10-дәріс Табиғи тілдерді өңдеу NLP

Пәні: Интеллектуалды ақпараттық жүйелер

Алдыңғы тақырып: Нейрондық желілер және терең оқыту

Бұл дәріс табиғи тілдегі мәтінді компьютерлік өңдеудің толық жолын түсіндіреді. Материал 25 слайдпен бірдей ретте құрылған. Мен әр бөлімде жаңа ұғымды алдыңғы оймен байланыстырып, формуладан кейін қарапайым мысал беремін.

Дәрістің мақсаты

Студенттерге мәтінді дайындау, токенизациялау, сандық векторға айналдыру, классификациялау және генерациялау қағидаларын түсіндіру; NLP модельдерінің сапасын бағалау мен жауапты қолдану талаптарын нақты мысалдар арқылы көрсету.

Күтілетін оқу нәтижелері

- NLP ұғымын және негізгі қолдану салаларын түсіндіреді
- мәтінді тазалау, токенизация және қалыпқа келтіру кезеңдерін ажыратады
- bag of words, TF-IDF және эмбеддинг тәсілдерін салыстырады
- RNN, attention және transformer логикасын байланыстырады
- классификация метрикаларын қате салдарына қарай таңдайды
- қазақ тіліндегі NLP ерекшеліктері мен жауапты қолдану талаптарын негіздейді

Негізгі ұғымдардың қысқаша картасы

Деңгей	Нені зерттейді	Мысал	NLP міндеті
Морфология	Сөздің құрылымы	оқушы лар ымыз ға	Лемматизация
Синтаксис	Сөздердің байланысы	Студент өтініш берді	Талдау
Семантика	Мағына	банк сөзінің мағынасы	Ұқсастық
Прагматика	Жағдайдағы мағына	Керемет тағы да істемеді	Сезімді талдау

Табиғи тілдерді өңдеу ұғымы

Адам үшін сөзді оқу, сөйлемнің мағынасын түсіну және сұраққа жауап беру күнделікті әрекет болып көрінеді. Компьютер үшін мәтін алдымен таңбалар тізбегі ғана. Сондықтан біз тілдік ақпаратты сандық түрде көрсету жолын табуымыз керек.

NLP компьютерге адам тіліндегі мәтінді немесе сөйлеуді талдауға, мағынасын шамамен анықтауға және пайдалы нәтиже шығаруға мүмкіндік береді. Бұл бағыт лингвистиканы, машиналық оқытуды және бағдарламалауды біріктіреді. Дәріс барысында мен ережелік тәсілден бастап трансформерлерге дейінгі жолды бірізді көрсетемін.

Қарапайым мысал. Мысалы, университет сайтына «Жатақханаға өтінішті қалай беремін?» деген сұрақ түссе, NLP жүйесі сұрақтың жатақхана туралы екенін анықтап, тиісті нұсқаулықты ұсына алады.

Негізгі ой: табиғи тіл адамға ыңғайлы, ал компьютер оны өңдеу үшін мәтінді құрылымдалған сандық көрініске айналдырады.

Дәрістің мақсаты және оқу нәтижелері

Енді мақсатымызды нақтылап алайық.

Алдымен бүгінгі жұмыстың бағытын белгілеймін. Менің мақсатым тек терминдерді атау емес, NLP жүйесінің мәтінді қалай қабылдайтынын, қалай санға айналдыратынын және қалай шешім шығаратынын түсіндіру.

Дәріс соңында сіздер мәтінді алдын ала өңдеу кезеңдерін ажырата аласыздар, bag of words пен TF-IDF логикасын түсіндіресіздер, сөз эмбеддингінің мағынасын ашасыздар және RNN мен transformer арасындағы айырмашылықты негіздей аласыздар. Сонымен бірге модель сапасын бағалау мен жауапты қолдану талаптарын атайсыздар.

Қарапайым мысал. Мысалы, пікірлерді «оң» және «теріс» деп бөлу тапсырмасы берілсе, сіздер деректі тазалау, белгілеу, модельді оқыту, метриканы таңдау және қателерді талдау қадамдарын жоспарлай аласыздар.

Негізгі ой: біз NLP моделін жеке алгоритм емес, дерек, модель, бағалау және адам бақылауынан тұратын толық жүйе ретінде қараймыз.

NLP не үшін қажет

Осы түсініктен кейін қолдану қажеттілігіне көшемін.

Енді NLP-дің практикалық қажеттілігіне көшейін. Ұйымдарда өтініштер, хаттар, пікірлер, есептер және анықтамалар түрінде үлкен көлемде мәтін жиналады. Олардың барлығын қолмен оқу көп уақыт алады, ал маңызды ақпарат кеш байқалуы мүмкін.

NLP мәтінді іздеу, тақырып бойынша бөлу, қысқаша мазмұндау, сұраққа жауап беру, аудару және эмоциялық реңкті анықтау сияқты міндеттерді автоматтандырады. Дегенмен автоматтандыру шешімді толықтай машинаға беру дегенді білдірмейді. Қате салдары жоғары жағдайда маман нәтижені тексеруі керек.

Қарапайым мысал. Мысалы, деканатқа бір күнде 500 өтініш келсе, жүйе оларды «анықтама», «жатақхана», «төлем» және «академиялық мәселе» санаттарына бөледі. Қызметкер әр өтінішті нөлден іздемей, дайын кезекпен жұмыс істейді.

Негізгі ой: NLP мәтіндік ағынды реттейді және маманның уақытын үнемдейді, бірақ жауапкершілікті жоймайды.

Табиғи тілдің күрделілігі

Практикалық міндеттерді көрген соң тілдің күрделілігін ашамын.

NLP-ді түсіну үшін тілдің неге күрделі екенін білу маңызды. Бір сөз бірнеше мағына беруі мүмкін, бір мағына әртүрлі сөзбен айтылуы мүмкін, ал сөйлемнің мағынасы контекстке тәуелді болады. Адам бұл айырмашылықты өмірлік тәжірибесі арқылы сезеді.

Тілде омоним, синоним, астарлы мағына, қысқарту, қате жазу және аралас тіл жиі кездеседі. Қазақ тілінде жалғаулар арқылы сөздің көптеген формасы жасалады. Сондықтан сөзді дәл сол күйінде салыстыру жеткіліксіз.

Қарапайым мысал. Мысалы, «банк» сөзі қаржы ұйымын білдіруі мүмкін, ал «өзен банкі» деп қате аударылған мәтінде жаға мағынасын беруі ықтимал. «Ол кітапты үстелге қойды, кейін оны алды» сөйлеміндегі «оны» сөзінің қай нысанға қатысты екенін контекст анықтайды.

Негізгі ой: тілдегі мағына жеке сөзден ғана емес, оның формасынан, орны мен контекстінен құралады.

NLP жүйесінің жалпы құбыры

Бұл қиындықтар бізді жүйелі өңдеу құбырына әкеледі.

Осы күрделілікті басқару үшін мен NLP жобасын бірнеше байланысты кезеңге бөлемін. Алдымен міндетті нақтылаймын, кейін мәтінді жинаймын және тексеремін. Содан соң мәтінді өңдеп, сандық белгілерге айналдырамын, модельді оқытамын және жаңа деректе бағалаймын.

Жалпы тізбек былай көрінеді: мәселе қою, дерек жинау, тазалау, токенизация, сандық көрініс, модель, бағалау және енгізу. Әр кезең келесі кезеңнің сапасына әсер етеді. Нашар белгіленген деректі күрделі модель өздігінен түзете алмайды.

Қарапайым мысал. Мысалы, пікірлерді талдау жобасында «бейтарап» пікірлерді алдын ала қалай белгілейтінімізді шешпесек, екі сарапшы бір мәтінге әртүрлі жауап береді. Мұндай белгісіздік модельге де өтеді.

Негізгі ой: NLP нәтижесі модельден ғана емес, бүкіл өңдеу құбырының сапасынан тәуелді.

Мәтінді тазалау және қалыпқа келтіру

Бұл бөлім алдыңғы тәсілдің мүмкіндігі мен шектеуін толықтырады.

Енді мәтінді дайындаудың бірінші техникалық қадамын қарастырамын. Тазалау кезінде қажетсіз HTML белгілерін, қайталанған бос орындарды және техникалық

таңбаларды реттейміз. Қалыпқа келтіру кезінде регистр, тыныс белгісі және жазылу нұсқалары жөнінде саналы шешім қабылдаймыз.

Барлық белгіні автоматты түрде алып тастау дұрыс емес. Сұрау белгісі сөйлемнің сұрақ екенін, леп белгісі эмоцияны, ал бас әріп атауды көрсетуі мүмкін. Сондықтан өңдеу ережесі нақты міндетке сәйкес болуы керек.

Қарапайым мысал. Мысалы, «Керемет!!!» пікіріндегі леп белгілерін толық жойсақ, эмоцияның күшін жоғалтамыз. Ал іздеу жүйесінде «Университет» пен «университет» сөздерін бір қалыпқа келтіру пайдалы болуы мүмкін.

Негізгі ой: мәтінді тазалау ақпаратты механикалық жою емес, пайдалы сигналды сақтай отырып шуды азайту.

Токенизация

Енді сұрақты сәл тереңдетіп қарастырамын.

Тазаланған мәтінді модельге беру үшін оны шағын бірліктерге бөлемін. Бұл процесс токенизация деп аталады. Токен сөз, сөз бөлігі, таңба немесе сөйлем болуы мүмкін. Таңдау модель мен тілге байланысты.

Сөздік токенизация түсінікті, бірақ сирек сөздер мен жалғаулы формалар сөздікті үлкейтеді. Сөз бөлігіне негізделген токенизация жиі кездесетін бөлшектерді сақтап, жаңа сөздерді де құрастыра алады. Қазіргі трансформерлер көбіне осы тәсілді қолданады.

Қарапайым мысал. Мысалы, «оқушыларымызға» сөзін бір токен ретінде алуға болады. Сөз бөлігі тәсілі оны «оқушы», «лар», «ымыз», «ға» сияқты бөліктерге ажыратуы мүмкін. Нақты бөліну қолданылған сөздік алгоритміне тәуелді.

Негізгі ой: токенизация мәтін мен сандық модель арасындағы алғашқы көпірді құрады.

Аудиторияға сұрақ: қазақ тіліндегі ұзын сөзді бір токен ретінде сақтау қандай қиындық тудырады?

Стоп сөздер, стемминг және лемматизация

Енді алдыңғы ойды келесі ұғыммен байланыстырамын.

Келесі қадамда сөз формаларын қалай ықшамдайтынымызды түсіндіремін. Стоп сөздер өте жиі кездесетін қызметтік сөздер болуы мүмкін. Стемминг сөздің соңын ереже бойынша қысқартады, ал лемматизация сөзді негізгі сөздік формаға келтіруге тырысады.

Стоп сөздерді жою кейбір іздеу есептерінде пайдалы, бірақ сезімді немесе мағыналық байланысты бұзуы мүмкін. «Жақсы емес» тіркесінен «емес» сөзін алып тастасақ, мағына кері өзгереді. Қазақ тілі сияқты морфологиясы бай тілде лемматизация сапалы тілдік талдауды талап етеді.

Қарапайым мысал. Мысалы, «оқыдым», «оқиды», «оқыған» формаларын бір «оқу» леммасымен байланыстыру құжаттарды тақырып бойынша салыстыруды жеңілдетеді.

Негізгі ой: қалыпқа келтіру әдісі міндетке сай таңдалады, себебі әр қысқарту пайдалы ақпаратты жоғалтуы мүмкін.

Мәтінді сандық түрде көрсету

Осыны түсінгеннен кейін келесі қадамға өтемін.

Енді негізгі сұраққа келемін: компьютер сөзді қалай санға айналдырады? Ең қарапайым тәсілде біз сөздік құрамыз және әр құжатта қай сөз неше рет кездескенін санаймыз. Бұл bag of words деп аталады.

Bag of words сөздердің ретін елемейді, бірақ тақырыптық белгілерді жақсы көрсете алады. Вектордың әр өлшемі сөздікке кіретін бір сөзге сәйкес келеді. Сөздік өте үлкен болса, вектор сирек және көлемді болады.

Қарапайым мысал. Мысалы, сөздік «курс, жақсы, қиын» деген үш сөзден тұрсын. «Курс жақсы» мәтіні [1, 1, 0], ал «курс қиын» мәтіні [1, 0, 1] векторы арқылы беріледі.

Негізгі ой: сандық көрініс мәтінді математикалық амалдар қолдануға болатын векторға айналдырады.

Bag of words және n граммалар

Бұл бөлім алдыңғы тәсілдің мүмкіндігі мен шектеуін толықтырады.

Bag of words қарапайым болғанымен, сөздердің ретін жоғалтады. Мұны азайту үшін қатар тұрған n токеннен тұратын n-граммаларды қолданамыз. Бір сөз униграмма, екі сөз биграмма, үш сөз триграмма деп аталады.

Биграмма «жақсы емес» сияқты тіркесті жеке белгі ретінде сақтайды. Бірақ n артқан сайын мүмкін тіркестер саны тез өседі және сирек комбинациялар көбейеді. Сондықтан біз пайдалы контекст пен модель көлемі арасында теңгерім іздейміз.

Қарапайым мысал. Мысалы, «қызмет жақсы емес» сөйлемінің униграммалары «қызмет», «жақсы», «емес» болады. Биграммалар «қызмет жақсы» және «жақсы емес» болып, терістеуді жақсырақ көрсетеді.

Негізгі ой: n-грамма қысқа контексті сақтайды, бірақ өлшемділікті үлкейтеді.

TF IDF

Енді сұрақты сәл тереңдетіп қарастырамын.

Сөз санының бір кемшілігі бар: барлық құжатта жиі кездесетін сөздер маңызды болып көрінуі мүмкін. TF-IDF осы мәселені азайтады. TF сөздің нақты құжаттағы жиілігін, ал IDF сөздің бүкіл жинақта қаншалықты сирек екенін өлшейді.

Формуласы: TF-IDF(t,d) = TF(t,d) × log(N / DF(t)). Мұнда N — құжаттар саны, DF(t) — t сөзі кездесетін құжаттар саны. Сөз бір құжатта жиі, бірақ барлық құжатта сирек кездессе, оның салмағы жоғары болады.

Қарапайым мысал. Мысалы, барлық өтініште «өтініш» сөзі бар болса, ол санатты ажыратуға көмектеспейді. Ал «жатақхана» сөзі тек бір топ құжатта кездессе, сол тақырып үшін маңызды белгі болады.

Негізгі ой: TF-IDF құжатқа тән сөздерді жалпы жиі сөздерден жоғары қояды.

TF-IDF(t,d) = TF(t,d) × log(N / DF(t))

Аудиторияға сұрақ: барлық құжатта кездесетін сөздің IDF салмағы неге төмен болады?

Векторлар ұқсастығы

Енді алдыңғы ойды келесі ұғыммен байланыстырамын.

Құжаттарды векторға айналдырған соң олардың ұқсастығын есептей аламыз. Кең тараған өлшемдердің бірі — косинустық ұқсастық. Ол екі вектор арасындағы бұрышты салыстырады және мәтін ұзындығының әсерін азайтады.

Формуласы: cos(θ) = (A · B) / (||A|| × ||B||). Нәтиже 1-ге жақындаған сайын бағыттар ұқсас болады. TF-IDF векторларымен бірге бұл тәсіл іздеу мен ұқсас құжаттарды табуда жиі қолданылады.

Қарапайым мысал. Мысалы, «жатақханаға орын керек» және «студенттер үйінен бөлме қажет» сөйлемдерінде дәлме-дәл сөздер аз. Қарапайым TF-IDF оларды әлсіз байланыстыруы мүмкін, ал мағыналық эмбеддинг ұқсастықты жақсырақ табады.

Негізгі ой: ұқсастық өлшемі сандық векторларды мазмұндық жақындық ретінде түсіндіруге мүмкіндік береді.

cos(θ) = (A · B) / (||A|| × ||B||)

Сөз эмбеддингтері

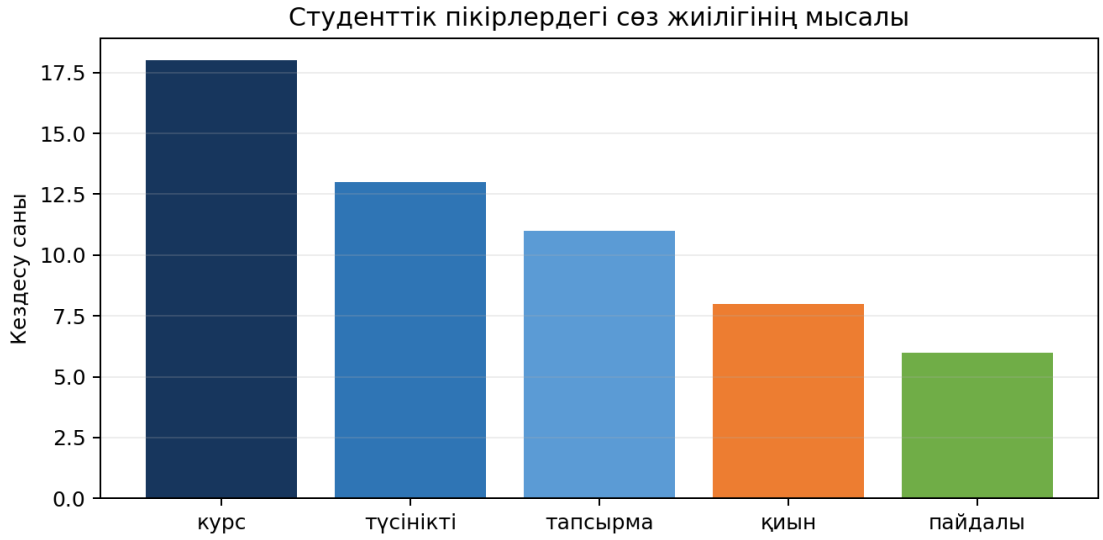
Осыны түсінгеннен кейін келесі қадамға өтемін.

Bag of words сөздерді бір-бірінен тәуелсіз белгілер деп қарайды. Эмбеддинг әр сөзді ықшам тығыз векторға айналдырып, мағынасы ұқсас сөздерді кеңістікте жақын орналастыруға тырысады.

Word2Vec және GloVe сияқты классикалық әдістер сөздің мағынасын оның айналасындағы сөздерден үйренеді. Бұл идеяны «сөзді жанындағы сөздері арқылы танимыз» деп түсіндіруге болады. Алайда бір сөзге бір ғана вектор берілсе, көпмағыналылық толық шешілмейді.

Қарапайым мысал. Мысалы, «оқытушы» мен «мұғалім» векторлары жақын болуы мүмкін. Бірақ «алма» сөзі жеміс пе, әлде «алма» етістігінің болымсыз түрі ме деген айырмашылықты тұрақты эмбеддинг контекстсіз ажырата алмайды.

Негізгі ой: эмбеддинг сөздің мағыналық қатынасын сандық кеңістікте көрсетеді.



1-сурет. Сөз жиілігінің оқу мысалы

Слайд 14. Мәтінді классификациялау

Бұл бөлім алдыңғы тәсілдің мүмкіндігі мен шектеуін толықтырады.

Мәтінді сандық түрде көрсеткеннен кейін классификация міндетін шешуге болады. Мұнда әр мәтінге алдын ала анықталған класс беріледі. Класс тақырып, эмоция, өтініш түрі немесе қауіп белгісі болуы мүмкін.

Классикалық құбырда TF-IDF белгілерін логистикалық регрессияға, наивті Байеске немесе тірек векторлар әдісіне береміз. Терең оқытуда эмбеддинг пен нейрондық желі белгілерді бірге үйренеді. Қарапайым базалық модельді күрделі модельмен салыстыру маңызды.

Қарапайым мысал. Мысалы, қолдау қызметіндегі хабарламаларды «техникалық ақау», «төлем» және «есептік жазба» деп бөлеміз. Бір хабарлама бірнеше тақырыпқа қатысты болса, көпбелгілі классификация қажет.

Негізгі ой: классификация мәтінді алдын ала берілген санаттардың біріне немесе бірнешеуіне жатқызады.

Аудиторияға сұрақ: үш кластың орнына бір мәтінге бірнеше белгі беру қажет болатын жағдайды атаңыз.

Сезімді талдау

Енді сұрақты сәл тереңдетіп қарастырамын.

Сезімді талдау мәтіндегі бағалау реңкін анықтайды. Ең қарапайым жағдайда модель оң, теріс және бейтарап кластарын болжайды. Нақты жүйеде эмоцияның күші, нысаны және себептері де маңызды болуы мүмкін.

Бұл міндетте сарказм, терістеу және мәдени контекст қиындық туғызады. «Керемет, тағы да жүйе істемей қалды» сөйлеміндегі «керемет» сөзі оң болғанымен, жалпы мағына теріс. Сондықтан жеке сөздерге ғана сүйену жеткіліксіз.

Қарапайым мысал. Мысалы, университет асханасы туралы пікірлерді талдағанда «дәмді, бірақ кезек ұзақ» мәтінінде тағамға оң, қызмет жылдамдығына теріс баға бар. Аспектілік сезім талдауы осы екі бөлікті бөледі.

Негізгі ой: сезімді талдау жалпы реңкті ғана емес, пікірдің қай нысанға бағытталғанын да ескеруі керек.

Атаулы мәндерді тану

Енді алдыңғы ойды келесі ұғыммен байланыстырамын.

Келесі маңызды міндет — атаулы мәндерді тану, яғни NER. Жүйе мәтіннен адам атын, ұйымды, орынды, күнді, соманы және басқа маңызды нысандарды табады. Нәтижесінде еркін мәтін құрылымдалған жазбаға айналады.

NER әдетте әр токенге белгі береді. B-PER адам атының басталуын, I-PER жалғасын, O атаулы мәнге жатпайтын сөзді көрсетуі мүмкін. Белгілер жиыны нақты жобаға сәйкес кеңейтіледі.

Қарапайым мысал. Мысалы, «Айдана 18 қазанда Астанаға барады» сөйлемінен PERSON=Айдана, DATE=18 қазан, LOCATION=Астана мәндерін аламыз.

Негізгі ой: NER мәтіндегі маңызды нысандарды тауып, оларды белгіленген өрістерге айналдырады.

Тізбекті модельдер RNN және LSTM

Осыны түсінгеннен кейін келесі қадамға өтемін.

Тілде сөздердің реті маңызды болғандықтан, тізбекті модельдер пайда болды. RNN алдыңғы қадамдағы жасырын күйді келесі қадамға береді. Осылайша модель бұрынғы токендер туралы ақпаратты сақтауға тырысады.

Қарапайым RNN ұзақ аралықтағы байланысты ұмытуы мүмкін. LSTM мен GRU арнайы қақпалар арқылы қандай ақпаратты сақтау немесе ұмыту керегін үйренеді. Олар аударма, мәтін классификациясы және уақыттық тізбек есептерінде кең қолданылды.

Қарапайым мысал. Мысалы, «Фильм басында баяу болды, бірақ соңы өте ұнады» сөйлемінде соңғы баға жалпы пікірге күшті әсер етеді. Тізбекті модель сөздердің ретін ескереді.

Негізгі ой: RNN мен LSTM мәтінді бірінен кейін бірі келетін элементтер тізбегі ретінде өңдейді.

Attention механизмі

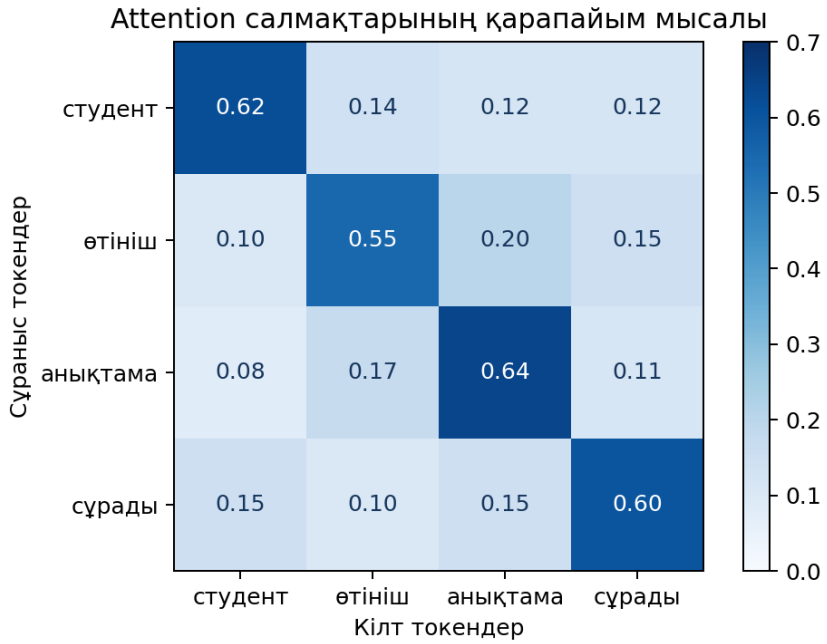
Бұл бөлім алдыңғы тәсілдің мүмкіндігі мен шектеуін толықтырады.

Ұзақ мәтіндегі барлық ақпаратты бір ғана жасырын күйге сыйғызу қиын. Attention механизмі модельге ағымдағы есеп үшін қай токендер маңызды екенін салмақ арқылы көрсетуге мүмкіндік береді.

Жалпы түрде attention салмақтары $\mathrm{softmax}(QK^T / \sqrt{d})$ арқылы есептеліп, кейін V мәндерімен біріктіріледі. Q сұранысты, K кілтті, V мәнді көрсетеді. Мұны кітаптан сұраққа жауап іздеумен салыстыруға болады: біз бүкіл мәтінді бірдей пайдаланбаймыз, сұраққа қатысты бөліктерге көбірек назар аударамыз.

Қарапайым мысал. Мысалы, «Студент өтінішті деканға берді, себебі ол анықтама сұрады» сөйлемінде «ол» сөзіне түсінік беру үшін модель «студент» сөзіне көбірек назар аударуы мүмкін.

Негізгі ой: attention контексттегі маңызды байланыстарға әртүрлі салмақ береді.



2-сурет. Attention салмақтарының шартты көрінісі

Аудиторияға сұрақ: attention салмағын толық түсіндірме деп қабылдауға бола ма?

Transformer архитектурасы

Енді сұрақты сәл тереңдетіп қарастырамын.

Attention идеясы transformer архитектурасының негізіне айналды. Transformer мәтін элементтерін параллель өңдеп, self-attention арқылы әр токеннің басқа токендермен байланысын есептейді. Токендердің реті позициялық ақпарат арқылы беріледі.

Encoder мәтінді түсінуге арналған көрініс құрады, decoder келесі токенді болжау арқылы мәтін жасайды. Кейбір модельдер тек encoder, кейбірі тек decoder, ал аударма жүйелері encoder-decoder құрылымын қолданады.

Қарапайым мысал. Мысалы, өтінішті жіктеу үшін encoder типті модель ыңғайлы. Ал сұраққа толық жауап құру үшін decoder типті генеративті модель қажет болуы мүмкін.

Негізгі ой: transformer ұзақ байланыстарды attention арқылы өңдейді және үлкен көлемде алдын ала оқытуға қолайлы.

Алдын ала оқыту және fine tuning

Енді алдыңғы ойды келесі ұғыммен байланыстырамын.

Қазіргі NLP жүйелерінде модельді нөлден оқыту міндетті емес. Үлкен мәтін жинағында алдын ала оқытылған модель тілдің жалпы заңдылықтарын меңгереді. Кейін оны шағын арнайы дерекпен нақты міндетке бейімдейміз.

Fine tuning кезінде модель параметрлерінің барлығы немесе бір бөлігі жаңартылады. Prompt арқылы бейімдеу кезінде тапсырма нұсқауы мәтін түрінде беріледі. Таңдау дерек көлеміне, есептеу ресурсына және құпиялық талаптарына байланысты.

Қарапайым мысал. Мысалы, жалпы қазақша және орысша мәтінде оқытылған көптілді модельді университет өтініштерінің 3000 белгіленген мысалымен санаттауға бейімдеуге болады.

Негізгі ой: алдын ала оқыту жалпы тілдік білім береді, ал fine tuning оны нақты салаға бейімдейді.

Генеративті тілдік модельдер

Осыны түсінгеннен кейін келесі қадамға өтемін.

Генеративті тілдік модель келесі токеннің ықтималдығын болжай отырып мәтін құрады. Ықтималдықты softmax функциясы береді: $P(token_i) = \exp(z_i) / \Sigma \exp(z_j)$. Модель әр қадамда ықтимал токендердің бірін таңдайды.

Мұндай модель мәтінді жалғастырады, қысқаша мазмұн жасайды, сұраққа жауап береді және нұсқаулық құрастырады. Бірақ ол шындықты тікелей тексермейді. Тілдік жағынан сенімді көрінетін қате жауап, яғни галлюцинация пайда болуы мүмкін.

Қарапайым мысал. Мысалы, модельден университеттің ішкі ережесін сұрасақ, ол нақты құжатқа қосылмаған жағдайда ойдан мерзім немесе талап айтуы мүмкін. Сондықтан маңызды жауапты ресми дерекпен байланыстыру керек.

Негізгі ой: генеративті модель ықтимал мәтін құрады, сондықтан оның жауабы дереккөз және адам тексеруімен толықтырылады.

Аудиторияға сұрақ: сирек қауіпті класта precision мен recall көрсеткіштерінің қайсысы маңыздырақ?

NLP сапасын бағалау

Бұл бөлім алдыңғы тәсілдің мүмкіндігі мен шектеуін толықтырады.

Модельді оқытқаннан кейін оның сапасын бөлек тест жиынында бағалаймын. Классификацияда accuracy, precision, recall және F1 қолданылады. Тең емес кластарда accuracy жоғары көрінгенімен, сирек маңызды класты модель өткізіп алуы мүмкін.

Аударма мен мәтін жасауда автоматты метрикалар пайдалы, бірақ олар мағынаның дұрыстығын толық өлшемейді. Сондықтан адам бағалауы, қате мысалдарын талдау және нақты қолдану жағдайындағы тексеру қажет.

Қарапайым мысал. Мысалы, қауіпті өтініштер барлық хабарламаның 2 пайызын құраса, бәрін «қауіпсіз» деген модель 98 пайыз accuracy алады. Бірақ оның recall мәні нөлге тең болады.

Негізгі ой: метрика қате салдарына сай таңдалады, ал орташа сан топтар мен мысалдар бойынша талдаумен толықтырылады.

Қате матрицасы және мысалдарды талдау

Енді сұрақты сәл тереңдетіп қарастырамын.

Қате матрицасы модельдің қай класты қай класспен шатастыратынын көрсетеді. Екі классты есепте біз true positive, false positive, true negative және false negative мәндерін қараймыз.

Тек жалпы метрикаға қарап тоқтамаймын. Қате мәтіндерді оқып, олардың қысқа болуы, аралас тілде жазылуы, қате жазылуы немесе сирек тақырыпқа жатуы сияқты себептерді іздеймін. Бұл талдау деректі немесе нұсқаулықты жақсартуға көмектеседі.

Қарапайым мысал. Мысалы, «төлем түспеді» хабарламасын жүйе техникалық ақауға жатқызса, себебі «түспеді» сөзі жүйелік қателерде де жиі кездесуі мүмкін. «Төлем» контекстін күшейту үшін қосымша мысал қажет.

Негізгі ой: қате матрицасы сандық көрініс береді, ал нақты мысалдар қателің себебін ашады.

Қазақ тіліндегі NLP ерекшеліктері

Енді алдыңғы ойды келесі ұғыммен байланыстырамын.

Енді қазақ тіліне тән ерекшеліктерді қарастырамын. Қазақ тілі агглютинативті болғандықтан, бір түбірге бірнеше жұрнақ пен жалғау қосылады. Бір сөз көптеген грамматикалық мағына бере алады. Бұл сөздік көлемін және сирек формалар санын арттырады.

Қазақша мәтінде кирилл және латын нұсқалары, орысша аралас сөздер, әріп қателері мен транслитерация кездеседі. Сапалы корпус пен белгіленген дерек көлемі ағылшын тіліне қарағанда аз болуы мүмкін. Сондықтан көптілді модель, морфологиялық талдау және деректі мұқият тексеру маңызды.

Қарапайым мысал. Мысалы, «өтініштеріңізді» сөзі «өтініш» түбіріне көптік, тәуелдік және табыс септік жалғауларын қосады. Сөз бөлігі токенизациясы мұндай формаларды тиімді өңдей алады.

Негізгі ой: қазақ тіліндегі NLP тілдің морфологиясын, жазу нұсқаларын және аралас тілді ескеруі керек.

Жауапты қолдану және қорытынды

Осыны түсінгеннен кейін келесі қадамға өтемін.

Соңғы бөлімде техникалық сапаны жауапты қолданумен байланыстырамын. Мәтінде жеке дерек, денсаулық, үлгерім немесе әлеуметтік жағдай туралы ақпарат болуы мүмкін. Оны жинау мен сақтау мақсаты анық және қолжетімділігі шектеулі болуы керек.

Модель кейбір тілдік топтарға немесе жазу стиліне нашар жұмыс істеуі мүмкін. Сондықтан сапаны тіл, диалект және пайдаланушы топтары бойынша тексереміз. Маңызды шешімде адамға түсіндіру, шағымдану және түзету мүмкіндігін береміз.

Қарапайым мысал. Мысалы, студенттің өтінішін модель қате санатқа жіберсе, қызметкер санатты өзгерте алуы және бұл түзету кейінгі сапаны жақсартуға пайдаланылуы керек. Автоматты жауап ресми шешім ретінде көрсетілмеуі тиіс.

Қорытынды ой: NLP мәтінді токендерге бөледі, сандық көрініс жасайды, модель арқылы заңдылық үйренеді және метрикамен тексеріледі. Сенімді жүйе техникалық дәлдікті дерек сапасымен, құпиялықпен және адам бақылауымен бірге қамтамасыз етеді.

Салыстыру кестелері

1-кесте. Мәтінді сандық көрсету тәсілдері

Тәсіл	Артықшылығы	Шектеуі	Қолдану мысалы
Bag of words	Қарапайым және түсінікті	Сөз реті жоғалады	Базалық классификация
TF-IDF	Маңызды сирек сөздерді күшейтеді	Терең мағынаны көрмейді	Құжат іздеу
Эмбеддинг	Мағыналық жақындықты көрсетеді	Дерек пен есептеу қажет	Ұқсастық және нейрондық модель
Контекстік эмбеддинг	Сөз мағынасын контекстпен өзгертеді	Ресурс шығыны жоғары	Transformer модельдері

2-кесте. NLP модельдерін салыстыру

Модель	Контекст	Артықшылығы	Шектеуі
TF-IDF және логистикалық регрессия	Қысқа n-грамма	Жылдам, түсіндіру оңай	Мағына шектеулі
RNN немесе LSTM	Тізбекті күй	Сөз ретін ескереді	Ұзақ мәтін баяу өңделеді
Transformer	Self-attention	Ұзақ байланысты жақсы өңдейді	Есептеу және дерек қажет

3-кесте. Классификация метрикалары

Метрика	Қарапайым түсінік	Қай кезде маңызды
Accuracy	Барлық болжамның қаншасы дұрыс	Кластар шамалас болғанда
Precision	Оң деген болжамның қаншасы дұрыс	Жалған дабыл қымбат болғанда
Recall	Нақты оң мысалдың қаншасын таптық	Қауіпті өткізіп алмау керек болғанда
F1	Precision мен recall теңгерімі	Екі қате түрі де маңызды болғанда

Шағын практикалық тапсырма

Тапсырма: «Курс өте пайдалы, бірақ тапсырмалар тым көп» сөйлемін талдаңыз. 1) токендерге бөліңіз; 2) жалпы сезімді және аспектілер бойынша сезімді анықтаңыз; 3) қандай сөздер шешімге әсер еткенін түсіндіріңіз; 4) модель қате жіберуі мүмкін екі себепті атаңыз.

Күтілетін талдау: жалпы пікір аралас. «Пайдалы» сөзі курс мазмұнына оң баға береді, «тым көп» тіркесі тапсырма көлеміне теріс баға береді. Қарапайым bag of words аралас мағынаны бір класқа қысқартуы мүмкін, ал аспектілік модель екі нысанды бөлек талдайды.

Бақылау сұрақтары

- NLP жүйесі мәтінді неге сандық көрініске айналдырады?
- Токенизацияның сөздік және сөз бөлігі тәсілдері қалай ерекшеленеді?
- TF-IDF жиі сөздің салмағын қалай өзгертеді?
- Эмбеддинг bag of words тәсілінен несімен ерекшеленеді?
- Attention механизмі қандай мәселені шешеді?
- Transformer мен RNN арасындағы негізгі айырмашылық қандай?
- Тең емес кластарда accuracy неге жеткіліксіз?
- Қазақ тіліндегі NLP үшін қандай ерекшеліктер маңызды?
- Генеративті модельдің жауабын неге тексеру керек?
- NLP жүйесінде адам бақылауы қай кезде міндетті?

Негізгі терминдер

Термин	Қысқаша анықтама
NLP	Адам тіліндегі мәтін мен сөйлеуді компьютерлік өңдеу саласы
Токен	Мәтінді өңдеуге арналған бірлік
Лемма	Сөздің негізгі сөздік формасы
TF-IDF	Құжаттағы сөздің салыстырмалы маңызын өлшейтін салмақ
Эмбеддинг	Тілдік бірліктің тығыз сандық векторы
NER	Мәтіндегі атаулы мәндерді табу
Attention	Маңызды контекст бөліктеріне салмақ беру механизмі
Transformer	Self-attention негізіндегі нейрондық архитектура
Fine tuning	Алдын ала оқытылған модельді арнайы міндетке бейімдеу
Галлюцинация	Модельдің сенімді көрінетін, бірақ негізсіз жауап құруы

Ұсынылатын әдебиеттер

- Jurafsky D. және Martin J. Speech and Language Processing.
- Manning C. және Schütze H. Foundations of Statistical Natural Language Processing.
- Vaswani A. және басқалар. Attention Is All You Need.
- Devlin J. және басқалар. BERT Pre-training of Deep Bidirectional Transformers for Language Understanding.
- Bird S., Klein E., Loper E. Natural Language Processing with Python.