылғанын бақылайды; ұяшық күйін қалпына келтіре алады

$$f_t = \sigma\left(\mathbf{W}_{fx}\mathbf{x}^{\langle t\rangle} + \mathbf{W}_{fh}\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_f\right)$$



Сурет 1.4.5. LSTM ұмыту қақпасы

Кіріс қақпасы (i_t) және кіріс түйіні (g_t) (сурет 1.4.6) ұяшық күйін жаңартуға жауапты. Олар келесідей есептеледі:

$$i_t = \sigma(W_{xi}x^{(t)} + W_{hi}h^{(t-1)} + b_i) \qquad (1.2)$$

i_t – t уақыт қадамындағы кіріс қақпасы;

σ – 0 мен 1 арасындағы кіріс мәндерін кішірейтетін сигма тәрізді белсендіру функциясы;

W_{xi} – кіріс қақпасымен және $x^{(t)}$ кірісімен байланысты салмақ матрицасы;

$x^{(t)}$ – ағымдағы кіріс;

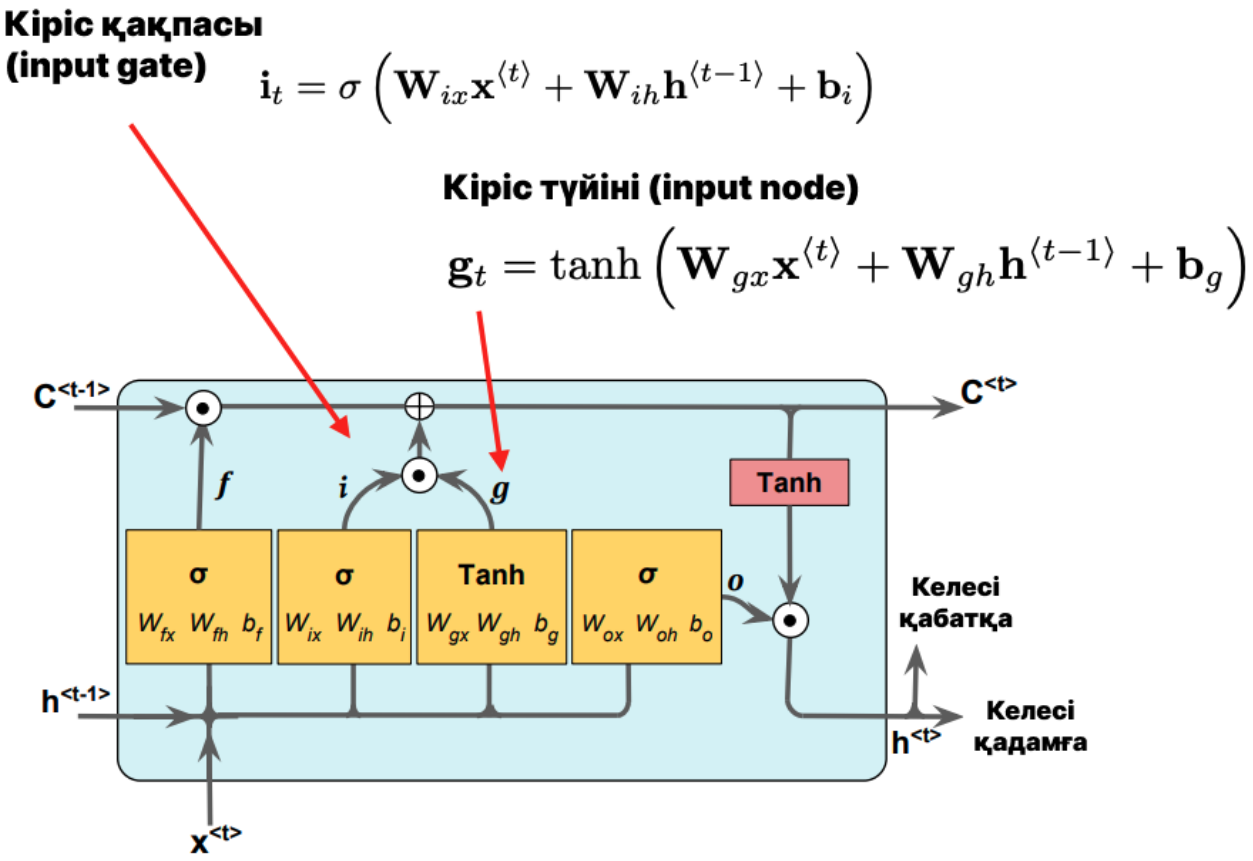
W_{hi} – кіріс қақпасымен және алдыңғы жасырын күймен байланысты салмақ матрицасы;

$h^{(t-1)}$ – алдыңғы жасырын күй;

b_i – кіріс қақпасымен байланысты бейтарап термині;

$$g_t = tanh(W_{xc}x^{(t)} + W_{hc}h^{(t-1)} + b_g) \quad (1.3)$$

g_t – t уақыт қадамындағы үміткер мәндер (кіріс түйіні);
$tanh$ – гиперболалық тангенс белсендіру функциясы;
W_{xc} – үміткер мәндерімен және x(t) кірісімен байланысты салмақ матрицасы;
$x^{(t)}$ – ағымдағы кіріс;
W_{hc} – үміткер мәндерімен және алдыңғы жасырын күймен байланысты салмақ матрицасы;
$h^{(t-1)}$ – алдыңғы жасырын күй;
b_g – үміткер мәндерімен байланысты бейтарап термині;

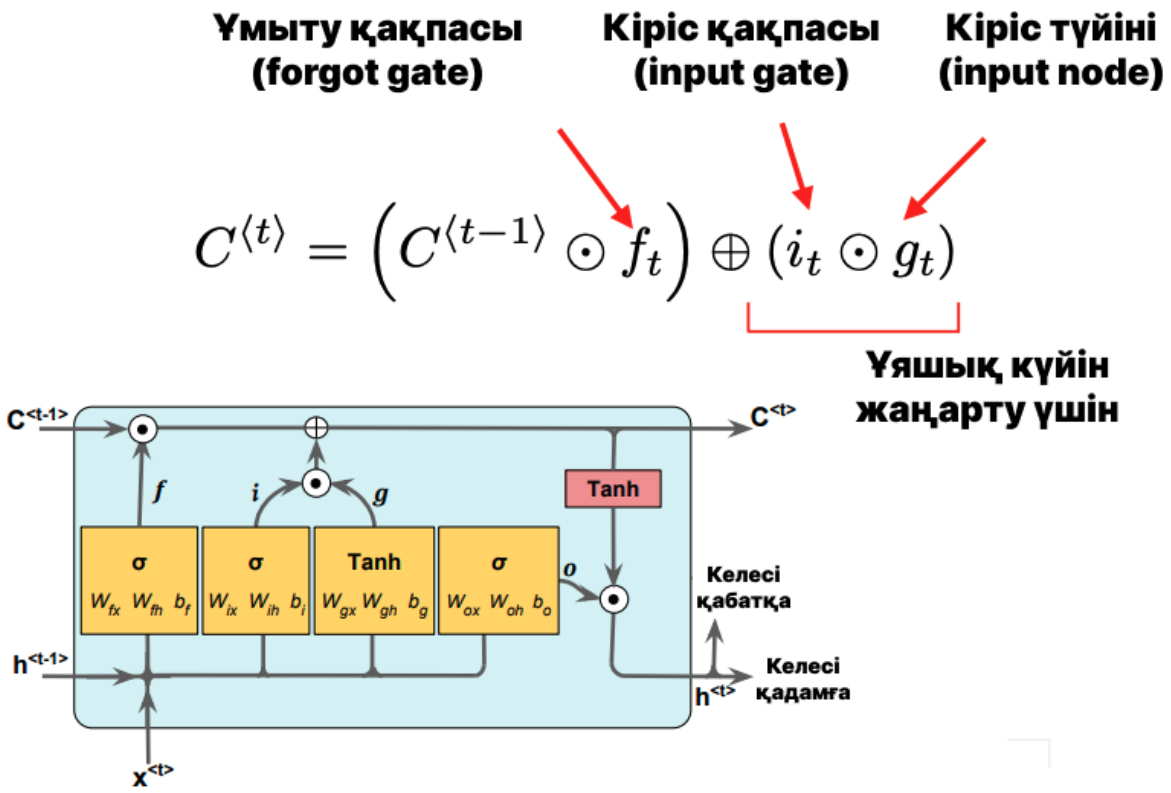


Сурет 1.4.6. LSTM кіріс қақпасы және кіріс түйіні

Ал t уақытындағы ұяшық күйі келесідей есептеледі:

$$C^{(t)} = (C^{(t-1)} \odot f_t) \oplus (i_t \odot g_t) \quad (1.4)$$

$C^{(t)}$ – t уақыт қадамындағы ұяшық күйі;
$C^{(t-1)}$ – алдыңғы ұяшық күйі;
$\odot$ – элементтік көбейту;
f_t – t уақыт қадамында ұмыту қақпасы;
$\oplus$ – элементтік қосу;
i_t – t уақыт қадамындағы кіріс қақпасы;
g_t – t уақыт қадамындағы үміткер мәндер;



Сурет 1.4.7. LSTM ұяшық күйі

Шығару қақпасы (O_t) жасырын бірліктердің мәндерін жаңарту жолын (сурет 1.4.8) шешеді:

$$O_t = \sigma(W_{xo}x^{(t)} + W_{ho}h^{(t-1)} + b_o) \quad (1.5)$$

O_t – t уақыт қадамындағы шығыс қақпасы;

σ – 0 мен 1 арасындағы кіріс мәндерін кішірейтетін сигма тәрізді белсендіру функциясы;

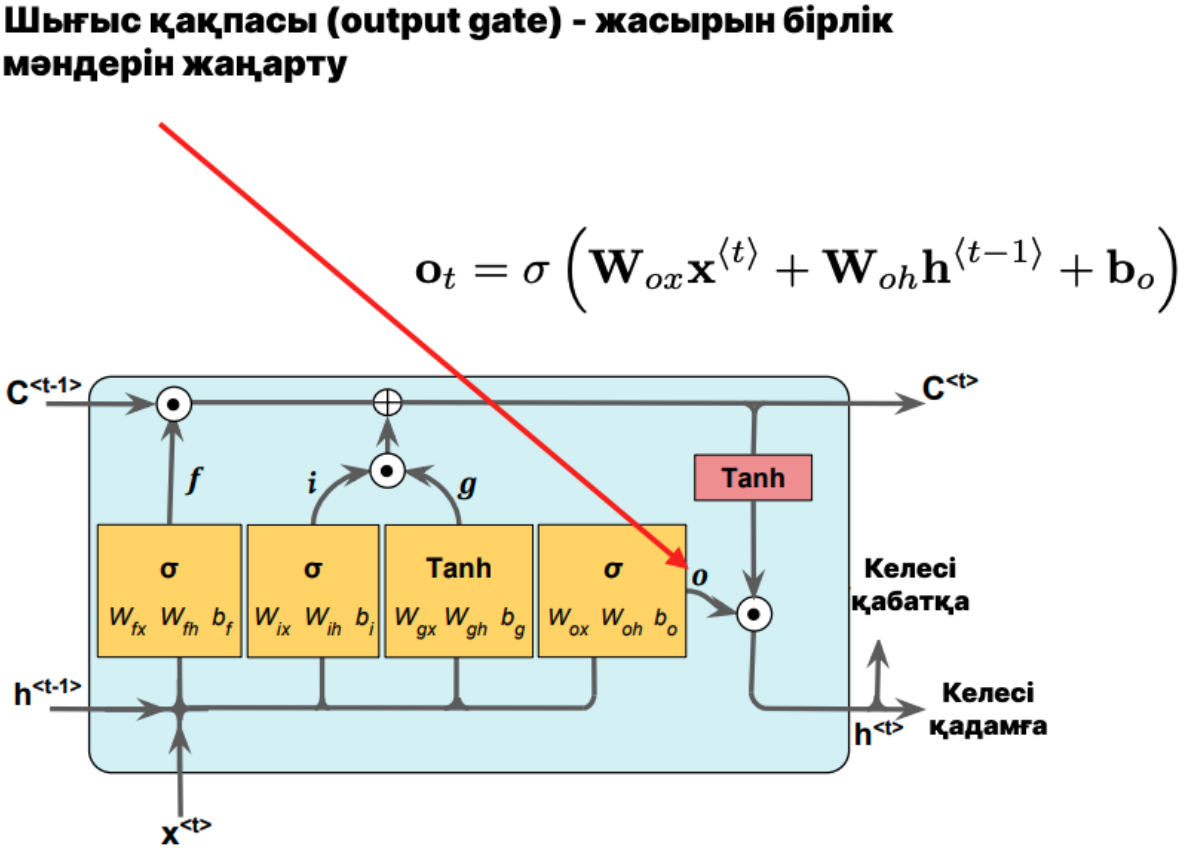
W_{xo} – шығыс қақпасымен және $x^{(t)}$ кірісімен байланысты салмақ матрицасы;

$x^{(t)}$ – ағымдағы кіріс;

W_{ho} – шығыс қақпасымен және алдыңғы жасырын күймен байланысты салмақ матрицасы;

$h^{(t-1)}$ – алдыңғы жасырын күй;

b_o – шығыс қақпасымен байланысты бейтарап термині;



Сурет 1.4.8. LSTM шығыс қақпасы

Осыны ескере отырып, ағымдағы уақыт қадамындағы жасырын бірліктер келесідей есептеледі:

$$h^t = o_t \odot tanh(C^{(t)}) \quad (1.6)$$

h_t – t уақыт қадамындағы жасырын күй;

o_t – t уақыт қадамындағы шығыс қақпасы;

$\oplus$ – элементтік қосу;

$tanh$ – гиперболалық тангенс белсендіру функциясы;

$C^{(t)}$ – t уақыт қадамындағы ұяшық күйі;

Бөлімге қорытынды

Қорытындылай келе бұл бөлімде зерттеу алды жұмысы ретінде тақырыпты ашу үшін жалпылама сипаттамалар мен статистикалық мәліметтер келтірілді. Нақтырақ айтар болсақ сентименталды талдау жайлы жалпы мәліметтер және қазақ тіліндегі мәтіндерге сентименті талдау әдістері қарастырылды.

Лексиконға негізделген әдіс ретінде AFINN, машиналық оқыту әдістері ретінде логистикалық регрессия, Multinomial Naive Bayes, Сызықтық SVM, XGBoost қарастырылды. Ал терең оқыту әдісі ретінде Long short-term memory (LSTM), LSTM жақсарту үшін LSTM-pack sequence оқыту әдістеріне шолу жасалды.

2 Веб-ресурстардан өзекті деректерді жинау, олардың негізінде машина алдын ала өңдеуді жүзеге асыру

2.1 Өзекті пікірлерді жинау

Қазақ тілінде пікірлерден тұратын деректер жинағын құру қазақ тіліндегі сентименталды деректерді талдау моделі мен әдістерін зерттеу жолындағы маңызды қадам.

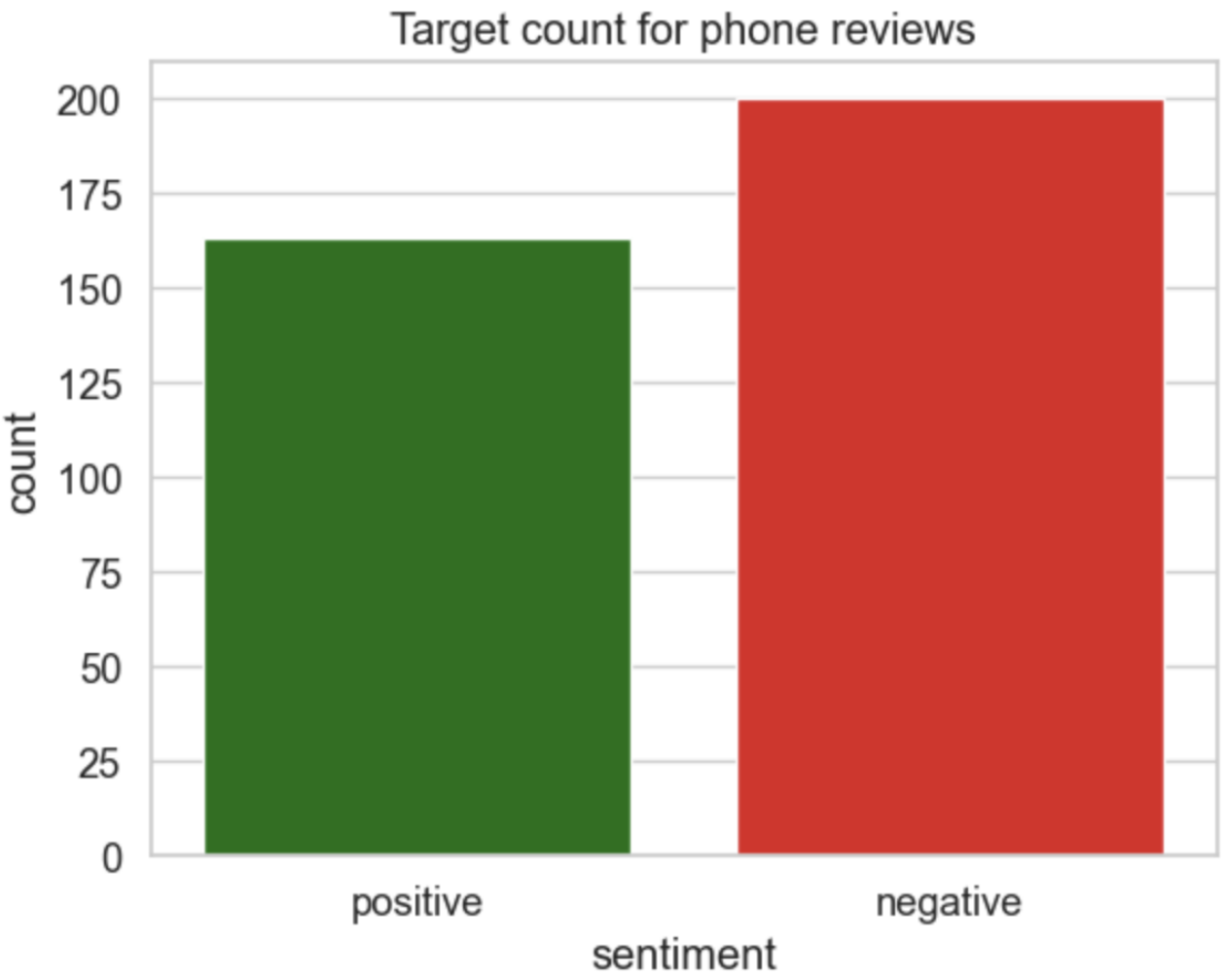
Қазақ тілінде пікір деректер жинағын құрудың бір жолы электронды коммерция веб-сайттары, әлеуметтік медиа платформалары және басқа да веб-сайттар сияқты әртүрлі онлайн көздерден пікірлерді жинау болып табылады. Деректер жинағының сапасын қамтамасыз ету үшін мәтін дәл және дәйекті болуы керек. Пікірлер әртүрлі электрондық коммерция веб-сайттарынан, соның ішінде Kaspi Магазин және 2GIS сайттарынан жиналды.

Пікірлер жиналғаннан кейін олар мәтінде айтылған жалпы пікір негізінде оң және теріс көңіл-күйге (sentiment) жіктелуі керек. Деректер жиынында оң немесе бейтарап деп қолмен белгіленген жалпы 363 пікірлер бар.

Пікірлердің қазақ тілінде болуын қамтамасыз ету үшін барлық қазақша емес пікір деректер жинағынан жойылды.

Соңғы деректер жинағы екі бөлікке бөлінді: оқу жинағы және сынақ жинағы. Тренинг жинағы 290 пікірден тұрады, ал тестілеу жиынтығы 73 пікірден тұрады (сурет 2.1.1). Жаттығу жиыны сентименталды талдау үлгісін үйрету үшін пайдаланылды, ал тестілеу жиынтығы модельдің дәлдігін бағалау үшін пайдаланылды.

Деректер жинағы CSV (сурет 2.1.2) пішімінде қол жетімді және сентименталды талдау үшін әртүрлі машиналық оқыту алгоритмдерін үйрету және бағалау үшін пайдаланылуы мүмкін. Осы деректер жинағын пайдалану арқылы зерттеушілер модельдерінің дәлдігін жақсарта алады және пікірлердің көңіл-күйі туралы көбірек негізделген шешімдер қабылдай алады.



Сурет 2.1.1. Деректер жинағының көңіл-күй үлесі

	review	sentiment
2	Өте жақсы ризамын телефон ұнады!	1
3	Барлығы ұнады, уақытында келді, сатушы жақсы, бәрін айтып түсіндіріп берді, жұмыстарынызға береке берсін	1
4	Керемет телефон екен.	1
5	Уақытында келді	1
6	Рахмет сіздерге! Сапалы, жақсы телефон екен	1
7	Маған қатты ұнады! Бірінші рет IPhone телефонын алып тұрмын. Камерасы ұнады. Және тез келді	1
8	Керемет. Ұлыма ұнады.	1
9	Маған ұнады керемет.	1
10	Өте керемет, жеткізу уақытынан бұрын әкелді	1
11	Телефон балама ұнады	1
12	Күшті. Бәрі дұрыс	1
13	Заряд жақсы ұстайды. Камера керемет!	1
14	Бәрі жақсы. Уақытымен келді. Курьерге рахмет	1
15	Рахмет, қайын сіңіліме алып едім, ұнады	1
16	Смартфон жылдам жұмыс істеп тұр. Экраны үлкен 6.6! Камера, интернет, дыбысы жақсы.	1
17	Қызыма ұнады, мектеп балаларға ұстауға болады	1
18	Тез жеткізді. Телефон керемет, көңілімнен шығып жатыр.	1
19	Өзіме алғанмын, жақсы телефон, кино көруге ыңғайлы, ойындар ойнағанда жақсы, музыка құлаққа жағымды, таза ойналады, сөйлескенде	1
20	Әйеліме алып бердім, мәз болып қуанып қалды. Ақшыл түсін алуым керек еді әттең. Дауысы керемет шығады	1
21	Алуға кеңес беремін, фото, видео күшті түсіреді, өзі де қолға ыңғайлы, жақсы, әдемі	1

Сурет 2.1.2. Деректер жинағынан оң пікірлердің үзіндісі

38	Бұл телфон өте нашар екен алмаңыздар	0
39	Көптеген кемшіліктер бар, егер сіздің соңғы телефоныңыз iPhone желісінен болса, онда барлық жағынан жақсартулар болады, бірақ егер с	0
40	Телефон өте қымбат, мен баламды туған күніне сатып алдым. Пластмасса өте жұмсақ және қазірдің өзінде сызылған. Экран 11 жастағы ба	0
41	Орташа телефон, ойланып жұмыс жасайды. Енді ашып көріп жатырмын	0
42	Қосылғаннан бастап қата берді	0
43	Батарейкасы тез отырып қалады, құлаққап жоқ және дауысы қатты шықпайды. Алғаныма өкіндім	0
44	Қателігі көп. Қалтада тұрып блоктан да ашылады, басылып кетеді. 1ші звонокта дыбыс шықпайды, байланысты үзіп қайта қоңырау шалась	0
45	Тек зарядкасынын басы жоқ, шнуры кана келді. Құлаққаптар жоқ. Келешекте шнуры жоқ болса да тан қалманыз.	0
46	Нашар телефон, ұнамады маған	0
47	Телефон ұнамады, сеть нашар, үнемі самсунг алушы едім, осы жолы өкініп қалдым	0
48	Екі ай ұстадық, жасамай қалды. Өте нашар телефон	0
49	Бүгін алдық, телефон қатады, ештеңеге кірмейді, кірсең, шыға алмайсың, экраны істемейді.	0
50	телефонды жақында алдым, сапасы онша емес, қатып қала береді	0
51	Маған ұнамады, зарядкасы тез өліп қала береді	0
52	Кредитке алмауға кеңес беремін. 300 мың телефонды 470 мыңға алдым	0
53	Сеть кейде ұстамай қалады	0
54	Ұнамады, Зарядкасының басы жоқ	0
55	Заряд ойнап жатыр	0
56	Зарядтау құрылғысы жоқ, құлаққап жоқ	0
57	Қуаты тез өліп қалады	0
58	Құлаққап пен адаптер жоқтығы	0

Сурет 2.1.3. Деректер жинағынан теріс пікірлердің үзіндісі

2.2 Деректер жиынтығын алдын ала өңдеу және дайындау

Мәтінді алдын ала өңдеу және дайындау сентименталды талдаудағы маңызды қадам болып табылады. Ол талдауды жеңілдету үшін арналған мәтіндік деректерді тазалауды, түрлендіруді және ұйымдастыруды қамтиды. Мәтінді алдын ала өңдеудің негізгі мақсаты қажет емес ақпаратты жою, мәтінді жалпы регистрге түрлендіру және деректер өлшемдерін азайту арқылы өңделмеген мәтінді талдауға қолайлы пішімге түрлендіру болып табылады.

Токенизация — сөйлем немесе абзац сияқты үлкенірек мәтін бөлігін лексема деп аталатын кішірек бірліктерге бөлу процесі. Сезімдерді талдауда токенизация мәтінді алдын ала өңдеу мен дайындаудағы маңызды қадам болып табылады, себебі ол құрылымдалмаған мәтіндік деректерді құрылымдық деректерге түрлендіруге көмектеседі.

Сезімдерді талдаудағы токенизацияның негізгі мақсаты мәтіннің сезімін жеткізетін сәйкес сөздерді немесе сөз тіркестерін алу болып табылады. Токенизация сөзге негізделген, таңбаға негізделген және ішкі сөзге негізделген токенизация сияқты әртүрлі әдістерді қолдану арқылы орындалуы мүмкін.

Сөзге негізделген токенизация - мәтін бос орындар немесе тыныс белгілері негізінде жеке сөздерге бөлінген ең жиі қолданылатын әдіс. Мысалы, «Маған телефон ұнады» деген сөйлемді «Маған», «телефон» және «ұнады» сияқты жеке сөздерге айналдыруға болады.

Стопсөзді жою мәтінді алдын ала өңдеудегі және көңіл-күйді талдауға дайындықтағы маңызды қадам болып табылады. Тоқтау сөздер – тілде жиі

кездесетін сөздер, мысалы, «мен», «сен», «ол», «міне», «осынбай» және т.б. Бұл сөздердің негізгі мағынасы жоқ және көбінесе сөйлем құрау үшін қолданылады. Мәтіннен тоқтау сөздерін жою деректердегі шудың мөлшерін азайтып, көңіл-күйді талдаудың дәлдігін арттырады.

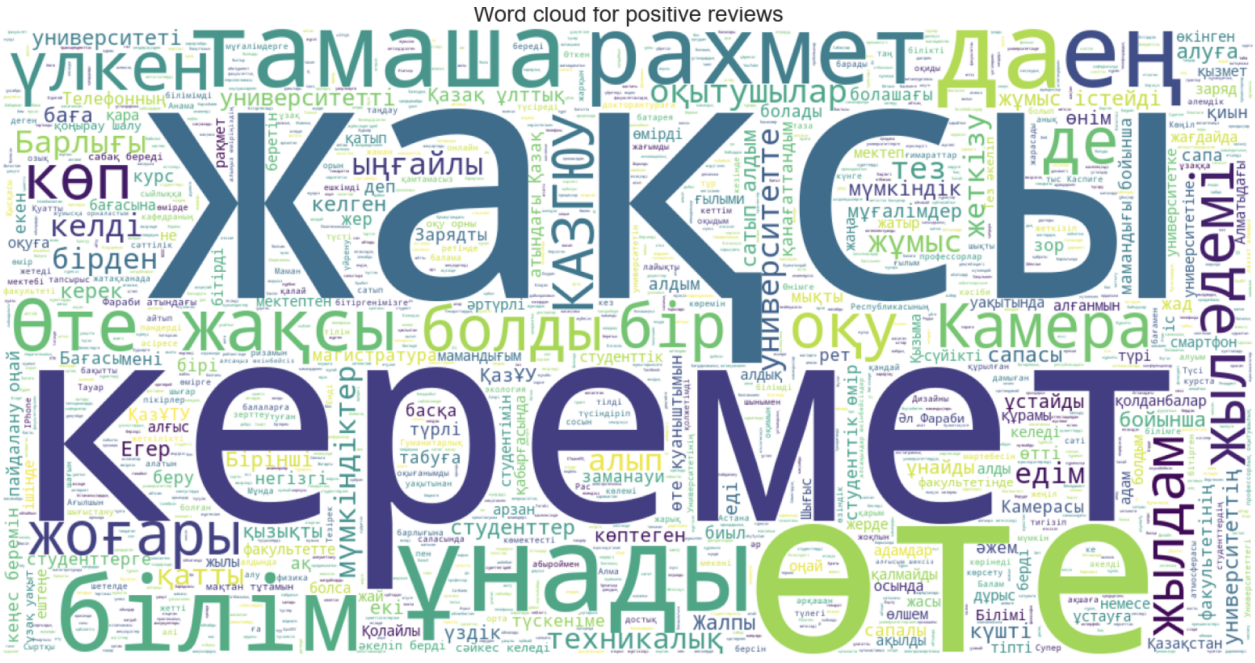
Тоқтау сөздерді алып тастау процесі мәтіндегі тоқтау сөздерді анықтауды және оларды алып тастауды қамтиды. Мұны алдын ала анықталған тоқтату сөздер тізімдері арқылы жасауға болады.

NLTK (Natural Language Toolkit), spaCy және Scikit-learn сияқты мәтіндік деректерден тоқтату сөздерді жою үшін пайдалануға болатын бірнеше кітапханалар мен құралдар бар. NLTK-де, мысалы, тоқтату сөздерін бірнеше тілге арналған тоқтату сөздерінің тізімі бар тоқтату сөздері модулі арқылы жоюға болады. Жұмыста аталған NLTK бағдарламасының қазақ тіліне арналған стопсөзі қолданылған.

Аталған әдісті қолдана отырып мәтінді алдын ала өңдеудегі маңызды қадам болып табылатын стопсөздерді алып, ол талдаудың дәлдігі мен сенімділігін арттырдық. Мәтін деректерінде талдаудың дәлдігіне әсер ететін емле қателері болуы мүмкін. Емлені тексеру және түзету қате жазылған сөздерді анықтауды және түзетуді қамтиды.

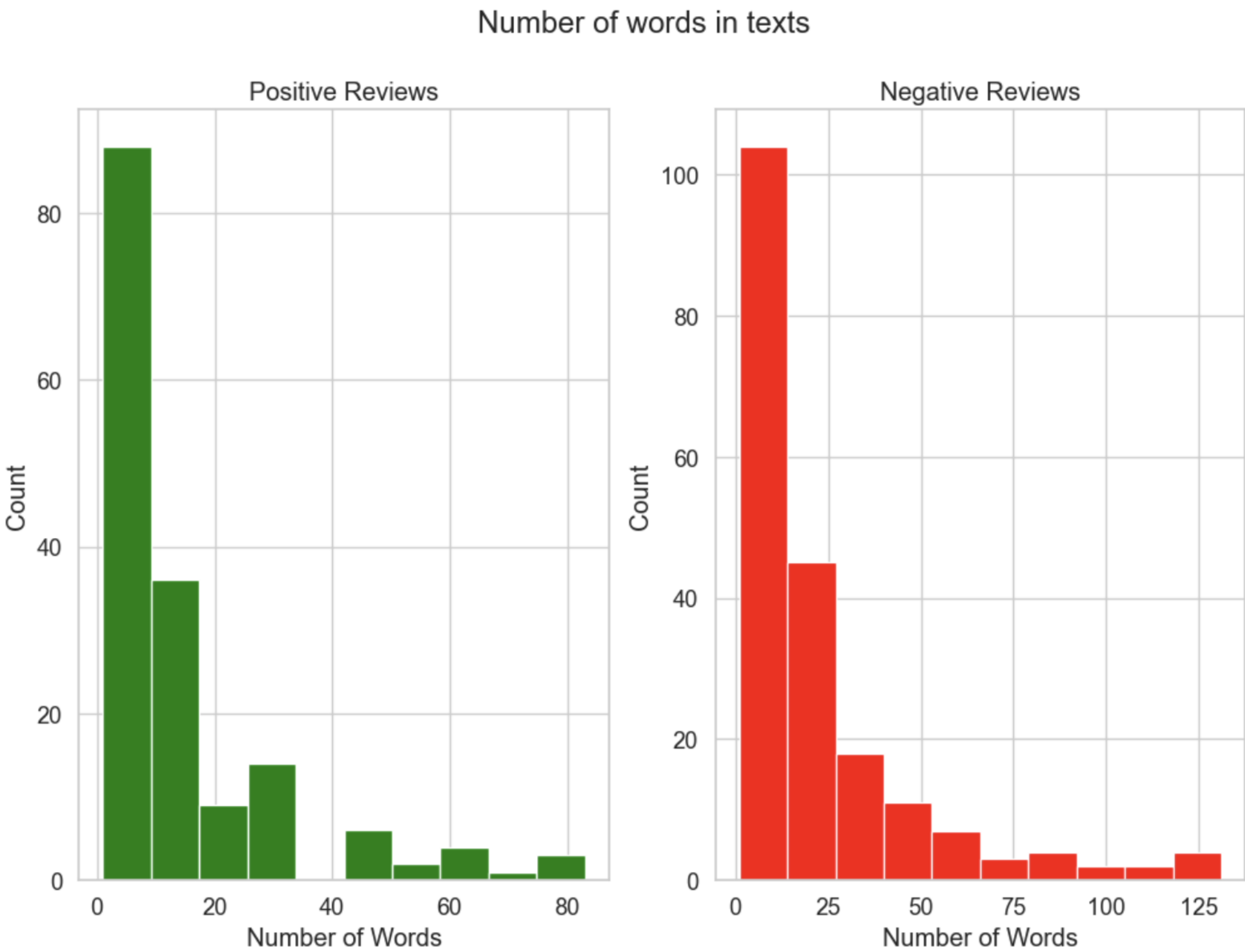
2.3 Өңделген деректер жиынтығына шолу

Қайталанған деректерді өшіріп, токенизация мен стопсөздерді қолдана отырып, қате жазылған сөздерді дұрыстағаннан кейін деректер жиынтығына көз салсақ (сурет 2.3.1, сурет 2.3.2, сурет 2.3.3, сурет 2.3.4).

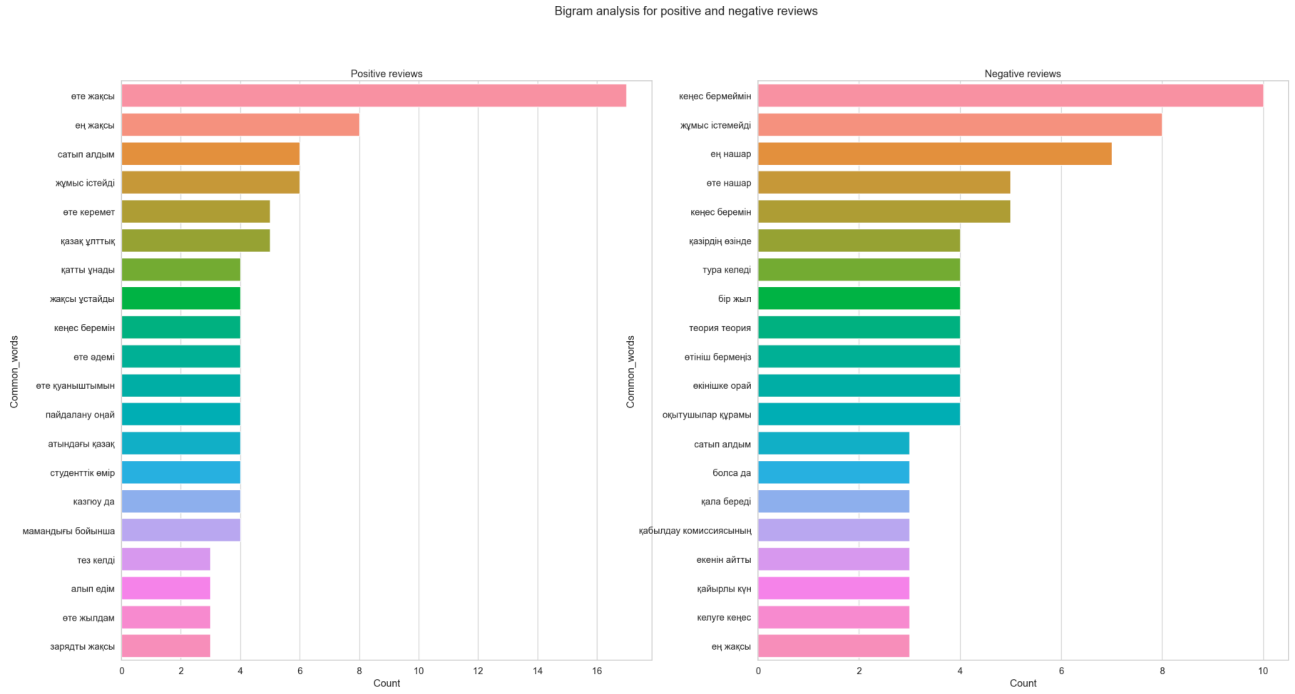


Сурет 2.3.1. Оң пікірлердің сөз бұлты (word cloud)

Сурет 2.3.2. Теріс пікірлердің сөз бұлты (word cloud)



Сурет 2.3.3. Мәтіндегі сөздер саны



Сурет 2.3.4. Теріс және оң пікірлерге арналған биграмды талдау

2.4 Өңделген деректерден белгілерді шығару

Белгілерді шығару (Feature extraction) мәтіндік деректерді машиналық оқыту алгоритмдері түсінетін сандық векторларға түрлендіруді қамтитын маңызды қадам болып табылады. Белгілерді шығару үшін қолданылатын кең таралған әдістердің бірі - Term Frequency - Inverse Document Frequency (TF-IDF) векторизаторы.

TF-IDF – жинақтағы немесе корпустағы құжат үшін сөздің маңыздылығын көрсететін сандық статистика. Ол екі компоненттің туындысы ретінде есептеледі: мерзімді жиілік (Term Frequency, TF) және кері құжат жиілігі (Inverse Document Frequency, IDF) [17].

Term Frequency компоненті белгілі бір құжаттағы сөздің жиілігін білдіреді және сөздің кездесу санын құжаттағы сөздердің жалпы санына бөлу арқылы есептеледі.

$$TF = \frac{\text{құжаттағы терминнің жиілігі}}{\text{құжаттағы терминдердің жалпы саны}} \quad (2.1)$$

Ал Inverse Document Frequency компоненті корпустағы сөздің сиректігін өлшейді. Ол корпустағы құжаттардың жалпы санын сөзді қамтитын құжаттардың санына бөлу, содан кейін нәтиженің логарифмін алу арқылы есептеледі.

$$IDF = log(\frac{\text{корпустағы құжаттардың саны}}{\text{терминді қамтитын корпустағы құжаттардың саны}}) \quad (2.2)$$

Содан кейін TF-IDF бағасы құжаттағы әрбір сөз үшін мерзімді жиілік құрамдастарын және құжаттың өзара жиілік құрамдастарын көбейту арқылы есептеледі. Бұл әрбір өлшем корпустағы бірегей сөзге сәйкес келетін әрбір құжат үшін сандық векторға әкеледі.

$$TF - IDF = TF * IDF \quad (2.3)$$

TF-IDF векторизаторы TF-IDF формуласын құжаттар корпусына автоматты түрде қолданатын және сандық векторлардың матрицасын жасайтын құрал. Матрицаның әрбір жолы құжатты, ал әрбір баған корпустағы бірегей сөзді білдіреді.

Алынған матрицаны Naive Bayes, Support Vector Machines және Logistic Regression сияқты сентименталды талдау үшін әртүрлі машиналық оқыту алгоритмдеріне кіріс ретінде пайдалануға болады. Бұл алгоритмдер TF-IDF құжаттарының векторлық көріністеріндегі үлгілерге негізделген құжаттарды жіктеуді үйрене алады.

TF-IDF Vectorizer — сезімді талдауда мәтіндік деректерден мүмкіндіктерді шығаруға арналған қуатты құрал, өйткені ол құжаттағы сөздердің маңыздылығын да, корпустағы сирек кездесетіндігін де көрсетеді. Бұл көңіл-күйді талдау үшін машиналық оқыту үлгілерінің дәлдігі мен тиімділігін жақсартуға көмектеседі.

2.5 Моделді бағалау жолдары

Жұмыста моделдің дәлдігін тексеру үшін оны 1-кестеде көрсетілген шатасу матрицасы арқылы болжамды және шынайы мәндер арасындағы алшақтықты анықтадық. Және де модел өнімділігін бағалау үшін анықтық (precision), толықтық (recall) және F1 ұпайлары көрсеткіштер ретінде пайдаланылады. [20]

Кесте 1

Шатасу матрицасы

Класстар	Оң	Теріс
Оң	True Positive (TP)	False Positive (FP)
Теріс	True Negative (TN)	False Negative (FN):

Шатасу матрицасы – бұл жіктеу үлгісін бағалау үшін пайдаланылатын болжамды және нақты мәндердің төрт комбинациясының кестесі:

- True Positive (TP): модель нақты мәнді дұрыс болжады және оң нәтиже көрсетеді

- True Negative (TN): модель нақты мәнді дұрыс болжады және теріс нәтиже көрсетеді
- False Positive (FP): модель нақты мәнді оң деп болжады және ол дұрыс емес
- False negative (FN): модель нақты мәнді теріс деп болжады және ол дұрыс емес

Анықтық және толықтық есебі төменде көрсетілген:

$$\text{Анықтық} = \frac{TP}{TP + FP} \qquad \text{Толықтық} = \frac{TP}{TP + FN} \tag{2.4}$$

Анықтық – дұрыс болжамдардың пропорциялары, ал толықтық – бұл модель оң деп жіктелген нүктелердің үлесі іс жүзінде оң екенін білдіреді.

Дәлдікті және толықтықты ескеретін F1-ұпай келесідей есептеледі:

$$F1 = \frac{2 * \text{Дәлдік} * \text{Толықтық}}{\text{Дәлдік} + \text{Толықтық}} \tag{2.5}$$

Бөлімге қорытынды

Қорытындылай келе бұл бөлімде Веб-ресурстардан өзекті деректерді жинау, олардың негізінде машина алдын ала өңдеуді жүзеге асыру туралы кеңінен айтылды. Деректер жиынтығын алдын ала өңдеу және дайындау және өңделген деректер жиынтығына шолу жасалды. Кейін өңделген деректерден машиналық оқыту алгоритмдері түсінетін сандық векторларға түрлендіруге арналған Term Frequency - Inverse Document Frequency (TF-IDF) әдісін қолдана отырып белгілерді шығарылды. Және де моделдің дәлдігін тексеру үшін оны шатасу матрицасы арқылы болжамды және шынайы мәндер арасындағы алшақтықты анықтауға шолу жасалды.

3 Сентименталды талдау модельдерін қолдану

3.1 Лексиконға негізделген тәсіл

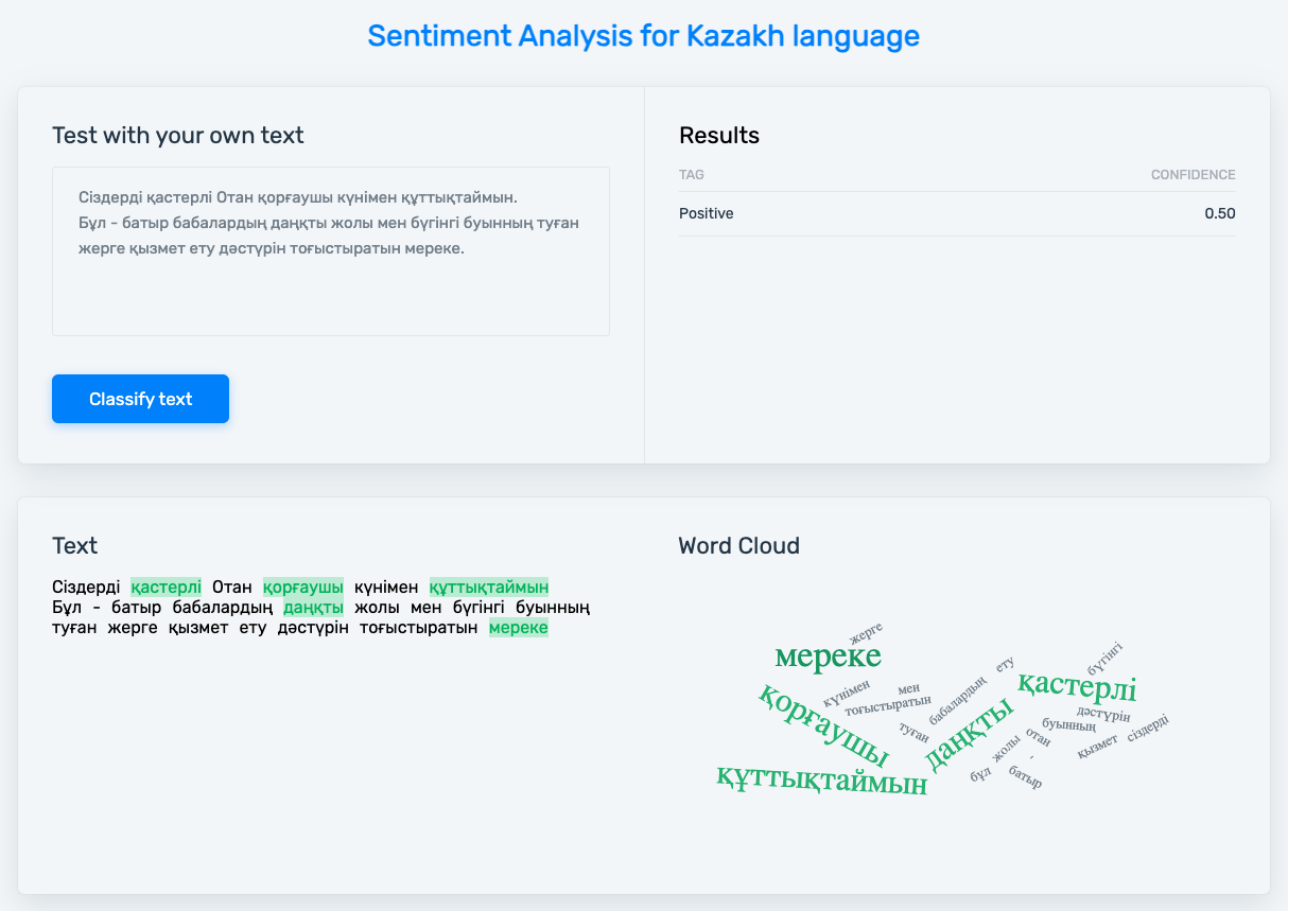
Жұмыста сөздікке негізделген әдістер сентименталды талдаудың танымал және жиі қолданылатын AFINN тәсілі қолданылды.

ReactJS – пайдаланушы интерфейстерін құруға арналған танымал JavaScript кітапханасы. Бұл интерактивті құрамдастарды жасауға және қолданбаның күйін тиімді басқаруға мүмкіндік береді. ReactJS көмегімен веб-сайтыңыздың пайдаланушы интерфейсін қайта пайдалануға болатын компоненттерге бөлу арқылы оңай жобалауға және құрылымдауға болады. ReactJS виртуалды DOM жылдам және тегіс пайдаланушы тәжірибесін қамтамасыз ете отырып, тек қажетті құрамдастарды тиімді жаңартады және көрсетеді. ReactJS қолдана отырып сентименталды талдау нәтижелерін көрсету үшін ыңғайлы интерфейс жасалды.

JavaScript – клиент жағында да (браузерде) де, сервер жағында да (Node.js көмегімен) пайдалануға болатын әмбебап бағдарламалау тілі. JavaScript мәтіндік деректерді өңдеу және талдау үшін қолайлы ететін қуатты жолды және тұрақты өрнек мүмкіндіктерін қамтамасыз етеді. JavaScript көмегімен AFINN сөздер тізімі файлын оқуға және талдауға, сөздердің көңіл-күй сөздігін жасауға және пайдаланушы енгізуі негізінде сезім талдауын орындауға болады. JavaScript икемділігі көңіл-күйді талдау алгоритмдері мен әдістерін енгізуді және өзгертуді жеңілдетеді.

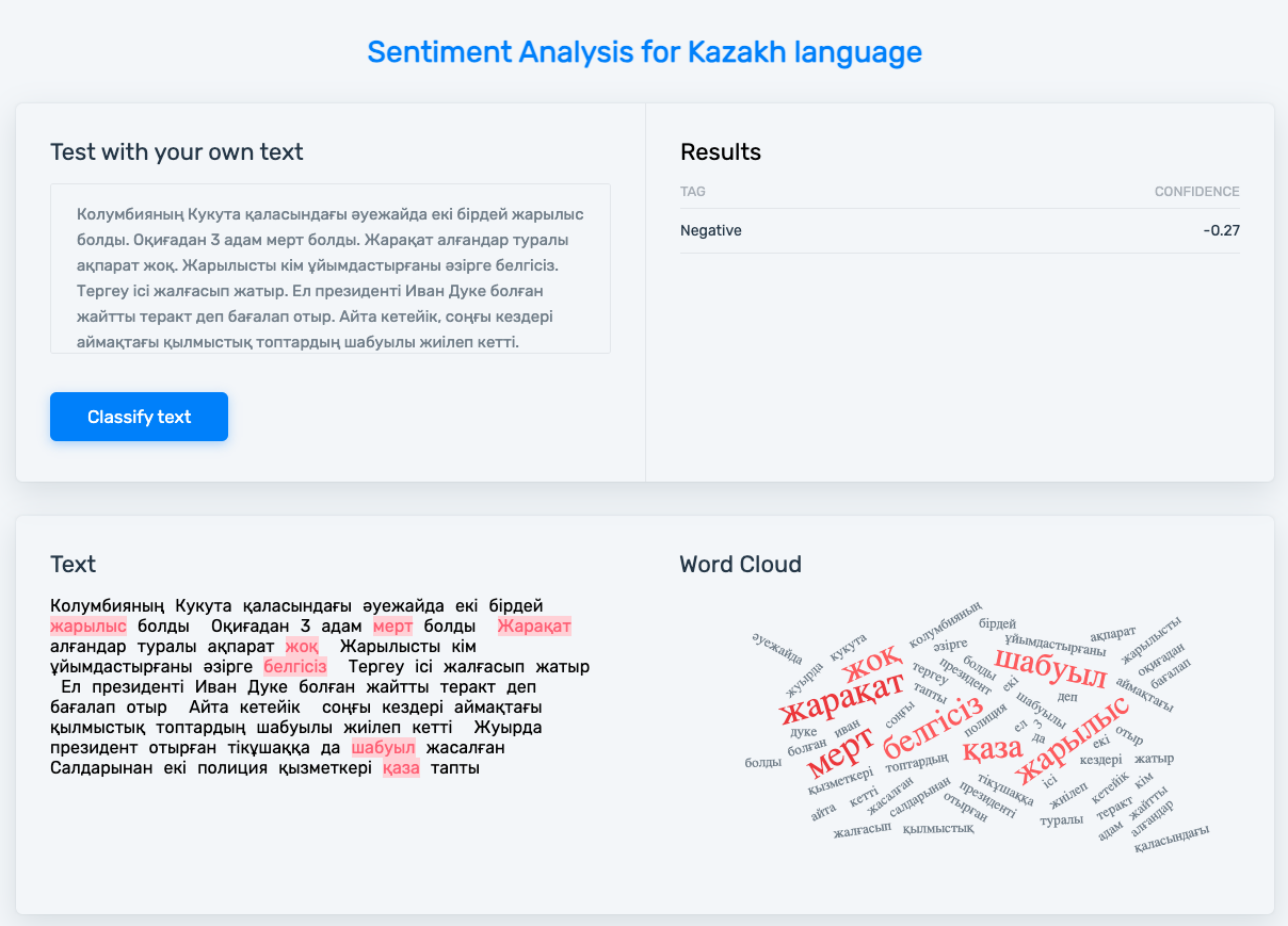
Веб-қосымшаны құру үшін жылдам және тиімді әзірлеу ортасын қамтамасыз ететін құрастыру құралы ретінде Vite пайдаланылды.

Интерфейс пайдаланушы өз мәтінін енгізе алатын енгізуден тұрады және «мәтінді жіктеу» түймесін басу арқылы пайдаланушы мәтінге негізделген көңіл-күйді мәнін және сөз бұлтын көре алады. Алдымен енгізілген мәтінге токенизация әдісі жасалып, кейін токендерді қарай отырып AFINN тізімінде мұндай токен бар-жоғы тексеріледі. Тексерістен кейін мәтін үшін жалпы көңіл-күй ұпайын алу үшін жеке ұпайлар жинақталады. Мысал ретінде “Сіздерді қастерлі Отан қорғаушы күнімен құттықтаймын. Бұл - батыр бабалардың даңқты жолы мен бүгінгі буынның туған жерге қызмет ету дәстүрін тоғыстыратын мереке.” мәтінін қарастырайық (сурет 3.1.1).



Сурет 3.1.1. Веб-қосымшаның оң көңіл-күйі

Ал теріс көңіл-күй мәтін ретінде “Колумбияның Кукута қаласындағы әуежайда екі бірдей жарылыс болды. Оқиғадан 3 адам мерт болды. Жарақат алғандар туралы ақпарат жоқ. Жарылысты кім ұйымдастырғаны әзірге белгісіз. Тергеу ісі жалғасып жатыр. Ел президенті Иван Дуке болған жайтты теракт деп бағалап отыр. Айта кетейік, соңғы кездері аймақтағы қылмыстық топтардың шабуылы жиілеп кетті. Жуырда президент отырған тікұшаққа да шабуыл жасалған. Салдарынан екі полиция қызметкері қаза тапты.” мәтіні қарастырылды (сурет 3.1.2).

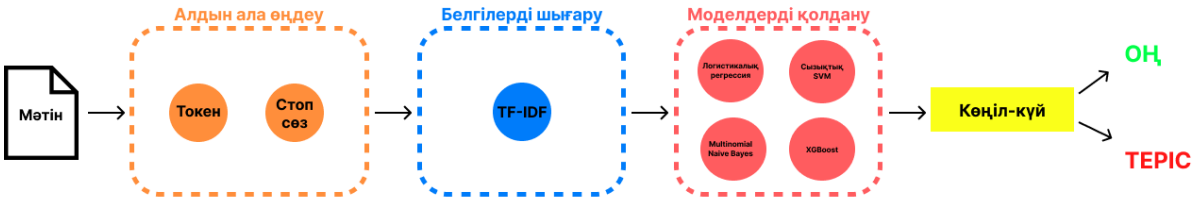


Сурет 3.1.2. Веб-қосымшаның теріс көңіл-күйі

3.2 Машиналық оқыту тәсілдері

Деректер жинағын scikit-learn кітапханасының train_test_split функциясын пайдалана отырып оқыту және сынақ жиындарына бөлінді. Test_size параметрі 0,2 мәніне орнатылды, яғни бұл деректердің 20% тестілеу үшін пайдаланылды, ал 80% оқыту үшін пайдаланылды.

Оқыту және сынақ жиындарындағы мәтіндік деректер TF-IDF көрсетіліміне негізделген сандық мүмкіндік векторларына түрлендірілді. Бұл түрлендіру машиналық оқыту алгоритмдеріне мәтіндік деректермен тиімді жұмыс істеуге мүмкіндік береді, өйткені олар әдетте сандық енгізуді қажет етеді. Алынған TF-IDF мүмкіндік векторларын машиналық оқыту үлгілерін оқыту және бағалау үшін кіріс ретінде пайдалануға болады (сурет 3.2.1).



Сурет 3.2.1. Машиналық оқыту тәсілдері

Кейін түрлендірілген TF-IDF деректерін пайдалана отырып, sklearn.linear_model функциясы арқыл логистикалық регрессия үлгісіне арналған оқыту, болжау, бағалау және визуализация қадамдары көрсетілді. Алдымен логистикалық регрессия үлгісін көрсететін LogisticRegression класының данасы жасалды. Логистикалық регрессия моделі және мақсатты белгілер оқу деректері арқылы оқытыла отырып, сәйкестендіру әдісі берілген деректерге ең жақсы сәйкестікті табу үшін үлгі параметрлерін реттейді. Үлгінің өнімділігін бағалау үшін дәлдік, жіктеу есебі және шатасу матрицасы төмендегідей (сурет 3.2.2):

	precision	recall	f1-score	support
0	0.82	0.88	0.85	107
1	0.87	0.80	0.83	105
accuracy			0.84	212
macro avg	0.84	0.84	0.84	212
weighted avg	0.84	0.84	0.84	212

Logistic Regression Accuracy : 83.96%

Сурет 3.2.2. Логистикалық регрессия моделінің дәлдігі

Ал Multinomial Naive Bayes моделін қолдану үшін sklearn.naive_bayes функциясы қолданылды. Multinomial Naive Bayes классификаторының данасын жасалып, оған TF-IDF векторланған мүмкіндіктер матрицасы және сәйкес белгілер матрицасы арқылы ұсынылған жаттығу деректеріне үйретілді. Оқытылған MNB классификаторын пайдаланып сынақ деректерінің белгілерін болжалды. Болжалды белгілерді шынайы белгілерменсалыстыру арқылы дәлдік ұпайын есептелді (сурет 3.2.3).

	precision	recall	f1-score	support
0	0.88	0.79	0.83	107
1	0.80	0.89	0.84	105
accuracy			0.83	212
macro avg	0.84	0.84	0.83	212
weighted avg	0.84	0.83	0.83	212

Multinomial Naive Bayes Classifier Accuracy : 83.49%

Сурет 3.2.3. Multinomial Naive Bayes моделінің дәлдігі

Linear SVM моделін қолдану үшін sklearn.svm қолданылса, XGboost моделі үшін xgboost.sklearn функциясы қолданылды (сурет 3.2.4, сурет 3.2.5)

	precision	recall	f1-score	support
0	0.82	0.87	0.85	107
1	0.86	0.81	0.83	105
accuracy			0.84	212
macro avg	0.84	0.84	0.84	212
weighted avg	0.84	0.84	0.84	212

Linear Support Vector Classifier Accuracy : 83.96%

Сурет 3.2.4. Linear SVM моделінің дәлдігі

	precision	recall	f1-score	support
0	0.82	0.76	0.79	107
1	0.77	0.83	0.80	105
accuracy			0.79	212
macro avg	0.79	0.79	0.79	212
weighted avg	0.79	0.79	0.79	212

XGBoost Accuracy : 79.25%

Сурет 3.2.5. XGboost моделінің дәлдігі

Оң көңіл-күйде жазылған “Телефон өте әдемі, ұнады, енеме сыйлыққа алғанмын” мәтініне болжау нәтижелері 2–кестеде көрсетілген.

Кесте 2

Машиналық оқыту әдістерінің оң көңіл-күй болжау мысалы

Әдіс	Көңіл-күй
Logistic regression	Оң
Multinomial Naive Bayes	Оң
Linear SVM	Оң
XGBoost	Оң

Ал, 3–кестеде көрсетілген “Батарейкасы тез отырып қалады, құлаққап жоқ және дауысы қатты шықпайды. Алғаныма өкіндім” мәтініне болжау жасалғанда, машиналық оқыту әдістеріні төмендегідей нәтиже қайтарды.

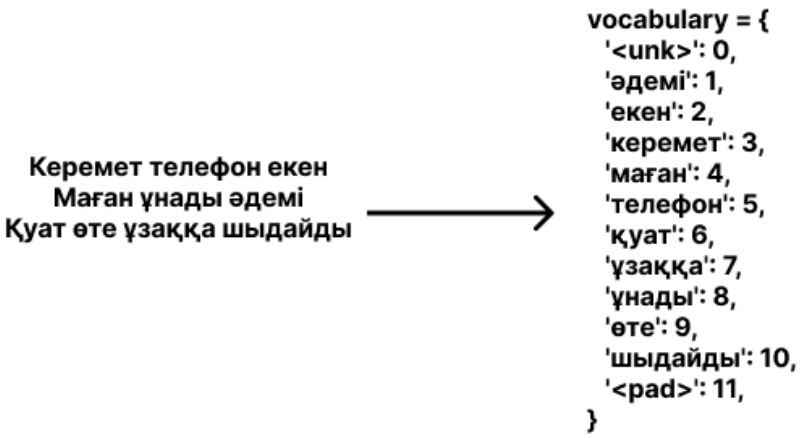
Кесте 3

Машиналық оқыту әдістерінің теріс көңіл-күй болжау мысалы

Әдіс	Көңіл-күй
Logistic regression	Теріс
Multinomial Naive Bayes	Теріс
Linear SVM	Теріс
XGBoost	Теріс

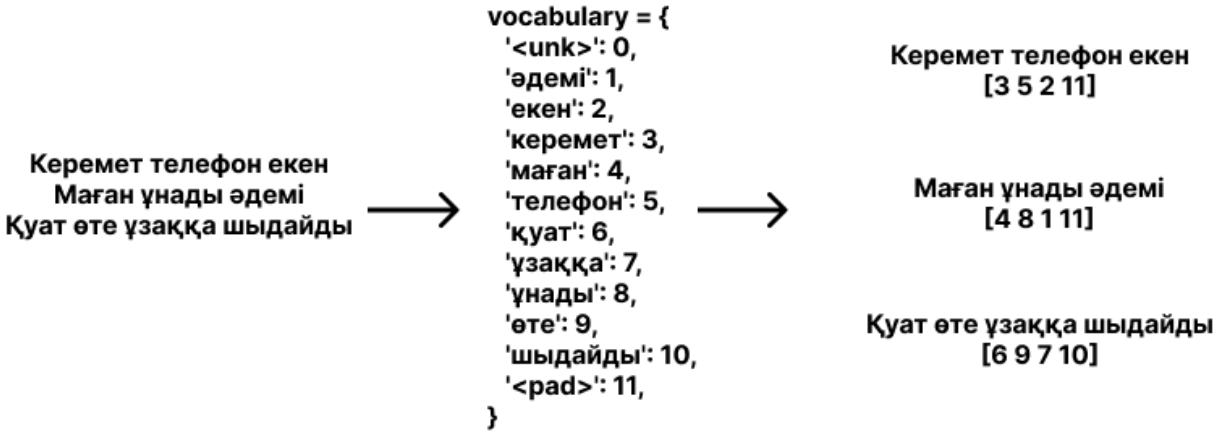
3.3 Терең оқыту тәсілдері

Қайталанатын нейрондық желі (RNN) тәсілін қолдану үшін алдымен сөздік (vocabulary) құрастыру керек болады. Бұл деректер жиынындағы әрбір бірегей сөздің сәйкес индексі (бүтін сан) болатын тиімді іздеу кестесі болып табылады. Сөздік құру мысалы төменде көрсетілген (сурет 3.3.1).



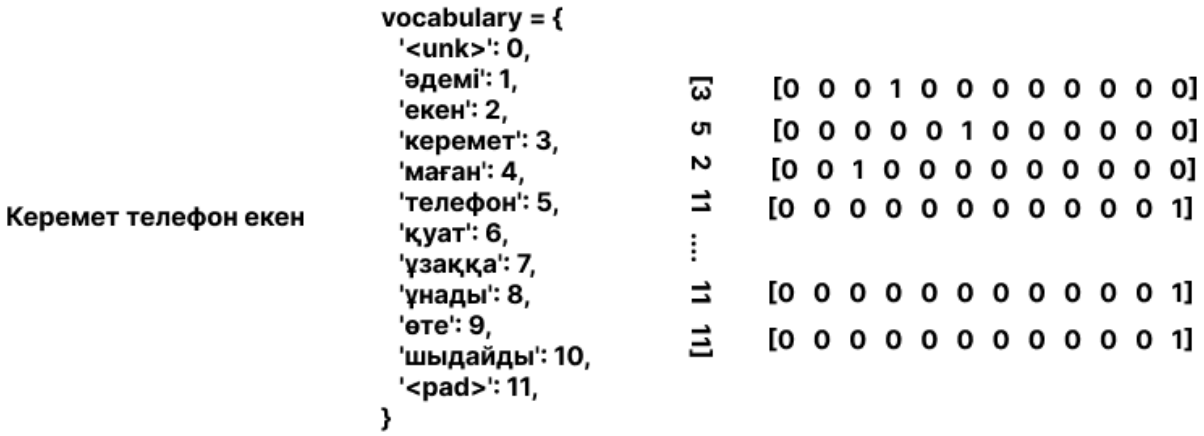
Сурет 3.3.1. Сөздік құру мысалы

Кейін сөздікті пайдаланы отырып, сынақ жинақтағы мәтін деректеріне индекстерді тағайыдық, деректердегі әрбір сөзді сөздіктегі сәйкес индекспен ауыстырдық. Бұл қадамда мәтіндік деректер RNN кірісі ретінде пайдаланылуы мүмкін сандық көріністерге түрлендіріледі. Мысал төменде көрсетілген (сурет 3.3.2).



Сурет 3.3.2. Деректерге индекстерді тағайындау мысалы

Әрбір индекс әрбір сөз үшін біртұтас векторды құру үшін пайдаланылады. Біртұтас вектор (one-hot vector) - өлшемі сөздіктегі бірегей сөздердің жалпы саны болатын, бір ғана элементі 1, ал басқа барлық элементтері 0 болатын вектор. Жұмыста бұл мақсатта torch кітапханасы қолданылды. Мысал төменде көрсетілген (сурет 3.3.3).



Сурет 3.3.3. Біртұтас вектор мысалы

Енгізу қабаты (embedding layer) біртұтас векторымызды тығыз кірістіру векторына түрлендіру үшін пайдаланылды, жұмыста torch кітапханасы пайдаланылды (сурет 3.3.4).

```
class RNN(torch.nn.Module):

    def __init__(self, input_dim, embedding_dim, hidden_dim, output_dim):
        super().__init__()

        self.embedding = torch.nn.Embedding(input_dim, embedding_dim)
        self.rnn = torch.nn.LSTM(embedding_dim,
                                 hidden_dim)

        self.fc = torch.nn.Linear(hidden_dim, output_dim)


    def forward(self, text):

        embedded = self.embedding(text)

        output, (hidden, cell) = self.rnn(embedded)

        hidden.squeeze_(0)

        output = self.fc(hidden)
        return output
```

Сурет 3.3.4. RNN моделін құру

Жұмыста torch.nn.Module класынан мұраланған RNN класы құрастырылды. Модел конструкторы енгізілетін сөздіктің өлшемі, яғни мәтіндік деректердегі бірегей таңбалауыштардың саны (input_dim), кіріс таңбалауыштарын көрсететін ендіру векторларының өлшемділігі (embedding_dim), LSTM жасырын күйіндегі бірліктердің саны (hidden_dim), көңіл-күй болжамын көрсететін шығыстың өлшемділігі (output_dim) мәндерінен тұрады.

Және де torch.nn.Embedding модулі жасалды, ол тығыз кірістіру векторлары бар енгізу таңбалауыштарын үйренуге және сәйкестендіруге жауап береді. Және ұзақ LSTM қабатының іске асырылуы үшін torch.nn.LSTM модулі құрастырылды. Бұл модул кіріс ретінде кірістірілген енгізуді қабылдайды және жасырын күйлерді жасау үшін сериялық ақпаратты өңдейді. Кейін көңіл-күйді болжауға жауапты соңғы жасырын күйді шығыс өлшеміне салыстыратын torch.nn.Linear модулі жасалды.

Forward әдісі модельдің алға өтуіне жауапты. Ол мәтінді кіріс мәтіндік деректер ретінде қабылдайды. Embedded тензор кіріс мәтінін кірістіру қабаты арқылы өткізу арқылы алынады, іздеуді орындай отырып кіріс токендерді тығыз векторлық көріністерге түрлендіреді. Кейін embedded тензор RNN қабаты арқылы өтіп, әрбір уақыт қадамы үшін LSTM шығыс мүмкіндіктерін қамтитын

(жасырын, ұяшық) мәндерін және LSTM соңғы жасырын күйін қайтарады. Жасырын тензор LSTM арқылы қосылған қосымша өлшемді жою үшін бірінші өлшем бойымен сүзіліеді.

Кейін құрастырылған моделды жаттықтырдық. Процесс 10 дәуірден тұрды. Есептелген градиенттер негізінде үлгі параметрлерін жаңарта отырып жаттығу барысын журналға жазады. Ол сондай-ақ әрбір дәуірден кейін оқыту және тексеру жиындарында үлгінің өнімділігін бағалайды. Соңында модельді оқытудан 60% дәлдік алынды.

Моделдің дәлдігін арттыру үшін packed padded sequences әдісі қолданылды. Packed padded sequences маңыздылығының негізгі себептері ретінде:

- Тиімді есептеу: жинақталған толтырылған тізбектер оқыту және қорытынды жасау кезінде тиімдірек есептеуге мүмкіндік береді. RNN-де енгізу реттіліктері әдетте шағын топтамада параллельді түрде өңделеді. Толтырылған толтырылған тізбектер толтырылған элементтердегі қажетсіз есептеулердің қажеттілігін болдырмайды, бұл жылдам жаттығу мен қорытынды уақытына әкеледі.
- Жад тиімділігі: толтырылған тізбектер жад талаптарын арттырады, өйткені олар толтырылған элементтердің айтарлықтай санын енгізеді. Тізбектерді орау арқылы толтырылған элементтер тиімді түрде маскирленеді, жад ізін азайтады және жадты пайдалануды оңтайландырады.
- Жақсартылған модел өнімділігі: толтырылған тізбектер шу мен қажетсіз есептеулерді енгізу арқылы үлгі өнімділігіне теріс әсер етуі мүмкін.

Жұмыста torch.nn.utils.rnn.pack_padded_sequence() функциясын пайдаланып, packed padded sequences әдісі қолданылды (сурет 3.3.5).

```
class RNN(torch.nn.Module):

    def __init__(self, input_dim, embedding_dim, hidden_dim, output_dim):
        super().__init__()

        self.embedding = torch.nn.Embedding(input_dim, embedding_dim)
        self.rnn = torch.nn.LSTM(embedding_dim,
                                 hidden_dim)

        self.fc = torch.nn.Linear(hidden_dim, output_dim)


    def forward(self, text, text_length):
        embedded = self.embedding(text)

        ## NEW
        packed = torch.nn.utils.rnn.pack_padded_sequence(embedded, text_length.to('cpu'))

        packed_output, (hidden, cell) = self.rnn(packed)

        hidden.squeeze_(0)

        output = self.fc(hidden)
        return output
```

Сурет 3.3.5. Жақсартылған RNN моделі

Бұл функция толтыру таңбалауыштары жоқ реттіліктерді көрсететін қорапталған реттілік нысанын жасайды. Аталған әдісті қолдана отырып 80% дәлдікке қол жеткізілді.

Бөлімге қорытынды

Қорытындылай келе үшінші бөлімде жасалынған сентименталды талдау модельдерін қолдануға тоқталып өтілді. Лексиконға негізделген әдіс ретінде AFINN, машиналық оқыту әдістері ретінде логистикалық регрессия, Multinomial Naive Bayes, Сызықтық SVM, XGBoost қарастырылды. Ал терең оқыту әдісі ретінде Long short-term memory (LSTM), LSTM жақсарту ұшін LSTM-pack sequence оқыту әдістері python тілінің көмегімен жасалды.

4 Нәтижелер және талқылау

4.1 Нәтижелер

Бұл жұмыста қазақ тілінде веб-ресурстардан пікірлер жиналды.

Бұл деректерге Логистикалық регрессия, Multinomial Naive Bayes, Сызықтық SVM, XGBoost, Long short-term memory (LSTM), LSTM жақсарту үшін LSTM-pack sequence оқыту әдістері қолданылды.

Жоғарыда аталған моделдер мен әдістерді бағалау отырып эксперименттің жалпы нәтижелерін 4 – кестеден көруге болады

Кесте 4

Бағалау көрсеткіштері

Әдіс	Көңіл-күй
Logistic regression	83.96%
Multinomial Naive Bayes	83.49%
Linear SVM	83.96%
XGBoost	79.25%
LSTM	60.00%
LSTM (pack padded sequence)	80.00%

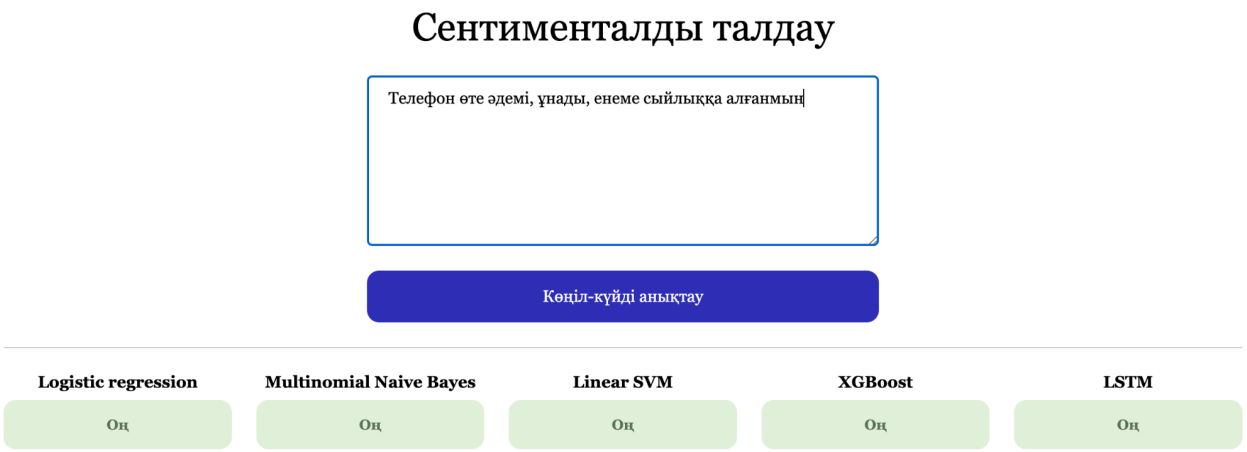
Деректер жинағы көп болмағандықтан машиналық оқыту әдістері дәлірек нәтиже беретінін көрсетті. Логистикалық регрессия үлгісі төмен өлшемді деректер болса және олардың мүмкіндігі сызықты түрде бөлінетін болса, тиімді болуы мүмкін, бірақ жақсы нәтижелерге қол жеткізу үшін үлкен деректер жиынтық қажет етіледі.

LSTM моделінің танымалдылығын оны жүзеге асырудың қарапайымдылығымен сипаттауға болады.

4.2 Визуализациялау

Сентименталды талдау жұмысы әр түрлі қолданбалар арқылы өтуі мүмкін, мұнда пайдаланушы сөздің нені білдіретінін оңай анықтай алады. Бұның бір тәсілі ретінде веб-қосымшалар болуы мүмкін. Практикалық тәжірибенің бір бөлігі осы тәсілдің идеясын көрсете алатын қарапайым веб-қосымшаны құру болды. Веб-қосымша Python кітапханаларына, көңіл-күйді талдауға арналған машиналық және терең оқыту тәсіліне негізделген. Қолданба JavaScript тілінде жазылған және Python кітапханаларын веб-қосымшаға қосу үшін Flask пайдаланылды. Пайдаланушы сөзді немесе сөйлемдерді енгізеді, содан кейін көңіл-күйді анықтау түймешігін басып, әрбір әдіс блогы үшін нәтижелерді алады. Мысал үшін “Телефон өте әдемі, ұнады,

енеме сыйлыққа алғанмын” мәтініне болжау жасалды (сурет 4.2.1). Нәтиже машиналық оқыту тәсілдерінде болғандай оң нәтиже. Ал, “Батарейкасы тез отырып қалады, құлаққап жоқ және дауысы қатты шықпайды. Алғаныма өкіндім” мәтініне болжау жасалғанда, сәйкесінше теріс нәтиже қайтарылды (сурет 4.2.2).



Сурет 4.2.1. Оң көңіл-күй интерфейсі

Сурет 4.2.2. Теріс көңіл-күй интерфейсі

Бөлімге қорытынды

Қорытындылай келе бұл бөлімде жұмыстағы қазақ тіліндегі деректерді сентименталды талдау моделі мен әдістердің нәтижелері көрсетілген. Лексиконға негізделген AFINN, машиналық оқыту әдістері логистикалық регрессия, Multinomial Naive Bayes, Сызықтық SVM, XGBoost және терең оқыту әдісі ретінде Long short-term memory (LSTM), LSTM жақсарту үшін LSTM-pack sequence оқыту әдістеріне бағалау көрсеткіштері дайындалыды.

Деректер жинағы көп болмағандықтан машиналық оқыту әдістері дәлірек нәтиже беретінін көрсетті.

JavaScript тілінде жазылған және Python кітапханаларын веб-қосымшаға қосу үшін Flask пайдалана отырып веб-қосымша жасалды. Пайдаланушы сөзді немесе сөйлемдерді енгізеді, содан кейін көңіл-күйді анықтау түймешігін басып, әрбір әдіс блогы үшін нәтижелерді алады.

ҚОРЫТЫНДЫ

Қазіргі уақытта әлеуметтік желідегі жазбалар, жаңалықтар және тұтынушылардың пікірлері сияқты қазақ тіліндегі цифрлық контентті пайдаланудың артуымен үлкен көлемдегі деректерді өңдей алатын және адамдардың пікірі мен көзқарасы туралы құнды ақпарат беретін автоматтандырылған сентименталды талдау әдістеріне қажеттілік артып отыр. Қазақ тіліндегі деректерді сентименталды талдау табиғи тілді өңдеу саласында барған сайын маңызды бола түсуде. Бұл әдісті әлеуметтану, психология және саясаттану, денсаулық сақтау, қаржы және білім беруді қоса алғанда, әртүрлі салаларда салаларда зерттеу мақсатында пайдаланылуы мүмкін.

Бұл жұмыс қазақ тіліне арналған деректер ғылымын зерттеу мен сентименталды зерттеулерге эксперименттік үлес болып табылады. Пікір деректеріне негізделген мәтіндерді қолдана отырып, әртүрлі әдістер және олардың дәлдігі салыстырылды. Әлеуметтік желідегі жазбалар, жаңалықтар және тұтынушылардың пікірлері сияқты қазақ тіліндегі цифрлық контентті пайдаланудың артуымен үлкен көлемдегі деректерді өңдей алатын және адамдардың пікірі мен көзқарасы туралы құнды ақпарат беретін автоматтандырылған сентименталды талдау әдістеріне қажеттілік артып отыр.

Негізгі мақсат – сентименталды талдау әдістерін зерттей отырып, әртүрлі тәсілдерді салыстырып қазақ тіліне арналған сентименталды талдаудың барынша нақты үлгісін беру болып табылады.

Жұмыста сентименталды деректерді талдауда қолданылатын негізгі ұғымдар мен әдістерге шолу жасалып, веб-ресурстардан өзекті деректерді жинау, олардың негізінде машина алдын ала өңдеуді жүзеге асырылды. Кейін жасалған үлгілер мен әдістерді нақты деректер жиынына қолдану және олардың тиімділігін бағаланып, ұсынылған үлгілер мен әдістер бойынша озық тәжірибелерді қалай тиімді қолдану керектігі туралы ақпарат пен нұсқаулар берілді.Кейін веб-ресурстардан өзекті деректерді жинау, олардың негізінде машина алдын ала өңдеуді жүзеге асыру туралы кеңінен айтылды. Деректер жиынтығын алдын ала өңдеу және дайындау және өңделген деректер жиынтығына шолу жасалды. Кейін өңделген деректерден машиналық оқыту алгоритмдері түсінетін сандық векторларға түрлендіруге арналған Term Frequency - Inverse Document Frequency (TF-IDF) әдісін қолдана отырып белгілерді шығарылды. Және де моделдің дәлдігін тексеру үшін оны шатасу матрицасы арқылы болжамды және шынайы мәндер арасындағы алшақтықты анықтауға шолу жасалды.

AFINN, логистикалық регрессия, Multinomial Naive Bayes, сызықтық SVM, XGBoost, Long short-term memory (LSTM), LSTM жақсарту ұшін LSTM-pack sequence оқыту әдістері қолдана отырып, берілген мәтінге сентименталды талдау жасауға болатын веб-қосымша әзірленді. JavaScript тілінде жазылған және Python кітапханаларын веб-қосымшаға қосу үшін Flask пайдалана отырып веб-қосымша жасалды. Пайдаланушы сөзді немесе

сөйлемдерді енгізеді, содан кейін көңіл-күйді анықтау түймешігін басып, әрбір әдіс блогы үшін нәтижелерді алады. Деректер жинағы көп болмағандықтан машиналық оқыту әдістері дәлірек нәтиже беретінін көрсетті. Логистикалық регрессия үлгісі төмен өлшемді деректер болса және олардың мүмкіндігі сызықты түрде бөлінетін болса, тиімді болуы мүмкін, бірақ жақсы нәтижелерге қол жеткізу үшін үлкен деректер жиынтық қажет етіледі.

Деректерді қатесіз талдау мүмкін емес, тілдің барлық мүмкіндіктерін ескеріп, алдын ала өңдеуді жақсарып және үлкен деректер жинақтарында эксперименттер жүргізу қажет.

ҚЫСҚАРТЫЛҒАН ЖӘНЕ ТҮСІНІКТЕМЕЛІК СӨЗДЕР

AFINN – Ағылшын сөздеріне арналған аффективті нормалар

TF-IDF (Term Frequency - Inverse Document Frequency) – жинақтағы немесе корпустағы құжат үшін сөздің маңыздылығын көрсететін сандық статистика

RNN (Recurrent Neural Network) – қайталанатын нейрондық желі

SVM (Support Vector Machine) – қолдау векторлық машина

ReactJS – Пайдаланушы интерфейстерін құруға арналған декларативті, тиімді және икемді JavaScript кітапханасы

LSTM (Long Short-Term Memory) – табиғи тілді өңдеу тапсырмаларында, соның ішінде сентименталды талдауда жиі қолданылатын нейрондық желі архитектурасының бір түрі

CSV (comma-separated values) – үтірмен бөлінген мәндер файлы

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1. Pasolini, Roberto. Learning Methods and Algorithms for Semantic Text Classification across Multiple Domains, [Dissertation thesis], Alma Mater Studiorum Università di Bologna. Dottorato di ricerca in Ingegneria elettronica, informatica e delle telecomunicazioni, 27 Ciclo. DOI 10.6092/unibo/amsdottorato/7058, 2015.
2. Большакова Е.И., Воронцов К.В., Ефремова Н.Э., Клышинский Э.С., Лукашевич Н.В., Сапин А.С. Автоматическая обработка текстов на естественном языке и анализ данных : учеб. пособие, — М.: Изд-во НИУ ВШЭ, 2017. — 269 с
3. Gulmira Bekmanova, Gaziza Yelibayeva, Saltanat Aubakirova, Nurgul Dyussupova, Altynbek Sharipbay, Rozamgul Nyazova, Methods for analyzing polarity of the Kazakh texts related to the terrorist threats, 2019
4. Dinara Gimadi, Richard Evans, Kiril Simov, Web-sentiment Analysis Of Public Comments (Public Reviews) For Languages With Limited Resources Such As The Kazakh Language
5. Darkhan Akhmed-Zaki, Madina Mansurova, Gulmira Madiyeva, Nurgali Kadyrbek & Marzhan Kyrgyzbayeva. Development of the information system for the Kazakh language preprocessing, Cogent Engineering, 8:1, 1896418, DOI: 10.1080/23311916.2021.1896418, 2021
6. Banu Yergesh, Gulmira Bekmanova, Altynbek A. Sentiment analysis of Kazakh text and their polarity, February 2019
7. Aru Omarali. Sentiment analysis and visualization of data from social networks using Machine learning algorithms, 2019
8. Huseyn Hasanli , Burak Ordin , Samir Rustamov. Sentiment analysis on twitter data for azerbaijani language, 2019
9. Sebastian Raschka. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition, 2019
10. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12, 2011 2825–2830.
11. Adam Paszke Sam Gross Soumith Chintala Gregory Chanan, Pytorch, https://github.com/pytorch/pytorch, 2023.
12. E. Grave, P. Bojanowski, P. Gupta, A. Joulin, T. Mikolov, Learning word vectors for 157 languages, in: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.
13. Chaturvedi I. Distunguishing between facts and opinions for sentiment analysis: Survey and challenges. Information Fusion, (44):65–77.
14. Steven Euijong Whang Yuji Roh, Geon Heo. A survey on data collection for machine learning. A Big Data - AI Integration Perspective

15. Savoy J Kummer O. Feature selection in sentiment analysis. 2000
16. Ruangkanokmas, P.; Achalakul, T.; Akkarajitsakul, K. Deep Belief Networks with Feature Selection forSentiment Classification. In Proceedings of the 2016 7th International Conference on Intelligent Systems,Modelling and Simulation (ISMS), Bangkok, Thailand, 25–27 January 2016; pp. 9–14
17. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.
18. I. Rabbimov, S. Kobilov, I. Mporas, Opinion classification via word and emoji embedding models with lstm, in: International Conference on Speech and Computer, Springer, 2021, pp. 589–601.
19. Vateekul, P.; Koomsubha, T. A study of sentiment analysis using deep learning techniques on Thai Twitterdata. In Proceedings of the 2016 13th International Joint Conference on Computer Science and SoftwareEngineering (JCSSE), Khon Kaen, Thailand, 13–15 July 2016; pp. 1–6
20. Hassan, A.; Mahmood, A. Deep learning approach for sentiment analysis of short texts. In Proceedingsof the Third International Conference on Control, Automation and Robotics (ICCAR), Nagoya, Japan,24–26 April 2017; pp. 705–710.

ҚОСЫМША А

Қазақ тіліндегі сентименталды деректерді талдаудың моделдері мен әдістерінің бағдарламалық жасақтама React листинг коды

```
function App() {
  const [input, setInput] = useState('')
  const [data, setDate] = useState();

  const onChange = (e) => setInput(e.target.value);
  const onSubmit = () => {
    const output = sentiment(input);
    setDate(output);
  }


  const handleStyle = (word) => {
    const pos = data.positive.find((d) => d.obj === word.toLowerCase());
    const neg = data.negative.find((d) => d.obj === word.toLowerCase());
    return {
      background: pos ? '#C7EED7' : neg ? '#FED6DB' : 'inherit',
      color: pos ? '#22BC61' : neg ? '#FA6A7D' : 'inherit'
    }
  }

  return (
    <main>
      <div className="container">
        <h1 className="title">Semantic Analysis for Kazakh language</h1>

        <div className="form">
          <div className="input-container">
            <h3 className="input__title">Test with your own text</h3>
            <form>
              <textarea className="textarea" value={input} onChange={onChange} />
              <button className="btn" type="button" onClick={onSubmit}>
                Classify text
              </button>
            </form>
          </div>
          <div className="results-container">
            <h3 className="results__title">Results</h3>
            <div className="results__header results__table">
```

```
            <span>Tag</span>
            <span>Confidence</span>
          </div>
          <div className="results">
            <div className="result__content results__table">
              {data && (
                <>
                  <span>{data.comparative > 0 ? 'Positive' : data.comparative < 0 ?
'Negative' : 'Neutral'}</span>
                  <span>{data.comparative.toFixed(2)}</span>
                </>
              )}
            </div>
          </div>
        </div>
      </div>

      {data && (
        <div className="output">
          <div className="text-container">
            <h3 className="text-title">Text</h3>
            <div className="text">
              {input.replace(/\n/g, ' ')
                .replace(/[.,\/#!?$%^&*;:{}=_`"~()]/g, ' ').split(' ').map((word) => {
                return <span style={handleStyle(word)}>{word}</span>
              })}
            </div>
          </div>
          <div className="word-cloud">
            <h3 className="word-cloud__title">Word Cloud</h3>

            <WordCloud
              data={data.all.map((w) => ({
                text: w.text,
                value: w.value * 20 + 100,
              }))}
              rotate={({value}) => Math.random() * value % 80 - 45}
              width={400}
              height={200}
              spiral="rectangular"
              fill={({ value }) => {
                console.log(value)
                return handleColor(value)
              }}
```

```
            fontSize={(word) => handleFontSize(word.value)}
          />
        </div>
      </div>
    )}
  </div>
 </main>
 )
}


const handleColor = (value) => {
  if (value < 20) {
    return '#9B2C2C'
  } else if (value >= 20 && value < 40) {
    return '#C53030'
  } else if (value >= 40 && value < 60) {
    return '#E53E3E'
  } else if (value >= 60 && value < 80) {
    return '#F56565'
  } else if (value >= 80 && value < 100) {
    return '#FC8181'
  } else if (value >= 100 && value < 120) {
    return '#7f8a96'
  } else if (value >= 120 && value < 140) {
    return '#68D391'
  } else if (value >= 140 && value < 160) {
    return '#48BB78'
  } else if (value >= 160 && value < 180) {
    return '#38A169'
  } else if (value >= 180 && value < 200) {
    return '#2F855A'
  } else if (value >= 200) {
    return '#276749'
  }
}


const handleFontSize = (value) => {
  if (value < 20) {
    return 30
  } else if (value >= 20 && value < 40) {
    return 28
  } else if (value >= 40 && value < 60) {
```

```
    return 26
  } else if (value >= 60 && value < 80) {
    return 24
  } else if (value >= 80 && value < 100) {
    return 22
  } else if (value >= 100 && value < 120) {
    return 10
  } else if (value >= 120 && value < 140) {
    return 22
  } else if (value >= 140 && value < 160) {
    return 24
  } else if (value >= 160 && value < 180) {
    return 26
  } else if (value >= 180 && value < 200) {
    return 28
  } else if (value >= 200) {
    return 30
  }
}

export default function (input) {
    return input
        .toLowerCase()
        .replace(/\n/g, ' ')
        .replace(/[.,\/#!?$%^&*;:{}=_`"~()]/g, ' ')
        .replace(/\s\s+/g, ' ')
        .trim()
        .split(' ');
};
```

ҚОСЫМША Ә

Қазақ тіліндегі сентименталды деректерді талдаудың моделдері мен әдістерінің бағдарламалық жасақтама Python листинг коды

```
import pandas as pd
import nltk

import pandas as pd
import nltk

from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re
from nltk.tokenize import word_tokenize
nltk.download('punkt')

import pandas as pd
data=pd.read_csv('kazakh_revs.csv')
data.head(10)
def clean_text(text):
    alphaPattern      = "[^\W\d_]"
    text = re.sub(r"[\W\d_]", " ", text)

    filtered_list = []
    stop_words = stopwords.words('kazakh')

    # my new custom stopwords
    my_extra = ['және', 'телефон', 'телефонды', 'оны', 'университет']
    # add the new custom stopwrds to my stopwords
    stop_words.extend(my_extra)
    # Tokenize the sentence
    words = word_tokenize(text)
    for w in words:
        if w.lower() not in stop_words:
            filtered_list.append(w)

    return ' '.join(filtered_list)

clean_text('Телефон жақсы екен. Және ұнады')

import numpy as np
import seaborn as sns
```

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud,STOPWORDS
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

### Count Plot
sns.set(style = "whitegrid" , font_scale = 1.2)
sns.countplot(data.sentiment,palette = ['green','red'],order = [1,0])
plt.xticks(ticks = np.arange(2),labels = ['positive','negative'])
plt.title('Target count for phone reviews')
plt.show()

#word cloud for positive reviews
positive_data = data[data.sentiment == 1]['review']
positive_data_string = ' '.join(positive_data)
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000, width=1200,
height=600,background_color="white").generate(positive_data_string)
plt.imshow(wc)
plt.axis('off')
plt.title('Word cloud for positive reviews',fontsize = 20)
plt.show()
#word cloud for negative reviews
negative_data = data[data.sentiment == 0]['review']
negative_data_string = ' '.join(negative_data)
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000, width=1200,
height=600,background_color="white").generate(negative_data_string)
plt.imshow(wc , interpolation = 'bilinear')
plt.axis('off')
plt.title('Word cloud for negative reviews',fontsize = 20)
plt.show()

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

from sklearn.metrics import classification_report, accuracy_score, confusion_matrix,
plot_confusion_matrix, plot_roc_curve, plot_precision_recall_curve

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from xgboost.sklearn import XGBClassifier

#splitting into train and test
```

```
train, test= train_test_split(data, test_size=0.2, random_state=42)
Xtrain, ytrain = train['review'], train['sentiment']
Xtest, ytest = test['review'], test['sentiment']
print(train, test)

#Vectorizing data

tfidf_vect = TfidfVectorizer() #tfidfVectorizer
Xtrain_tfidf = tfidf_vect.fit_transform(Xtrain)
Xtest_tfidf = tfidf_vect.transform(Xtest)


count_vect = CountVectorizer() # CountVectorizer
Xtrain_count = count_vect.fit_transform(Xtrain)
Xtest_count = count_vect.transform(Xtest)

lr = LogisticRegression()
lr.fit(Xtrain_tfidf,ytrain)
p1=lr.predict(Xtest_tfidf)
s1=accuracy_score(ytest,p1)
print(classification_report(ytest, p1))
print("Logistic Regression Accuracy :", "{:.2f}%".format(100*s1))
plot_confusion_matrix(lr, Xtest_tfidf, ytest,cmap = 'Blues')
plt.grid(False)

LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)
LRmodel.fit(Xtrain_tfidf, ytrain)
def predict(vectoriser, model, text):
    # Predict the sentiment
    textdata = vectoriser.transform((text))
    sentiment = model.predict(textdata)

    # Make a list of text with sentiment.
    data = []
    for text, pred in zip(text, sentiment):
        data.append((text,pred))

    # Convert the list into a Pandas DataFrame.
    df = pd.DataFrame(data, columns = ['text','sentiment'])
    df = df.replace([0,1], ["Negative","Positive"])
    return df

mnb= MultinomialNB()
mnb.fit(Xtrain_tfidf,ytrain)
```

```
p2=mnb.predict(Xtest_tfidf)
s2=accuracy_score(ytest,p2)
print(classification_report(ytest, p2))
print("Multinomial Naive Bayes Classifier Accuracy :", "{:.2f}%".format(100*s2))
plot_confusion_matrix(mnb, Xtest_tfidf, ytest,cmap = 'Blues')
plt.grid(False)

linear_svc = LinearSVC(penalty='l2',loss = 'hinge')
linear_svc.fit(Xtrain_tfidf,ytrain)
p3=linear_svc.predict(Xtest_tfidf)
s3=accuracy_score(ytest,p3)
print(classification_report(ytest, p3))
print("Linear Support Vector Classifier Accuracy :", "{:.2f}%".format(100*s3))
plot_confusion_matrix(linear_svc, Xtest_tfidf, ytest,cmap = 'Blues')
plt.grid(False)

xgbo = XGBClassifier()
xgbo.fit(Xtrain_tfidf,ytrain)
p4=xgbo.predict(Xtest_tfidf)
s4=accuracy_score(ytest,p4)
print(classification_report(ytest, p4))
print("XGBoost Accuracy :", "{:.2f}%".format(100*s4))
plot_confusion_matrix(xgbo, Xtest_tfidf, ytest, cmap = 'Blues')
plt.grid(False)

import torch
import torch.nn.functional as F
import torchtext
import time
import random
import pandas as pd

torch.backends.cudnn.deterministic = True

RANDOM_SEED = 123
torch.manual_seed(RANDOM_SEED)

VOCABULARY_SIZE = 200
LEARNING_RATE = 0.005
BATCH_SIZE = 128
NUM_EPOCHS = 30
DEVICE = torch.device('cuda:1' if torch.cuda.is_available() else 'cpu')

EMBEDDING_DIM = 128
```

```
HIDDEN_DIM = 256
NUM_CLASSES = 2

class RNN(torch.nn.Module):

    def __init__(self, input_dim, embedding_dim, hidden_dim, output_dim):
        super().__init__()

        self.embedding = torch.nn.Embedding(input_dim, embedding_dim)
        #self.rnn = torch.nn.RNN(embedding_dim,
        #                        hidden_dim,
        #                        nonlinearity='relu')
        self.rnn = torch.nn.LSTM(embedding_dim,
                                 hidden_dim)

        self.fc = torch.nn.Linear(hidden_dim, output_dim)


    def forward(self, text):
        # text dim: [sentence length, batch size]

        embedded = self.embedding(text)
        # embedded dim: [sentence length, batch size, embedding dim]

        output, (hidden, cell) = self.rnn(embedded)
        # output dim: [sentence length, batch size, hidden dim]
        # hidden dim: [1, batch size, hidden dim]

        hidden.squeeze_(0)
        # hidden dim: [batch size, hidden dim]

        output = self.fc(hidden)
        return output

torch.manual_seed(RANDOM_SEED)
model = RNN(input_dim=len(TEXT.vocab),
            embedding_dim=EMBEDDING_DIM,
            hidden_dim=HIDDEN_DIM,
            output_dim=NUM_CLASSES # could use 1 for binary classification
)

model = model.to(DEVICE)
optimizer = torch.optim.Adam(model.parameters(), lr=0.005)
```

```
def compute_accuracy(model, data_loader, device):

    with torch.no_grad():

        correct_pred, num_examples = 0, 0

        for i, (features, targets) in enumerate(data_loader):

            features = features.to(device)
            targets = targets.float().to(device)

            logits = model(features)
            _, predicted_labels = torch.max(logits, 1)

            num_examples += targets.size(0)
            correct_pred += (predicted_labels == targets).sum()
    return correct_pred.float()/num_examples * 100

start_time = time.time()

for epoch in range(NUM_EPOCHS):
    model.train()
    for batch_idx, batch_data in enumerate(train_loader):

        text = batch_data.review.to(DEVICE)
        labels = batch_data.sentiment.to(DEVICE)

        ### FORWARD AND BACK PROP
        logits = model(text)
        loss = F.cross_entropy(logits, labels)
        optimizer.zero_grad()

        loss.backward()

        ### UPDATE MODEL PARAMETERS
        optimizer.step()

        ### LOGGING
        if not batch_idx % 50:
            print (f'Epoch: {epoch+1:03d}/{NUM_EPOCHS:03d} | '
                   f'Batch {batch_idx:03d}/{len(train_loader):03d} | '
                   f'Loss: {loss:.4f}')

    with torch.set_grad_enabled(False):
```

```
        print(f'training accuracy: '
              f'{compute_accuracy(model, train_loader, DEVICE):.2f}%'
              f'\nvalid accuracy: '
              f'{compute_accuracy(model, valid_loader, DEVICE):.2f}%')

    print(f'Time elapsed: {(time.time() - start_time)/60:.2f} min')

print(f'Total Training Time: {(time.time() - start_time)/60:.2f} min')
print(f'Test accuracy: {compute_accuracy(model, test_loader, DEVICE):.2f}%')

class RNN(torch.nn.Module):

    def __init__(self, input_dim, embedding_dim, hidden_dim, output_dim):
        super().__init__()

        self.embedding = torch.nn.Embedding(input_dim, embedding_dim)
        self.rnn = torch.nn.LSTM(embedding_dim,
                                 hidden_dim)

        self.fc = torch.nn.Linear(hidden_dim, output_dim)


    def forward(self, text, text_length):
        embedded = self.embedding(text)

        ## NEW
        packed = torch.nn.utils.rnn.pack_padded_sequence(embedded,
text_length.to('cpu'))

        packed_output, (hidden, cell) = self.rnn(packed)

        hidden.squeeze_(0)

        output = self.fc(hidden)
        return output

def compute_accuracy(model, data_loader, device):

    with torch.no_grad():

        correct_pred, num_examples = 0, 0

        for batch_idx, batch_data in enumerate(data_loader):
```

```
            # NEW
            features, text_length = batch_data.review
            targets = batch_data.sentiment.to(DEVICE)

            logits = model(features, text_length)
            _, predicted_labels = torch.max(logits, 1)

            num_examples += targets.size(0)

            correct_pred += (predicted_labels == targets).sum()
        return correct_pred.float()/num_examples * 100

    start_time = time.time()

    for epoch in range(NUM_EPOCHS):
        model.train()
        for batch_idx, batch_data in enumerate(train_loader):

            # NEW
            features, text_length = batch_data.review
            labels = batch_data.sentiment.to(DEVICE)

            ### FORWARD AND BACK PROP
            logits = model(features, text_length)
            loss = F.cross_entropy(logits, labels)
            optimizer.zero_grad()

            loss.backward()

            ### UPDATE MODEL PARAMETERS
            optimizer.step()

            ### LOGGING
            if not batch_idx % 50:
                print (f'Epoch: {epoch+1:03d}/{NUM_EPOCHS:03d} | '
                       f'Batch {batch_idx:03d}/{len(train_loader):03d} | '
                       f'Loss: {loss:.4f}')

        with torch.set_grad_enabled(False):
            print(f'training accuracy: '
                  f'{compute_accuracy(model, train_loader, DEVICE):.2f}%'
                  f'\nvalid accuracy: '
                  f'{compute_accuracy(model, valid_loader, DEVICE):.2f}%')
```

```
    print(f'Time elapsed: {(time.time() - start_time)/60:.2f} min')

print(f'Total Training Time: {(time.time() - start_time)/60:.2f} min')
print(f'Test accuracy: {compute_accuracy(model, test_loader, DEVICE):.2f}%')
```