УНИВЕРЗИТЕТ У НИШУ
ЕЛЕКТРОНСКИ
ФАКУЛТЕТ

Примена графовских неуронских мрежа у реализацији система за препоруке

МАСТЕР РАД

Студијски програм: Вештачка интелигенција
и машинско учење

Кандидат:

Никола Петровић, бр. инд. 1466

Ментор:

Проф. др Александар Милосављевић

Ниш, децембар 2023. год.

Мастер рад
Примена графовских неуронских мрежа у реализацији система за препоруке

Задатак: *Упознати се са основним принципима система за препоруке. Приказати таксономију и описати начин рада ових система. Проучити структуру и елементе потребне за креирање графовских неуронских мрежа као и могућности примене ових мрежа на задатак система за препоруке. У практичном делу имплементирати прототип апликације која обезбеђује креирање ефикасног модела, уз помоћ графовских неуронских мрежа, за предикцију персонализованих препорука.*

Студент:	Комисија за одбрану:
	_____
Никола Петровић, бр. инд. 1466	1. Проф. др Име Презиме
	_____
Датум пријаве рада: 16.10.2023.	2. Доц. др Име1 Презиме1
Датум предаје рада:	_____
Датум одбране рада:	3. Проф. др Име2 Презиме2

Примена графовских неуронских мрежа у реализацији система за препоруке

САЖЕТАК

У савременом добу, коришћење онлајн апликација представља свакодневницу, Интернетом свакодневно кружи огромна количина информација, велики играчи на разне начине покушавају да дођу до што веће зараде коришћењем доступних информација, један од приступа реализацији тог циља јесу системи за препоруке. Системи за препоруке представљају софтверске алате и технике који корисницима пружају персонализоване сугестије о производима, садржајима, услугама или особама на основу ранијег понашања, преференција и сличности са другим корисницима. У овом раду ће теоретски бити описан начин рада оваквих система, почевши од најједноставнијих основних приступа, до савремених, *state-of-the-art* решења, другим речима, биће објашњен еволутивни пут развоја система за препоруке. Колаборативно филтрирање представља основу система за препоруке и карактеришу га једноставност имплементације, минимална комплексност података, али и многи недостаци, које градацијски решавају приступи као што су препоруке засноване на садржају, преко препорука на основу знања, све до хибридних система којима припада и систем од интереса у овом раду, а то је систем за препоруке заснован на графовским неуронским мрежама. Графовске неуронске мреже представљају револуцију у системима за препоруке и приступ на коме се заснивају сва тренутна истраживања у овој области. У основи је графовска структура која аутоматски решава проблем реткопоседнутих матрица, док предвиђање веза између чворова хетерогеног бипартитног графа идеално осликава проблем система за препоруке, који има за циљ да пронађе нове гране у графу на основу карактеристика чворова. У раду је описан систем прослеђивања порука између чворова у графу који се налази у основи поменутих мрежа, популарни просторно засновани и филтери који користе механизам пажње, пулинг слојеви, али и проблему специфични начини узорковања потенцијалних веза у графу.

Основна идеја практичног дела рада јесте имплементација система за препоруке заснованог на графовским неуронским мрежама од нуле, што подразумева проналажење адекватног скупа података, предобраду и препроцесирање Амазоновог скупа података за превођење у графовски облик, креирање хетерогеног бипартитног графа, поделу графа по конекцијама (линковима) на тренинг, тест и валидациони скуп, креирање *SAGE* и *GAT* конволуционих слојева, затим тренинг, евалуацију, валидацију и на крају приказ резултата примене поменутих модела на задатак предикције линкова између чворова графа и примену најбољег добијеног модела за креирање персонализованих препорука. Коришћењем технологија као што су Python, Jupiter Notebook, PyTorch Geometric (PyG), PyTorch… Акценат је на испитивању ефикасности овог приступа и проналажењу најбоље комбинације разних параметара у циљу креирања што прецизнијег модела за персонализоване препоруке заснованог на графовским неуронским мрежама.

Кључне речи: системи за препоруке, графовске неуронске мреже, *SAGE* модел, *GAT* модел.

The application of graph neural networks in the implementation of recommendation systems.

ABSTRACT

In the modern age, the use of online applications is a daily thing, a huge amount of informations circulates on the Internet every day, big players try in various ways to get as much profit as possible by using the available informations, one of the approaches to the realization of that goal is recommendation systems. Recommender systems are software tools and techniques that provide users with personalized suggestions about products, content, services, or people based on past behavior, preferences, and similarities to other users. This paper will theoretically describe how such systems work, starting from the simplest basic approaches to modern, *state-of-the-art* solutions, in other words, the evolutionary path of the recommendation system will be explained. Collaborative filtering is the basis of a recommendation system and is characterized by simplicity of implementation, minimal data complexity, but also many shortcomings, which are gradually solved by approaches such as recommendations based on content, through recommendations based on knowledge, up to hybrid systems to which the system of interest also belongs in this paper, which is a recommendation system based on a graph neural networks. Graph Neural Networks represent a revolution in recommender systems and the approach on which all current researches in this field are based. It is basically a graph structure that automatically solves the problem of sparse matrices, while the prediction of links between the nodes of a heterogeneous bipartite graph ideally represents the problem of a recommender system, which aims to find new branches in the graph based on the characteristics of the nodes. The paper describes the message forwarding system between nodes in the graph that is at the base of the mentioned networks, popular spatially based and filters that use the attention mechanism, pooling layers, but also problem-specific ways of sampling potential connections (links) in the graph.

The basic idea of the practical part of the work is the implementation of a recommendation system based on graph neural networks from scratch, which involves finding an adequate data set, pre-processing and pre-processing the Amazon data set for translation into graph form, creating a heterogeneous bipartite graph, dividing the graph by the connections (links) on training, test and validation set, creation of SAGE and GAT convolutional layers, then training, evaluation, validation and finally presentation of the results of applying the mentioned models to the task of predicting links between graph nodes and applying the best obtained model to create personalized recommendations. Using technologies such as Python, Jupiter Notebook, PyTorch Geometric (PyG), PyTorch... The emphasis is on testing the effectiveness of this approach and finding the best combination of various parameters in order to create the most accurate model for personalized recommendations based on graph neural networks.

Keywords: recommender systems, graph neural networks, SAGE model, GAT model

Садржај

УВОД

Свакодневно се сусрећемо са различитим изборима и питањима, шта ћемо доручковати, ручати, коју кафу ћемо попити, какав ће бити план тренинга, којим аутобусом ћемо отићи на посао, у које време ћемо изаћи у град, ког дана ћемо отићи на пецање или који ћемо спорт пратити за викенд. Понекад се чини као да се свакодневни живот састоји од низа одлука које морамо доносити у непрекидном низу. Много избора, превише опција, колико времена би уштедели када би имали неки систем који ће нам организовати дан, месец и годину на најоптималнији, најпаметнији и најјефтинији начин? Да ли би у том случају били задовољни одабиром без нашег питања, колико ће нам одговарати план и да ли ћемо имати воље да у датом тренутку испунимо наметнут задатак?

Одговор на ова питања није једноставан. Иако бисмо уштедели време и ослободили се сталног доношења одлука, живот би можда изгубио дубљи смисао уколико би све битне одлуке биле вођене неким спољним, вештачким планом. Можда не бисмо увек били сагласни са таквим изборима, можда бисмо уместо еспреса желели домаћу кафу, или уместо пецања средом желели роштиљ са пријатељима. Чини се да би такав систем у реалном животу могао бити контрапродуктиван.

Дакле, шта би нам олакшало живот и уштедело време, а да не утиче превише на наше личне изборе и тренутне жеље? Већ смо закључили да потпуна аутоматизација одлука није идеална. Потребан нам је неки систем који би нам помогао да сузимо избор. У данашњем дигиталном добу проводимо много времена на интернету претражујући сајтове, тражећи пословне прилике, филмове, музику, вести и слично. Све то у мору доступних информација и прилика, које можда нису намењене конкретно нама или нам не одговарају. Интернет је, дакле, идеално поље где би систем за персонализоване препоруке могао имати значајну улогу и могао бити од велике користи.

Међутим, велике глобалне компаније су одавно препознале потенцијал за профитабилност ових система. Ови системи, који већ постоје, значајно штеде наше време, олакшавајући нам одлуке у многим аспектима онлајн живота. Они нас повезују с пријатељима, препоручују послове, производе које бисмо могли купити, или филмове које бисмо желели да гледамо, често без наше свести о томе. Називамо их системима за препоруке (енг. Recommender systems). Распрострањени су на интернету и интегрални су део многих друштвених мрежа, платформи за стриминг филмова, е-трговине и сервиса за слушање музике. Ови системи користе различите алгоритамске методе за анализирање корисничког понашања и предлагање садржаја који би им могао највише одговарати. Употребљавајући обиље података који се непрестано прикупљају и обрађују, ови системи циљају на успостављање конекција између корисника и садржа ја или других корисника са сличним интересовањима.

Системима за препоруке данас се придаје кључни значај у дигиталној економији. Са растом *Big Data* ере, количине података које компаније обрађују расту експоненцијално. Према проценама, 35% прихода компаније Амазон и 80% онога што корисници гледају на Нетфликсу потиче из система за препоруке. Ова импресивна статистика не само да указује на значај ових система у савременом пословању, већ и мотивише компаније да непрестано иновирају и оптимизују своје алгоритме за препоруку. У трци за супериорност, прешло се од основних модела до напредних, *state-of-the-art* решења која имплементирају графовске неуронске мреже (енг. Graph Neural

Networks), постављајући тако нови стандард у индустрији. Потенцијал ових система није упитан и сваки напредак у овом смеру може потенцијално довести до огромних профита на страни великих корпорација, док на страни корисника може не само олакшати претрагу већ и прецизније предложити садржај који одговара њиховим преференцијама и потребама.

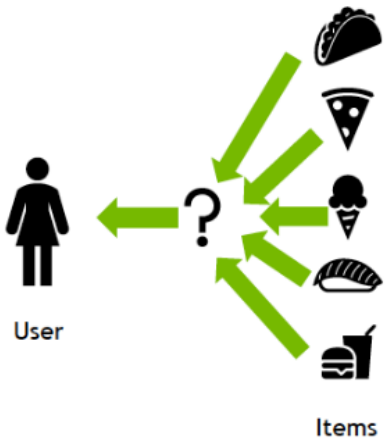
Док су иницијални системи за препоруке били прилично генерализовани, данашње методе се ослањају на дубље анализе понашања корисника, узимајући у обзир њихове претходне интеракције, претраге, куповине и друге релевантне информације. Ово омогућава креирање скоро персонализованих искустава за сваког корисника. У будућности, очекује се да ће се системи за препоруке развијати даље, интегришући још напредније технологије попут проширене и виртуалне стварности, као и узимајући у обзир емоције и расположење корисника. Такође, како се све више пажње поклања питањима приватности, компаније ће морати да пронађу начин да балансирају између пружања персонализованих искустава и очувања поверења својих корисника.

СИСТЕМИ ЗА ПРЕПОРУКЕ

Системи за препоруке (енг. Recommender systems) представљају софтверске алате и технике који корисницима пружају персонализоване сугестије о производима, садржајима, услугама или особама на основу ранијег понашања, преференција и сличности са другим корисницима. Циљ ових система јесте да пруже релевантне и корисне препоруке корисницима, помажући им да пронађу одговарајући садржај у обимном скупу информација [1].

Систем за препоруке представља алгоритам вештачке интелигенције (енг. Artificial intelligence), обично повезан са машинским учењем (енг. Machine Learning), који користи велике количине података (енг. Big Data) да предложи или препоручи додатне производе потрошачима. Ове препоруке могу бити базиране на различитим критеријумима, укључујући раније куповине, историју претраге, демографске податке и друге факторе. Системи за препоруке су веома корисни јер помажу корисницима да открију производе и услуге које можда сами не би пронашли. Обучени су да разумеју преференције, претходне одлуке и карактеристике људи и производа користећи податке прикупљене о њиховим интеракцијама. То укључује утиске, кликове, лајкове и куповине. Због своје способности да предвиде интересе и жеље потрошача на високо персонализованом нивоу, системи за препоруке су омиљени код пружалаца садржаја и производа. Они могу довести потрошаче до било ког производа или услуге која их интересује, од књига преко филмова и здравствених препорука до одеће или потенцијалних нових пријатељстава. Међутим, постоје и неперсонализоване препоруке које су много једноставније за генерисање. Типични примери укључују топ 10 избора уредника за књиге или филмове. Иако могу бити корисне и ефикасне у одређеним ситуацијама, на пример, када нема довољно информација о преференцијама или интересима циљаног корисника, ове врсте неперсонализованих препорука нису примарни фокус истраживања система за препоруке.

У најосновнијем облику (слика 1), персонализоване препоруке представљају рангиране листе предмета. Први системи за препоручивање користили су алгоритме да би процесуирали препоруке базиране на мишљењима заједнице корисника. Овај метод, назван колаборативно филтрирање, заснива се на принципу да ће препоруке сличних корисника бити релевантне за активног корисника. Растом е-трговине, порасла је потреба за филтрирањем великог броја опција доступних корисницима. Превелики избор понекад може преплавити кориснике и уместо користи донети осећај оптерећења. Схватило се да више опција не значи нужно и бољи избор. Системи за препоручивање постали су кључни у решавању проблема преплављености информацијама, усмеравајући кориснике ка новим или релевантним предметима за њихове тренутне потребе. На захтев корисника, системи за препоручивање генеришу препоруке користећи различите податке о корисницима, доступним предметима и претходним трансакцијама. Корисник може прегледати препоруке, одлучити да ли ће их прихватити и дати повратне информације, које се потом користе за генерисање нових препорука у будућим интеракцијама.



Слика 1. Пример једноставног система за препоруке

Основни разлози због којих компаније користе системе за препоруку:

- **Повећање продаје:** Главни циљ комерцијалних система за препоруке је повећање продаје артикала који се иначе не би продавали без препорука. Циљ је да се кориснику препоруче артикли који одговарају његовим потребама.
- **Продаја разноврснијих артикала:** Омогућавају корисницима да пронађу и одаберу артикле који можда нису лако доступни или популарни.
- **Повећање корисничког задовољства:** Добар систем побољшава корисничко искуство, нудећи релевантне препоруке кроз кориснички интерфејс.
- **Повећање лојалности корисника:** Препознају повратне кориснике и прилагођавају препоруке на основу претходних интеракција, чиме се повећава лојалност корисника.
- **Боље разумевање корисника:** Скупљају информације о преференцијама корисника, које се могу користити за друге циљеве, као што је управљање залихама или циљано обавештавање.

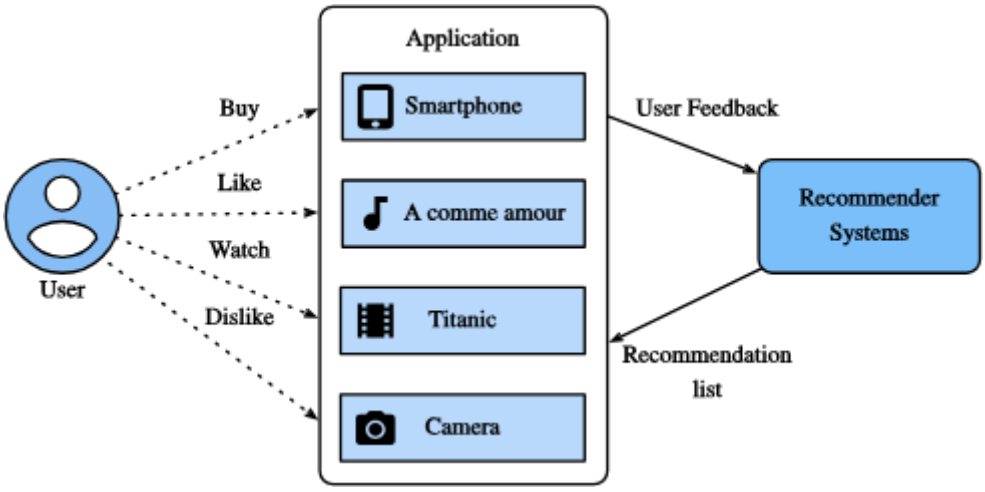
Компаније које користе најквалитетније и најсавременије системе за препоруке котирају се високо у својим индустријама и у највећем броју случајева држе монопол у индустрији. Оне се фокусирају на значајно повећање продаје путем персонализованих понуда и на остваривање позитивног корисничког искуства. Док препоруке убрзавају и поједностављују корисничку претрагу, омогућавајући корисницима лакши приступ садржајима од интереса, оне такође често изненаде кориснике понудама које можда никада самостално не би претражили. Користећи се софистицираном стратегијом комуникације, компаније шаљу е-поруке са линковима ка новим понудама које прецизно одговарају интересима примаоца, као и сугестијама филмова и телевизијских емисија које су у складу са њиховим профилима, на друштвеним мрежама предлажу нова пријатељства и прилике за посао. Овај персонализовани приступ чини да се корисник осећа препознато и дубље схваћено, постајући тако склонији додатним куповинама или конзумацији више садржаја. Кроз овакво разумевање потреба и жеља корисника, компанија стиче одлучујућу конкурентску предност, драстично смањујући ризик од губитка корисника у корист конкуренције. Имплементација оваквих система за препоруке доприноси вредности за кориснике и не само да омогућава компанијама да се поставе корак испред својих конкурената, већ их позиционира за дугорочно вођство у индустрији, гарантујући повећање својих прихода и очување репутације лидера на тржишту.

Слика 2. Бенефити великих компанија од система за препоруке

На слици 2 можемо видети да 80% филмова који се гледају на Нетфликсу буде одабрано помоћу система за препоруке. Ако узмемо у обзир да се на Нетфликсу дневно гледа 164 милиона сати садржаја, то би значило да 131 милион сати дневне гледаности садржаја, или 48 милијарди сати на годишњем нивоу на Нетфликсу, долази уз асистенцију система за препоруку. Компанија заради 31 милијарду долара на годишњем нивоу (2022. година), што говори о томе да је квалитет ових система од круцијалног значаја за пословање компаније [2]. С друге стране, Јутјуб заради око 29 милијарди долара на годишњем нивоу (2022. година). При томе, 60% кликова на препоручене видео записе на почетној страни и 14% одгледаних видео записа корисници одаберу из препоруке поред видеа. Узевши у обзир да је просечно проведено време једног корисника Јутјуба на овој апликацији дневно 48 минута, и да Јутјуб има 467 милиона корисника, добијамо да је на дневном нивоу укупно време проведено на Јутјубу, а иницирано системом за препоруке, око 3 милијарде минута [3]. Амазон, као гигант у својој индустрији е-трговине, је у 2022. години зарадио 225 милијарди долара од продаје, од чега је систем за препоруке допринео са 35%, што чини скоро 79 милијарди долара. Ако се осврнемо на податке, јасно је да се вртоглаве суме новца зарађују помоћу система за препоруке. Може се закључити да ови системи у некој мери утичу на светску економију, јер водеће светске компаније зависе управо од њих. Ово нам говори да системи за препоруке, поред тога што су свеопште присутни у свакодневном животу и олакшавају корисницима да дођу до жељених садржаја или производа, могу малим побољшањем да доносе огромну зараду компанијама. На пример, ако би Амазонов систем за препоруке дао боље резултате за пар процената, Амазон би зарадио пар милијарди долара више на годишњем нивоу. Поређења ради, изградња највише зграде на свету, Бурј Кхалифе, је коштала 1,5 милијарди долара, изградња Београдског метроа кошта 4,4 милијарде долара, а Научно-технолошки парк у Нишу кошта 13 милиона долара. То значи да би у свакој мало већој општини у Србији могли изградити по један Научно-технолошки парк за тај новац. Зарадом од побољшања једног система за препоруке би могла да се финансира изградња највеће зграде на свету или метроа у једној европској престоници.

Системи за препоруке имају широку примену у различитим индустријама, укључујући:

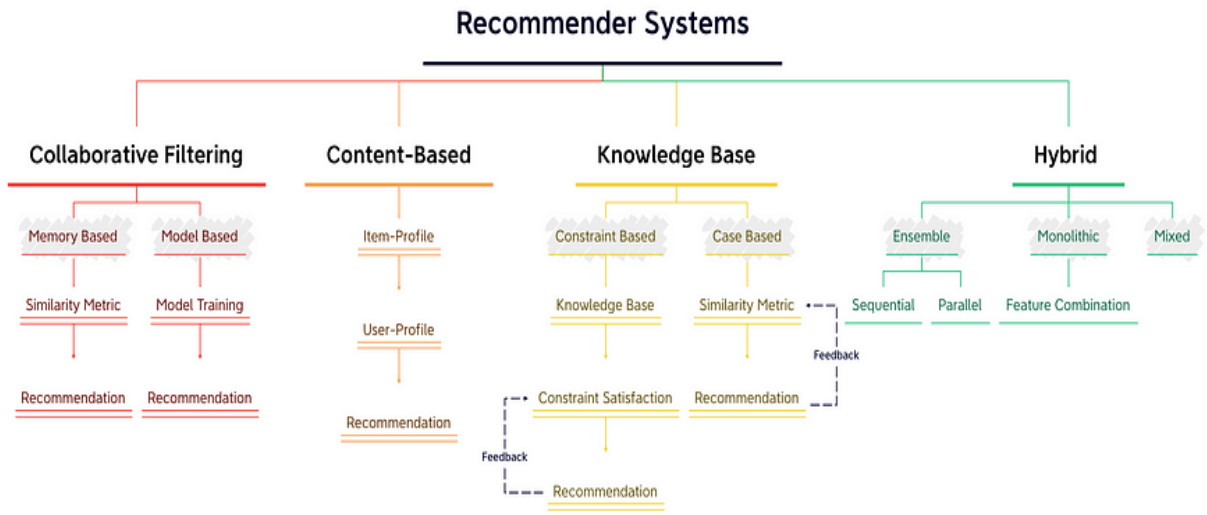
- **E - Trgovina (eng. E – Commerce)**: E - Трговина (eng. E – Commerce): Системи за препоруке су широко коришћени у eTrgovini како би пружали персонализоване препоруке производа купцима на основу њихових претходних активности и преференција (примери: КупујемПродајем, *Amazon, Alibaba, eBay*…).
- **Zabava**: Користе се у забави, као што су музичке и видео стриминг услуге, да препоруче садржај који ће вероватно бити занимљив корисницима (примери: *Netflix, Spotify, YouTube*…).
- **Vesti i mediji**: Користе се на платформама за вести и медије како би препоручивали чланке, видео записе и други садржај који је релевантан за интересовања корисника (примери: *BBC News, CNN, The New York Times*…).
- **Društveni mediji**: Користе се на друштвеним мрежама да препоруче пријатеље, групе или објаве које би могле бити занимљиве корисницима (примери: *Facebook, Instagram, Twitter, LinkedIn*…).
- **Zdravstvo**: Користе се у здравству да би пружили персонализоване препоруке за третмане, лекове и друге здравствене услуге (примери: *MyFitnessPal, eZdravlje, Doktor.rs*…).
- **Finansije**: Користе се у финансијама како би пружили персонализоване препоруке за инвестиције, кредитне производе и друге финансијске услуге (примери: *Robinhood, mBanking, Credit Karma*…).
- **Reklamiranje**: Користе се у рекламирању како би пружили персонализоване препоруке за рекламе и понуде које би могле бити занимљиве корисницима (примери: *Google Ads, Facebook Advertising, AdRoll*…).
- **Obrazovanje**: Користе се у образовању да би пружили персонализоване препоруке за курсеве, програме и други образовни садржај (примери: *Coursera, Khan Academy, Udemy*…).



Слика 3. Примена система за препоруке

Таксономија система за препоруке

Различити приступи у изградњи система за препоруке постоје због тога што има много специфичних проблема које треба решити и сваки од њих има нешто што га издваја од осталих. Сходно томе, настале су различите технике и приступи, како би се за сваки тип проблема пронашло адекватно решење. Постоји централна идеја о "корисности" или "вредности" артикла за корисника. Ово се често приказује кроз функцију ***R(u, i)***, која у суштини узима у обзир оцене корисника за одређене артикле. Кључни задатак система је да предвиди ову вредност за парове корисника и ставки, и кориснику ће бити препоручене ставке с највећом вредношћу за овај параметар. Док неки системи процењују корисност артикла пре препоруке, други примењују хеуристике и претпоставке. Ово је често у системима заснованим на знању. Међутим, ти приступи су мање уобичајени у стварном свету. Понекад корисност зависи од "контекстуалних" варијабли, као што су време и локација корисника. На пример, корисник би могао бити заинтересован за ресторане ближе тренутној локацији. Дакле, предвиђање праве препоруке постаје све комплексније, јер мора узети у обзир ове додатне услове. За различите проблеме, постоје различита решења, на слици 4. можемо видети основну поделу система за препоруке.



Слика 4. Таксономија система за препоруке [4]

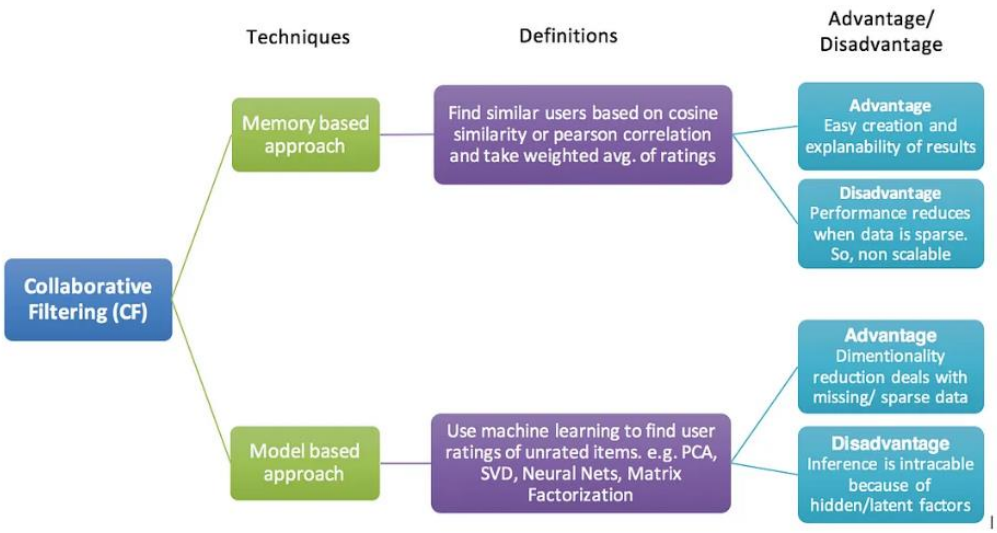
Кратак преглед четири главне категорије је следећи:

- **Колаборативно филтрирање (енг. Collaborative Filtering - CF)** - Систем користи оцене других корисника како би препоручио ставке изабраном кориснику.
- **Препорука на основу садржаја (енг. Content – Based - CB)** - Систем користи сопствене историјске податке корисника да би му препоручио ставке.
- **Препорука на основу знања (енг. Knowledge - Based - KB)** - Систем користи експлицитне захтеве корисника да би му препоручио ставке.
- **Хибридни приступ (енг. Hybrid)** - Комбинује претходне приступе на различите начине да би креирао свеобухватни систем за препоруке.

Колаборативно филтрирање (енг. Collaborative Filtering)

Колаборативно филтрирање такође познато и као социјално филтрирање користи алгоритме за филтрирање података из корисничких рецензија ради креирања персонализованих препорука за кориснике са сличним преференцијама. Колаборативно филтрирање се такође користи за избор садржаја и реклама за појединце на друштвеним мрежама.

У систему препорука, најшире примењен и најпопуларнији метод је колаборативно филтрирање. У систему заснованом на колаборативном филтрирању, препорука се врши на основу претходних оцена ставки особа које су сличне циљној особи. Главна идеја колаборативног филтрирања је прикупљање информација о старом корисничком понашању и мишљењима како би се утврдило да ли су нови и стари корисник слични или не. Бележе се све оцене корисника, историја гледања и куповине, што омогућава систему препорука да изгради везу између особа које имају идентичну природу и међу стварима које интересују сличног корисника. Овај приступ сматра да ће корисници који су раније имали сличне преференције сигурно имати сличан укус и у будућности. Примена технике колаборативног филтрирања је једноставна када се суочавамо са ниском зависношћу од података (енг. low data dependency) и даје тачне предлоге за препоруку. Колаборативно филтрирање може се поделити на три главне категорије: модел заснован на меморији **(енг. Memory - Based)**, заснован на моделу **(енг. Model - Based)** и хибридно колаборативно филтрирање **(енг. Hybrid),** где се модел заснован на меморији даље дели у две подгрупе, а то су подмодел заснован на производу **(енг. Item - Based)** и подмодел заснован на кориснику **(енг. User - Based).**



Слика 5. Типови колаборативног филтрирања [5]

Кључна разлика између приступа заснованог на меморији и техника заснованих на моделу је да не учимо никакав параметар користећи градијентни спуст (или било који други оптимизациони алгоритам). Најближи корисници или ставке се израчунавају само користећи косинусну сличност или коефицијенте Пирсонове корелације, који се заснивају само на аритметичким операцијама.

Приступ заснован на меморији

У техници заснованој на меморији, сличност између људи или ствари се израчунава ради препоруке ставки. Корисници са сличним афинитетима имају већу вредност сличности. Ова техника налази кориснике који се подударају са афинитетима циљног корисника и потом предвиђа афинитете циљног корисника за нове ставке. То је ефикасан и лак метод за примену. Корисници са сличним укусом се групишу заједно.

Модел заснован на кориснику или како се другачије назива *user-based nearest neighbor recommendation* је један од најраније коришћених метода. Основна идеја је врло једноставна и гласи овако: на основу базе података са оценама и ИД-а тренутног (активног) корисника као улаза, идентификујте друге кориснике (понекад називане корисницима-вршњацима или најближим суседима) који су у прошлости имали сличне преференције као активни корисник. Затим, за сваки производ p који активни корисник још није видео, предвиђање се израчунава на основу оцена за p датих од стране корисника-вршњака. Основне претпоставке таквих метода су да (a) ако су корисници имали сличне укусе у прошлости, имаће их и у будућности и (b) преференције корисника остају стабилне и доследне током времена.

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Табела 1. База података за колаборативно филтрирање

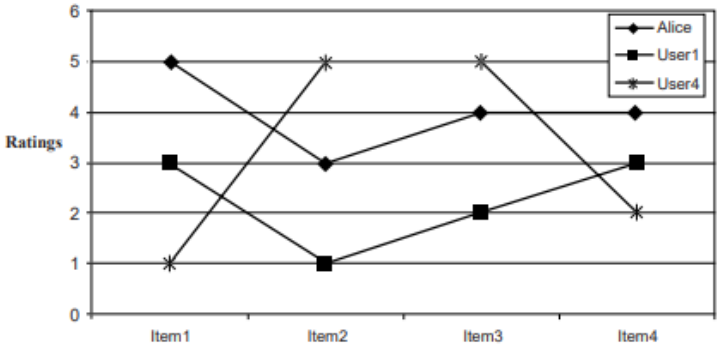
Пример 1. [6] Размотримо први пример. Табела 1 приказује базу података о оценама тренутног корисника, *Алисе*, и неким другим корисницима. *Алиса* је, на пример, оценила "*Ставка1*" са "5" на скали од 1 до 5, што значи да јој се та ставка врло допала. Задатак система за препоруку у овом једноставном примеру је да одреди да ли ће се *Алиси* свидети или не свидети "*Ставка5*", коју *Алиса* још није оценила или видела. Ако можемо предвидети да ће се *Алиси* ова ставка веома допадати, требали бисмо је укључити у листу препорука за *Алису*. У ту сврху, тражимо кориснике чији је укус сличан *Алисином* и затим узимамо оцене ове групе за "*Ставка5*" да бисмо предвидели да ли ће се *Алиси* допасти ова ставка.

Пре него што детаљније разговарамо о математичким израчунавањима потребним за ова предвиђања, уведимо следеће конвенције и симболе:

Користимо $U=\{u_1,...,u_n\}$ да обележимо скуп корисника, $P=\{p_1,...,p_m\}$ за скуп производа (ставки) и R као $n \times m$ матрицу оцена $r_{i,j}$, са $i \in 1...n$, $j \in 1...m$. Могуће вредности оцена дефинисане су на нумеричкој скали од 1 (јако се не допада/свиђа) до 5 (јако се допада/свиђа). Ако одређени корисник i није оценио ставку j, одговарајући унос матрице $r_{i,j}$ остаје празан.

У вези са одређивањем скупа сличних корисника, једна честа мера која се користи у системима за препоруку је Пирсонов коефицијент корелације. Сличност $sim(a,b)$ корисника a и b, на основу матрице оцена R, дефинисана је у формули 1 симбол $\overline{ra}$ одговара просечној оцени корисника a.

$$sim(a,b) = \frac{\sum_{p\in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\sum_{p\in P}(r_{a,p} - \overline{r_a})^2}\sqrt{\sum_{p\in P}(r_{b,p} - \overline{r_b})^2}} \quad (1)$$



Слика 6. Поређење Алисе са два корисника

Сличност *Алисе* и *Корисника1* је следећа ($\overline{rAlice} = \overline{ra} = 4$, $\overline{rUser1} = \overline{rb} = 2.4$):

$$\frac{(5-\overline{r_a})*(3-\overline{r_b}) + (3-\overline{r_a})*(1-\overline{r_b}) + \cdots + (4-\overline{r_a})*(3-\overline{r_b}))}{\sqrt{(5-\overline{r_a})^2 + (3-\overline{r_a})^2 + \cdots}\sqrt{(3-\overline{r_b})^2 + (1-\overline{r_b})^2 + \cdots}} = 0.85 \quad (2)$$

Пирсонов коефицијент корелације узима вредности од +1 (јака позитивна корелација) до −1 (јака негативна корелација). Сличности са осталим корисницима, *Корисник2* до *Корисник4*, су 0,70, 0,00 и −0,79, редом.

На основу ових израчунавања, примећујемо да су *Корисник1* и *Корисник2* били на неки начин слични *Алиси* у свом ранијем начину оцењивања. Такође видимо да Пирсонова мера узима у обзир чињеницу да су корисници различити у томе како тумаче скалу оцена. Неки корисници имају тенденцију да дају само високе оцене, док други никада неће дати 5 неком предмету. Пирсонов коефицијент у израчунавању избацује ове просеке да би кориснике учинио упоредивим, тј. иако су апсолутне вредности оцена *Алисе* и *Корисника1* врло различите, детектује се јасна линеарна корелација оцена и сличност корисника. Ова чињеница се такође може видети на визуелном приказу на слици 6, која илуструје и сличност између *Алисе* и *Корисника1* и разлике у оценама *Алисе* и *Корисника4*. Да бисмо направили предвиђање за *Предмет5*, сада морамо да одлучимо чије ћемо оцене суседа узети у обзир и колико ћемо ценити њихова мишљења. У овом примеру, очигледан избор би био да узмемо *Корисника1* и *Корисника2* као сличне кориснике да предвидимо *Алисину* оцену. Могућа формула за израчунавање предвиђања оцене корисника a за предмет *p* која такође узима у обзир релативну близину најближих суседа N и просечну оцену $\overline{ra}$ је следећа:

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b\in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b\in N} sim(a,b)} \quad (3)$$

У примеру, предвиђање *Алисине* оцене за *Предмет5* на основу оцена блиских суседа *Корисник1* и *Корисник2* биће:

$$4 + 1/(0.85 + 0.7) * (0.85 * (3 - 2.4) + 0.70 * (5 - 3.8)) = 4.87 \quad (4)$$

На основу ових шема израчунавања, сада можемо израчунати предвиђања оцена за *Алису* за све предмете које још није видела и укључити оне с највећим предвиђеним вредностима у листу препорука. У примеру, највероватније ће бити добар избор укључити *Предмет5* у такву листу. Пример базе оцена приказан изнад је, наравно, идеализација стварног света. У стварним апликацијама, базе оцена су много веће и могу обухватати хиљаде или чак милионе корисника и предмета, што значи да морамо

размишљати о рачунској сложености. Осим тога, матрица оцена је обично врло ретко-поседнута, што значи да ће сваки корисник оценити само врло мали подскуп доступних предмета. Поставља се питање шта можемо препоручити новим корисницима или како да се носимо са новим предметима за које не постоје оцене.

Поред Пирсоновог коефицијента корелације користе се још и *adjusted cosine similarity, Spearman's rank correlation coefficient, mean squared difference…* Међутим, с годинама се издвојио *adjusted cosine similarity* илити косинусна сличност. Ова метрика може се геометријски посматрати ако се ред (колона) матрице оцена за датог корисника (ставку) третира као вектор. За колаборативно филтрирање засновано на корисницима, сличност између два корисника мери се као косинус угла између два вектора корисника. За кориснике *u* и *u′*, косинусна сличност је:

$$sim(u,u') = cos(\theta) = \frac{\mathbf{r}_u \cdot \mathbf{r}_{u'}}{\|\mathbf{r}_u\| \|\mathbf{r}_{u'}\|} = \sum_i \frac{r_{ui} r_{u'i}}{\sqrt{\sum_i r_{ui}^2} \sqrt{\sum_i r_{u'i}^2}} \quad (5)$$

Можемо предвидети оцену корисника *u* за филм *i* тако што ћемо узети тежинску суму оцена филма *i* од свих осталих корисника (*u′s*), где је тежина број сличности између сваког корисника и корисника *u*.

$$\hat{r}_{ui} = \sum_{u'} sim(u,u') r_{u'i} \quad (6)$$

Требало би такође нормализовати оцене у односу на укупан број оцена од стране u′ (осталих корисника).

$$\hat{r}_{ui} = \frac{\sum_{u'} sim(u,u') r_{u'i}}{\sum_{u'} |sim(u,u')|} \quad (7)$$

Модел заснован на производима/ставкама/артиклима

(енг. *item-based recommendation).* Иако су приступи засновани на корисницима успешно примењени у различитим областима, остају неки озбиљни изазови када је реч о великим е-трговинским сајтовима на којима морамо управљати милионима корисника и милионима ставки из каталога. Посебно, потреба за прегледом огромног броја потенцијалних суседа чини немогућим израчунавање предвиђања у реалном времену. Велики е-трговачки сајтови често примењују другу технику, препоруке засноване на ставкама, која је погоднија за препроцесирање ван мреже и тако омогућава израчунавање препорука у реалном времену чак и за врло велику матрицу оцена.

Основна идеја алгоритама заснованих на ставкама је израчунавање предвиђања коришћењем сличности између ставки, а не сличности између корисника. Ако поново прегледамо нашу базу оцена и направимо предвиђање за *Алису* за *Ставку5*. Прво упоређујемо векторе оцена осталих ставки и тражимо ставке које имају оцене сличне *Ставци5*. У примеру видимо да су оцене за *Ставку5* (3, 5, 4, 1) сличне оценама *Ставке1* (3, 4, 3, 1) и такође постоји делимична сличност са *Ставком4* (3, 3, 5, 2). Идеја препоруке засноване на ставци је сада да једноставно погледамо Алисине оцене за ове сличне ставке. Алиса је дала „5“ за *Ставку1* и „4“ за *Ставку4*. Алгоритам заснован на ставкама израчунава тежински просек ових других оцена и предвиђаће оцену за *Ставку5* негде између 4 и 5.

Пример 2. Да бисмо пронашли сличне ставке, мора се дефинисати мера сличности. У приступима препоруци заснованим на ставкама, косинусна сличност је утврђена као стандардна мера, јер је показано да производи најтачније резултате. Ова мера сличности између два n-димензионална вектора базира се на углу између њих. Ова мера се такође често користи у областима претраге информација и анализе текста за упоређивање два текстуална документа, у којима су документи представљени као вектори појмова.

Сличност између две ставке a и b посматрано као одговарајући вектори оцена $\vec{a}$ и $\vec{b}$ формално је дефинисана на следећи начин:

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|} \quad (8)$$

Симбол · представља скаларни производ вектора. $|\vec{a}|$ је еуклидска дужина вектора, која је дефинисана као квадратни корен из скаларног производа вектора са самим собом.

Косинусна сличност између *Ставке5* и *Ставке1* стога се рачуна на следећи начин:

$$sim(I5, I1) = \frac{3*3+5*4+4*3+1*1}{\sqrt{3^2+5^2+4^2+1^2} * \sqrt{3^2+4^2+3^2+1^2}} = 0.99 \quad (9)$$

Могуће вредности сличности су између 0 и 1, где вредности близу 1 указују на јаку сличност. Основна косинусна мера не узима у обзир разлике у просечном оцењивању корисника. Овај проблем се решава коришћењем прилагођене косинусне мере, која одузима просечну оцену корисника од оцена. Вредности за прилагођену косинусну меру одговарајуће се крећу између −1 и +1, као у Пирсоновој мери.

Нека U буде скуп корисника који су оценили обе ставке a и b. Прилагођена косинусна мера се тада рачуна на следећи начин:

$$sim(a, b) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2} \sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}} \quad (10)$$

Стога можемо трансформисати оригиналну базу података о оценама и заменити оригиналне вредности оцена њиховим одступањем од просечних оцена, како је приказано у Табели 2.

	Item1	Item2	Item3	Item4	Item5
Alice	1.00	−1.00	0.00	0.00	?
User1	0.60	−1.40	−0.40	0.60	0.60
User2	0.20	−0.80	0.20	−0.80	1.20
User3	−0.20	−0.20	−2.20	2.80	0.80
User4	−1.80	2.20	2.20	−0.80	−1.80

Табела 2. Прилагођена косинусна мера

Прилагођена косинусна сличност за *Ставку5* и *Ставку1* за дати пример је следећа:

$$\frac{0.6 * 0.6 + 0.2 * 1.2 + (-0.2) * 0.80 + (-1.8) * (-1.8)}{\sqrt{(0.6^2 + 0.2^2 + (-0.2)^2 + (-1.8)^2} * \sqrt{0.6^2 + 1.2^2 + 0.8^2 + (-1.8)^2}} = 0.80 \quad (11)$$

Након што су одређене сличности између ставки, можемо предвидети оцену за *Алису* за *Ставку5* рачунајући тежинску суму *Алисиних* оцена за ставке које су сличне *Ставци5*. Формално, можемо предвидети оцену за корисника у за производ п на следећи начин:

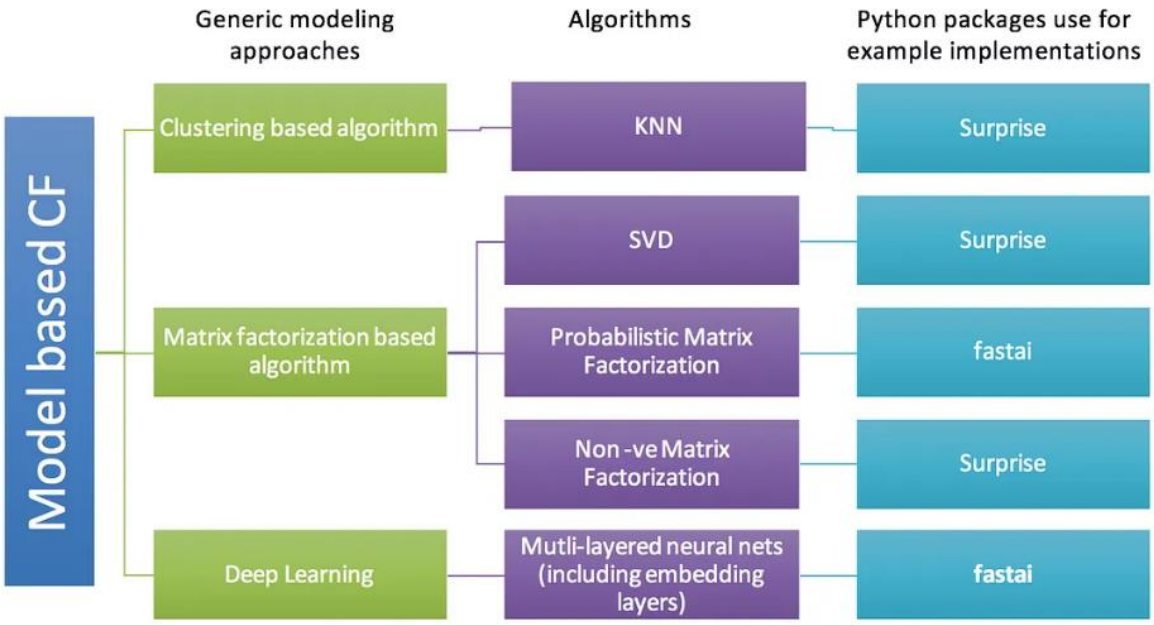
$$pred(u, p) = \frac{\sum_{i \in ratedItems(u)} sim(i, p) * r_{u,i}}{\sum_{i \in ratedItems(a)} sim(i, p)} \quad (12)$$

Као у приступу заснованом на кориснику, величина разматраног окружења је обично такође ограничена на одређену величину, тј. не узимају се у обзир сви суседи за предвиђање.

Техника колаборативног филтрирања од ставке до ставке користила се на *Amazon.com* за препоруку књига или CD-ова купцима. Продавница је 2003. године имала 29 милиона корисника и милионе ставки из каталога. Главни проблем са традиционалним корисничким КФ је да алгоритам не ради добро за тако велики број корисника и ставки из каталога. У стварности, већина корисника је оценила или купила само мали број ставки. Међутим, када број корисника досеже неколико милиона, рачунска предвиђања у реалном времену су неизводљива. За примену алгоритама препоруке заснованих на ставкама на великим електронским трговинама без жртвовања тачности, типично се бира приступ заснован на препроцесирању података. Идеја је да се унапред изгради матрица сличности ставки која описује међусобну сличност свих ставки из каталога. У реалном времену, предвиђање за производ и корисника се прави одређивањем најсличнијих ставки и израдом тежинске суме оцена корисника за ове ставке у окружењу. Што се тиче меморијских захтева, матрица сличности може имати до N^2 уноса. У пракси, међутим, број уноса је знатно мањи. У принципу, препроцесирање окружења је могуће и за корисничке приступе, али сличности међу корисницима су много стабилније у односу на сличности ставки.

Приступ заснован на моделу

Традиционална корисничка техника сматра се техником заснованом на меморији јер се оригинална база рејтинга/оцена чува у меморији и директно се користи за генерисање препорука. У техникама заснованим на моделу, сирови (енг. Raw data) подаци се прво обрађују офлајн, као што је описано за филтрирање засновано на артиклима или неке технике редукције димензионалности. У реалном времену потребан је само предрачунати или "научени" модел да би се дала предвиђања. Иако су технике засноване на меморији теоретски прецизније јер су комплетни подаци доступни за генерисање препорука, такви системи се суочавају са проблемима скалабилности када узмемо у обзир базе са десетинама милиона корисника и милионима артикала.



Слика 7. Типови колаборативног филтрирања заснованог на моделу

Факторизација матрице (енг. **Matrix Factorization)** Идеја иза оваквих модела је да се ставови или преференције корисника могу одредити малим бројем скривених фактора. Ове факторе можемо назвати уграђивањима (енг. **Embeddings**).
Такмичење за Netflix Prize [7], завршено 2009. године, показало је да напредне методе матричне факторизације, које су користили многи учесници такмичења, могу значајно побољшати предиктивну тачност система за препоруку. Углавном, методе матричне факторизације у системима за препоруке користе се за проналажење скривених фактора из образаца оцењивања и карактерисање и корисника и ставки помоћу таквих вектора фактора. У домену филмова, аутоматски идентификовани фактори могу одговарати очигледним аспектима филма као што су жанр или тип, али могу бити и неинтерпретабилни. Препорука за ставку ствара се када су активни корисник и ставка слични у погледу ових фактора. Ова општа идеја искоришћавања скривених "семантичких" фактора успешно је примењена у контексту претраживања информација. 1990 – те године предложено је коришћење сингуларне декомпозиције вредности (енг. **Singular Value Decomposition**) или **SVD** као методу за откривање скривених фактора у документима. У системима за препоруку, идеја искоришћавања скривених веза у подацима и коришћење техника матричне факторизације попут SVD брзо је пренета у домен система за препоруке.

	User1	User2	User3	User4
Item1	3	4	3	1
Item2	1	3	2	6
Item3	2	4	1	5
Item4	3	3	5	2

Табела 3. База података за пример SVD

Пример 3. Размотримо поново нашу матрицу оцена из Табеле 1, из које избацујемо *Алису* и коју транспонујемо како бисмо боље приказали различите операције (Табела 3).

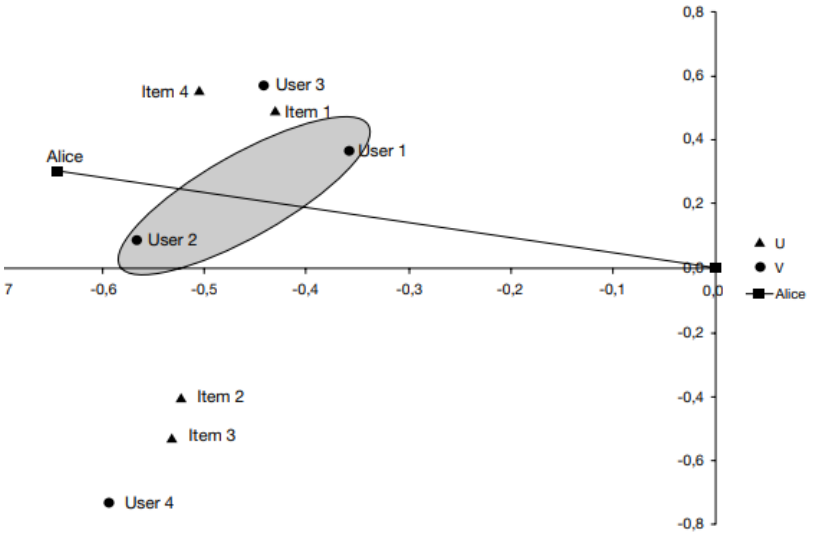
Неформално, теорема SVD тврди да дату матрицу M можемо декомпоновати у производ три матрице на следећи начин, где су U и V називани леви и десни сингуларни вектори, а вредности дијагонале Σ се називају сингуларним вредностима.

$$M = U \Sigma V^T \quad (13)$$

Зато што је 4 × 4-матрица M у Табели 3 квадратна, U, Σ и V су такође квадратне 4 × 4 матрице. Главна поента ове декомпозиције је да можемо приближити потпуну матрицу посматрајући само најважније особине - оне с највећим сингуларним вредностима. У примеру, израчунавамо U, V и Σ (уз помоћ неког софтвера за линеарну алгебру) али задржавамо само две најважније особине узимајући само прве две колоне U и V^T, Табела 4. Пројекција U и V^T у дводимензионалном простору (U_2, $V_2{}^T$) је приказана на слици 8. Матрица V одговара корисницима, а матрица U каталошким ставкама. Иако у нашем конкретном примеру не можемо посматрати никакве кластере корисника, видимо да ставке из U формирају две групе (изнад и испод x-осе). Када се гледају оригиналне оцене, можемо приметити да су *Ставка1* и *Ставка4* добиле сличне оцене. Исто важи и за *Ставка2* и *Ставка3*, који су приказани испод x-осе. У погледу корисника, можемо барем видети да је *Корисник4* мало удаљен од осталих.

U_2		V_2		Σ_2	
−0.4312452	0.4931501	−0.3593326	0.36767659	12.2215	0
−0.5327375	−0.5305257	−0.5675075	0.08799758	0	4.9282
−0.5237456	−0.4052007	−0.4428526	0.56862492		
−0.5058743	0.5578152	−0.5938829	−0.73057242		

Табела 4. Прве две колоне декомпонованих матрица и сингуларне вредности Σ.



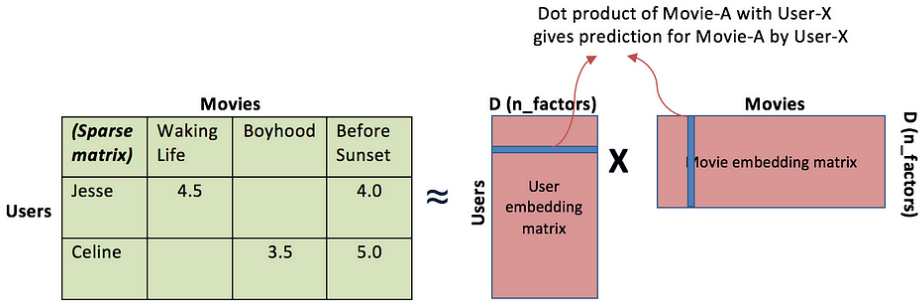
Слика 8. SVD-базирана пројекција у дводимензионалном простору.

Пошто је наш циљ да предвидимо *Алисине* оцене, прво морамо сазнати где би *Алиса* била позиционирана у овом дводимензионалном простору.

Да бисмо сазнали *Алисину* позицију, помножимо *Алисин* вектор оцена [5, 3, 4, 4] са подскупом од две колоне U и инверзом матрице сингуларних вредности са две колоне Σ.

$$Alice_{2D} = Alice \times U_2 \times \Sigma_2^{-1} = [-0.64, 0.30] \quad (14)$$

Узимајући *Алисину* позицију, могу се користити различите стратегије за стварање препоруке за њу. Једна опција би могла бити да се траже суседи у компримованом дводимензионалном простору и да се користе њихове оцене производа као предиктори за Алисину оцену. Ако се поново ослањамо на косинусну сличност за одређивање сличности корисника, *Корисник1* и *Корисник2* ће бити најбољи предиктори за *Алису* у примеру. Опет, различите шеме тежинске вредности, прагови сличности и стратегије за попуњавање недостајућих оцена производа (нпр. на основу просечних вредности производа) могу се користити за фина подешавања предикције.

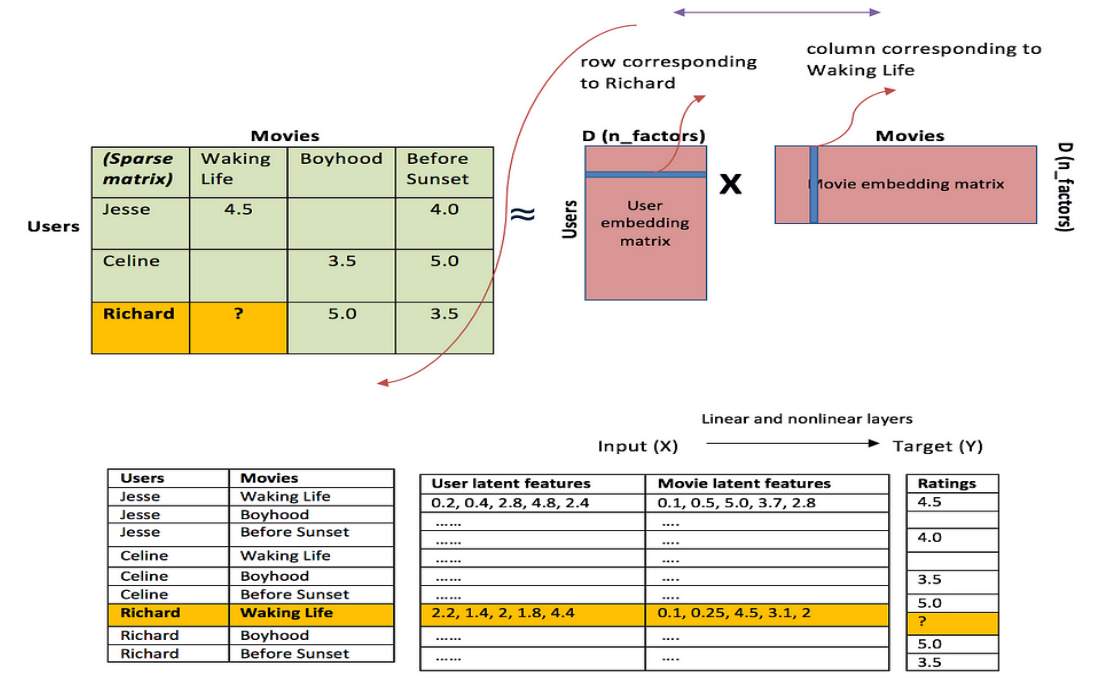


Слика 9. Визуелизација матрице факторизације

Анализа главних компоненти (енг. **Principal component analysis**) или **PCA**. Идеја је препроцесирати базу оцена користећи анализу главних компоненти (PCA) да би се издвојили "најважнији" аспекти података који објашњавају већи део варијансе. PCA је стандардна статистичка анализа заснована на израчунавању декомпозиције сопствене вредности матрице.
Након корака PCA, оригинални подаци о оценама се пројектују уз најрелевантније од главних сопствених вектора. Затим, на основу овог смањеног сета података, корисници се групишу у кластере суседа, и израчунава се просечна оцена за ставке. Сви ови кораци се обављају офлајн. У време извршавања, новим корисницима се тражи да оцене сет “шала” (мерни сет) на нумеричкој скали. Ове оцене се трансформишу на основу главних компоненти, и одређује се одговарајући кластер. Ставке с највишим оценама за овај кластер једноставно се добијају претрагом у препроцесираним подацима. Стога је рачунска сложеност у време извршавања независна од броја корисника, резултујући "**алгоритмом константног времена**". Емпиријска евалуација и упоређивање са основним алгоритмом најближег суседства показује да у неким експериментима PCA може обезбедити упоредиву тачност препоруке, док се време израчунавања може значајно смањити. Потреба за мерним сетом од, на пример, десет оцена је једна од особина која може ограничити практичност приступа у неким доменима.

Непараметарски приступ (**KNN**): Идеја је иста као код препоручивачких система заснованих на меморији. У алгоритмима заснованим на меморији, користимо сличности између корисника и/или предмета и користимо их као тежине да предвидимо оцену за корисника и предмет. Разлика је у томе што се сличности у овом приступу рачунају на основу модела ненадгледаног учења, а не на основу Пирсонове корелације или косинусне сличности. У овом приступу, такође ограничавамо број сличних корисника на k, што систем чини лакше скалабилним.

На слици испод је визуализација која објашњава шта се дешава када користимо **неуронске мреже** (енг. Neural Networks) за овај проблем.



Слика 10. Матрична факторизација и ембединзи за неуронску мрежу.

Предности колаборативног филтрирања

- Имплементација memory - based колаборативног филтрирања олакшава процес препоруке.
- Коришћењем model - based КФ побољшавају се перформансе предвиђања.
- Додавање нових података на инкременталан начин је олакшано у memory - based техници.

Мане колаборативног филтрирања

- Хладан почетак (енг. **Cold start**): Компликација хладног почетка настаје када се нови корисник пријави и не постоје претходни записи о кориснику. У КФ приступу потребна је велика количина информација да би се пружиле тачне препоруке.
- Скалабилност (енг. **Scalability**): Препоруке се дају када у организацији постоје милијарде људи и ствари. Стога, рачунарска снага треба да буде висока да би се предложиле ставке. Систем за препоруке који је изводљив на малом опсегу можда неће бити изводљив на већем. Временска и просторна сложеност алгоритама који се користе морају бити пажљиво узети у обзир.
- Реткост (енг. **Sparsity**): Ставке се продају у огромним количинама на интернету. Матрица оцена корисника и ставки обично неће бити попуњена. Биће недостајућих оцена јер многи корисници нису дали оцене или нису интераговали са многим ставкама. Ово доводи до тога да матрица буде ретка и због тога доводи до мање тачних предвиђања и препорука.
- Синоними (енг. **Sinonymy**): Када иста ставка има различита имена, онда настаје овај проблем. У систему за препоруку тешко је открити латентну асоцијацију, па систем сматра да су ове ставке различите. На пример, *'children's movie'* и *'children's film'* су исте ставке, али систем их сматра различитим. У случају memory - based колаборативног филтрирања, неће израчунавати сличност између ова два термина

јер их сматра различитим. Стога, синонимија прекида перформансе процеса препоруке.

- Сиве овце (енг. **Grey sheep**): Људи који не воле или воле ставке у групи корисника не добијају корист од система за препоруку. Нпр. између два скупа препорука може постојати група профила која припада и једној и другој групи, због недовољно интеракција.
- Црне овце (енг. **Black sheep**): Људи који ни на који начин не припадају ни једној групи профила.
- *Shilling* напад (енг**. Shilling attack**): Означава се као напад „инјекције лажних профила“, јер се праве лажни профили да би се утицало на понашање процеса препоруке. Лажни корисници би давали позитивне коментаре за свој производ и негативне коментаре за своје ривале.
- ***“Curse of Dimensionality”***: При раду са подацима у просторима велике димензионалности, анализа тих података и доношење предикција на основу њих може бити изазовно. Ово не представља проблем када се гледају само корисник, предмет и оцене, али у многим системима за препоруку желимо да укључимо и додатне информације као што су старост, локација, пол итд. ради снажније предикције. Међутим, обрада података високе димензионалности је рачунарски скупља и сложенија.

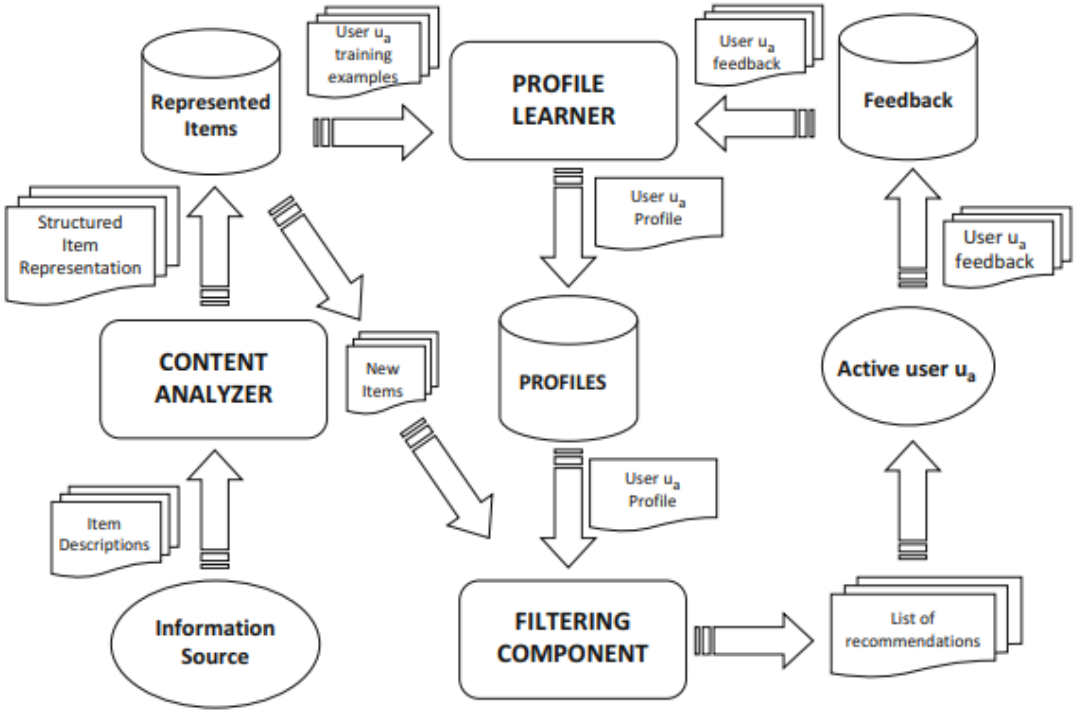
Препоруке на основу садржаја (енг. Content-based recommendations)

Из досадашњег истраживања, уочавамо да за примену техника колаборативног филтрирања, осим корисничких оцена, нам не треба додатно знање о предметима који се препоручују. Главна предност тога је избегавање потребе за обезбеђивањем детаљних и ажурних описа предмета систему. С друге стране, чисто колаборативним приступом не можемо на интуитиван начин изабрати производе на основу њихових карактеристика и конкретних корисничких преференција. На пример, у реалном свету било би логично препоручити нову књигу *Хари Потер Алиси* ако знамо да (а) је жанр ове књиге фантазија и (б) да је *Алиса* увек волела фантастичне романе. Електронски систем препоруке може да изврши овај задатак само ако су доступне две врсте информација: опис карактеристика предмета и кориснички профил који на неки начин описује (прошла) интересовања корисника. Задатак препоруке тада се састоји од одређивања предмета који најбоље одговарају корисничким преференцијама. Овај процес се обично назива препорука на основу садржаја. Иако се такав приступ ослања на додатне информације о предметима и корисничким преференцијама, не захтева постојање велике корисничке заједнице или историју оцењивања – другим речима, листе препорука могу се генерисати чак и ако постоји само један корисник.

У практичним ситуацијама, технички описи карактеристика предмета, као што су жанр књиге или списак глумаца у филму, често су доступни у електронској форми, пошто их често већ обезбеђују произвођачи или пружаоци услуга. Међутим, оно што остаје изазов је прикупљање субјективних, квалитативних карактеристика. У доменима као што су квалитет и укус, разлози зашто неко нешто воли не морају увек бити повезани с конкретним карактеристикама производа и могу се заснивати на субјективном утиску о дизајну предмета. Описе карактеристика предмета називамо "садржајем" јер већина техника које се користе су развијене за препоруку текстуалних докумената, као што су поруке на интернет групама или веб странице. Основна претпоставка је да се карактеристике предмета могу аутоматски извући из самог садржаја документа или из

неструктурираних текстуалних описа. Типични примери су системи који препоручују новинске чланке упоређујући кључне речи чланка с кључним речима других чланака које је корисник раније високо оценио.

Међутим, рани модели система заснованих на садржају базирани су на приступима заснованим на кључним речима који искоришћавају једноставно пребројавање термина. Стога, ови модели нису били способни да потпуно разумеју текстуални садржај који описује предмете нити да кодирају семантичке односе између термина. Овакви модели показују јасна ограничења због својстава елемената природног језика, као што су: Полисемија - више значења за једну реч, Синонимија - више речи са истим значењем, Изрази са више речи, низ речи чија својства нису предвидива, Идентификација ентитета, тежина локализације и класификације елемента у тексту, повезивање ентитета, тежина одређивања идентитета ентитета у тексту.



Слика 11. Основна архитектура система за препоруку заснованих на садржају [8]

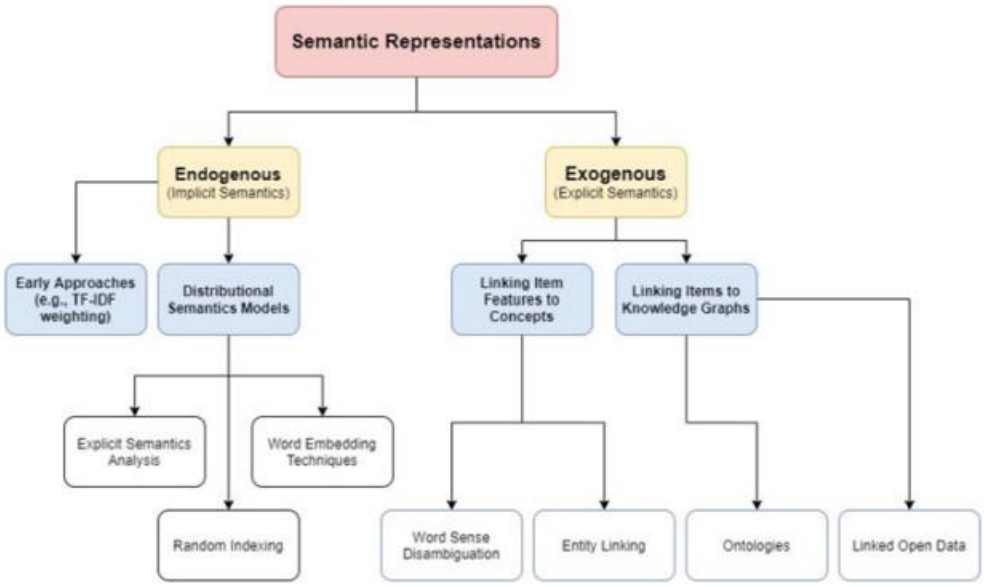
Процес препоруке изводи се у три корака, при чему сваким управља посебна компонента:

- **Анализатор садржаја** (енг. CONTENT ANALYZER) - Основни задатак ове компоненте је да представи садржај ставки (нпр. документи, описи производа итд.) у облику који је прикладан за наредне кораке обраде. Издваја карактеристике (кључне речи, n-граме, концепте...) из описа ставки и креира структурирану репрезентацију која се чува у репозиторијуму *Represented Items*. Типични системи за препоруке засновани на садржају користе једноставне моделе претраживања, као што је модел векторског простора. У том моделу, сваки документ је представљен вектором у вишедимензионалном простору, где свака димензија одговара термину из општег речника дате колекције докумената. Оваква репрезентација представља улазе за *PROFILE LEARNER* and *FILTERING COMPONENT.*

- **Модул за учење профила** (енг. PROFILE LEARNER) - Овај модул сакупља податке који представљају корисничке преференције и гради кориснички профил, модел који генерализује посматране податке. Преференције за ставке сакупљају се као оцене на дискретној скали и чувају се у репозиторијуму *Feedback*. Обично се стратегија генерализације реализује преко надгледаних алгоритама машинског учења, који изводе кориснички профил (*user profile*) из ставки и одговарајућих оцена.

- **Филтрирајућа компонента** (енг. FILTERING COMPONENT) - Овај модул предвиђа да ли је нова ставка вероватно интересантна за активног корисника (*active user*). Упоређује карактеристике у корисничком профилу са онима у репрезентацијама ставки и производи бинарну или континуалну пресуду о релевантности, при чему последњи случај резултира рангираним списком потенцијално интересантних ставки. Оцене се могу сакупљати на генерисаним препорукама, затим се процес учења поново изводи на новом скупу за обуку, и резултирајући профил се прилагођава ажурираним корисничким интересима. Итерација циклуса повратних информација и учења током времена омогућава систему да узме у обзир динамичку природу корисничких преференција.

Модел векторског простора може бити користан за развој веома једноставних интелигентних информационих система. Међутим, да би се суочили са проблемима који су уређено повезани са природним језиком, семантичке технике су кључне за прелазак са представе засноване на кључним речима на представу засновану на концептима ставки и корисничких профила. Слика 12 приказује класификацију семантичких техника. Ендогени (енг. **Endogenous**) приступи користе велике корпусе докумената да би закључили употребу речи, тј. њену имплицитну семантику. Егзогени (енг. **Exogenous**) приступи ослањају се на спољне изворе знања, као што су машински читљиви речници, таксономије или онтологије, за представљање ставки и корисничких профила.



Слика 12. Класификација техника семантичке репрезентације

Ендогена семантика - Технике за представљање ендогене семантике спадају у општу класу модела дистрибутивне семантике (DSMs) који су изворно представљени у

рачунарској лингвистици и когнитивним наукама. Ови приступи заснивају се на такозваној дистрибутивној хипотези која тврди да "Речи које се појављују у истим контекстима склоне су сличним значењима", па алгоритми који прате ове приступе извлаче информације о значењу речи анализирајући њену употребу у великим корпусима текстуалних докумената. DSMs се заснивају на Витгенштајновој идеји да је значење речи њена употреба у језику и да семантички сличне речи такође деле сличне контексте употребе.

Егзогена семантика - Приступи за егзогене семантичке репрезентације ослањају се на језичко, културно и позадинско знање које је кодирано и доступно кроз спољне базе знања. Главна разлика између ендогених и егзогених техника за семантичку репрезентацију је у природи база знања на које се ослањају. У првом случају, семантика се добија искоришћавањем неструктурираних података (корпуса) и директно се закључује из доступних информација. У другом, семантика долази изван, јер се добија рударењем и искоришћавањем података који су претходно кодирани у структурираним и спољним изворима знања.

Најпопуларнији структурирани извори знања који су данас доступни су: WordNet, BabelNet, The Linked Open Data cloud, Wikidata, Српски лексички речник (SLR), Отворени подаци за Србију (OPS), Српска Vikipedija у RDF формату, SrbData.

Предности система за препоруку заснованих на садржају у поређењу са КФ:

- **Независност од корисника** (енг. User Independence) - Системи препорука на основу садржаја искоришћавају само оцене које је дао активни корисник да би изградили његов профил. С друге стране, методе колаборативног филтрирања захтевају оцене од других корисника како би пронашли "најближе суседе" активног корисника или изградили напредне моделе машинског учења. Другим речима, мање су подложни проблемима прорећености података и могу бити ефикаснији када је доступна мала количина података.
- **Транспарентност** (енг. Transparency) - Објашњења о томе како систем препорука функционише могу се обезбедити тако што ће се експлицитно навести особине садржаја или описи који су узроковали да се ставка нађе на листи препорука. Те особине служе као индикатори које треба консултовати да би се одлучило да ли веровати препоруци. С друге стране, колаборативни системи су "црне кутије", јер је једино објашњење за препоруку ставке то што непознати корисници са сличним укусима воле ту ставку.
- **Нова ставка** (енг. New item) - Системи препорука на основу садржаја способни су да препоруче ставке које још нису оцењене од стране било ког корисника. Као последица, они не пате од проблема првог оцењивача, који погађа колаборативне системе препорука који се искључиво ослањају на корисничке преференције за давање препорука. Стога, док нова ставка не буде оцењена од доста корисника, систем не би могао да је препоручи.

Мане система за препоруку заснованих на садржају у поређењу са КФ:

- **Ограничена анализа садржаја** (енг. Limited content analysis) - Технике на основу садржаја имају природно ограничење у броју и типу карактеристика које су повезане, било аутоматски или ручно, са објектима које препоручују. Ниједан систем препоруке на основу садржаја не може да пружи одговарајуће предлоге

ако нису доступне описне карактеристике ставки. Наравно, скорашњи напредак у овој области (нпр. унапред обучени језички модели коришћени за обучавање уграђивања речи и структуриране карактеристике доступне у графовима знања) делимично је ублажио овај проблем, али потреба за садржајем је и даље обавезан захтев за овог приступа.

- **Превелика специјализација** (енг. Over specialization) - Ови системи немају уграђене методе за проналажење нечег неочекиваног. Систем предлаже ставке чији су резултати високи када се упореде са корисничким профилом, па ће кориснику бити препоручене ставке сличне онима које је већ оценио. Овај недостатак такође се назива проблемом "серендипитета" (енг. serendipity), како би се истакла тенденција система на основу садржаја да производе препоруке са ограниченим степеном новитета. На пример, када корисник оцени само филмове које је режирао Стенли Кјубрик, препоручиће му се само таква врста филмова. "Савршена" техника на основу садржаја би ретко пронашла нешто ново, ограничавајући опсег апликација за које би била корисна.
- **Нови корисник** (енг. New user) - Као и друге парадигме, и системи препорука на основу садржаја пате од проблема хладног почетка (*cold start problem*). Мора се прикупити довољно оцена пре него што систем препорука на основу садржаја може заиста да разуме корисничке преференције и даје тачне препоруке. Стога, када је доступан мали број оцена, као за новог корисника, систем неће бити у стању да пружи поуздане препоруке.

Препоруке засноване на знању (енг. Knowledge – based recommendations)

Већина комерцијалних система за препоруке ослања се на технике колаборативног филтрирања. КФ системи заснивају се искључиво на оценама корисника као јединим извором информација за генерисање препорука. Дакле, није потребно уносити додатне информације попут карактеристика доступних филмова. Технике препорука на основу садржаја, користе различите изворе информација за предвиђање да ли ће кориснику нешто свидети. Главни извори информација за ове системе укључују информације о категорији и жанру, као и кључне речи из описа предмета. И колаборативне и технике на основу садржаја имају своје предности. Међутим, постоје ситуације када нису најбољи избор. На пример, не купујемо често кућу или аутомобил, па КФ системи у таквим случајевима неће добро радити због малог броја оцена. Такође, време је важно, старе оцене за рачунаре или аутомобиле можда нису релевантне. Поред тога, у сложенијим доменима као што су аутомобили, корисници често желе да изразе своје захтеве, као што су максимална цена или боја. Таква формулација захтева није типична за чисте КФ и системе на основу садржаја.

Системи за препоруке засновани на знању помажу нам да се изборимо са претходно наведеним изазовима. Предност ових система је у томе што не постоје проблеми при почетку њихове употребе, јер нису потребни подаци о оценама за израчунавање препорука. Препоруке се формулишу независно од индивидуалних корисничких оцена: или у облику сличности између захтева купца и артикала или на основу експлицитних правила за препоруку. Традиционална тумачења система за препоруке фокусирају се на аспект филтрирања информација, где се ставке које су вероватно интересантне за одређеног купца издвајају. С друге стране, процес препоруке код апликација заснованих на знању је високо интерактиван, што је основ за њихову

карактеризацију као конверзационих система. Ова интерактивност је довела до промене тумачења са система за филтрирање ка широј дефиницији где се системи за препоруке дефинишу као системи који на персонализован начин воде корисника ка интересантним или корисним објектима у великом простору могућих опција. Системи за препоруке који се ослањају на изворе знања који нису искоришћени код колаборативних и приступа заснованих на садржају се подразумевано дефинишу као системи засновани на знању.

Два основна типа система за препоруке заснованих на знању су **системи засновани на ограничењима** (енг. Constraint-based) и **системи засновани на случајевима** (енг. case-based). Оба приступа су слична по питању процеса препоруке: корисник мора да наведе захтеве, а систем покушава да идентификује решење. Ако решење не може бити пронађено, корисник мора променити захтеве. Систем такође може да пружи објашњења за препоручене ставке. Међутим, ови системи за препоруке се разликују у начину на који користе пружено знање: **системи засновани на случајевима** се фокусирају на добијање сличних ставки на основу различитих типова мерења сличности, док се **системи засновани на ограничењима** ослањају на експлицитно дефинисани скуп правила за препоруку. У системима заснованим на ограничењима, скуп препоручених ставки се одређује, на пример, претрагом ставки које испуњавају правила препоруке. С друге стране, системи засновани на случајевима користе мере сличности да би пронашли ставке које су сличне (унутар предефинисаног прага) наведеним захтевима купца.

id	price(€)	mpix	opt-zoom	LCD-size	movies	sound	waterproof
p_1	148	8.0	4×	2.5	no	no	yes
p_2	182	8.0	5×	2.7	yes	yes	no
p_3	189	8.0	10×	2.5	yes	yes	no
p_4	196	10.0	12×	2.7	yes	no	yes
p_5	151	7.1	3×	3.0	yes	yes	no
p_6	199	9.0	3×	3.0	yes	yes	no
p_7	259	10.0	3×	3.0	yes	yes	no
p_8	278	9.1	10×	3.0	yes	yes	yes

Табела 5. Каталог ставки - Дигитална камера

Уопштено, системи засновани на знању ослањају се на детаљно знање о карактеристикама ставки. Пример таквог каталога ставки приказан је у Табели 5 за област дигиталних камера. Углавном, проблем препоруке састоји се од избора ставки из овог каталога које се подударају са потребама, преференцијама или строгим захтевима корисника. Захтеви корисника могу, на пример, бити изражени у терминима жељених вредности или опсега вредности за карактеристику ставке, попут "цена треба бити мања од 300e" или у терминима жељене функционалности, попут "камера треба бити прилагођена за спортску фотографију".

Пратећи категоризацију из претходног одељка, сада расправљамо о томе како је потребно доменско знање кодирано у типичним системима за препоруке заснованим на знању. Проблем препоруке заснован на ограничењима може, уопштено, бити представљен као проблем задовољења ограничења (енг. constraint satisfaction problem)

који може бити решен од стране *constraint solver*-а или у облику конјунктивног упита који се извршава и решава од стране базе података. Системи за препоруку засновани на случајевима (case-based) највише користе мере сличности за добијање ставки из каталога.

Главна предност система за препоруку заснованих на знању је непостојање проблема хладног почетка (*cold start*). Одговарајући **недостатак** је потенцијални застој у стицању знања изазван потребом да се знање о препорукама дефинише на експлицитан начин.

Хибридни системи за препоруке

Три најистакнута приступа препорукама о којима је било речи у претходним поглављима искоришћавају различите изворе информација и следе различите парадигме за давање препорука. Иако производе резултате који се сматрају персонализованим на основу претпостављених интересовања њихових примаоца, они показују различит степен успешности у различитим доменима примене. Колаборативно филтрирање искоришћава специфичну врсту информација, као што су оцене артикала, из модела корисника заједно са подацима заједнице да би се извеле препоруке, док се приступи засновани на садржају ослањају на карактеристике производа и текстуалне описе. Алгоритми засновани на знању, с друге стране, резонују на основу експлицитних модела знања из домена. Сваки од ових основних приступа има своје предности и мане, попут способности да се суочи са реткошћу података и проблемима почетног покретања или значајним напорима при стицању и инжењерингу знања. Систем за препоруке се може замислити као „црна кутија“ која претвара улазне податке у рангирану листу артикала као излаз. Модели корисника, контекстуалне информације, подаци заједнице и производа, и модели знања чине потенцијалне врсте улаза за препоруку. Међутим, ниједан од основних приступа није у стању да у потпуности искористи све ове моделе. Стога је изградња хибридних система који комбинују предности различитих алгоритама и модела да би се превазишли наведени недостаци и проблеми постала циљ скорашњих истраживања.

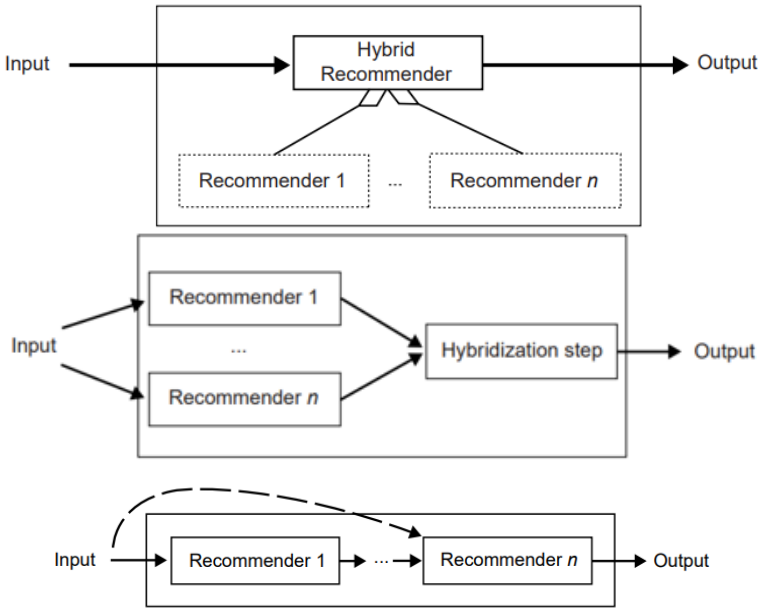
Paradigm	User profile and contextual parameters	Community data	Product features	Knowledge models
Collaborative	Yes	Yes	No	No
Content-based	Yes	No	Yes	No
Knowledge-based	Yes	No	Yes	Yes

Табела 7. Улазни подаци различитих алгоритма препоруке

Колаборативни принцип претпоставља да постоје групе корисника који се слично понашају и имају упоредиве потребе и преференције. Задатак колаборативног модела је да одреди сличне кориснике и изведе препоруке на основу њихових омиљених артикала. Док приступ заснован на садржају следи приступ "више истог" препоручујући артикле који су слични онима које је корисник волео у прошлости, препорука заснована на знању претпоставља додатни извор информација: експлицитно знање о персонализацији. Како је описано, ово знање може, на пример, бити у облику логичких ограничења која мапирају захтеве корисника на својства артикала. Одабир приступа модела препоруке одређује тип потребних улазних података. Четири различита типа постоје. **Кориснички**

модел и контекстуални параметри представљају корисника и конкретну ситуацију у којој се тренутно налази. Подаци о кориснику и специфичној ситуацији чувају се у корисничком профилу. Сви приступи препорукама требају приступ овом корисничком моделу како би персонализовали препоруке. Међутим, зависно о домену примене и сценарију употребе, могу бити доступни само ограничени делови корисничког модела. Стога нису све варијанте комбиновања могуће или препоручљиве у свакој области примене. Приказано је да приступи препорукама изборно захтевају **податке заједнице** (community data), **карактеристике производа** (product features) или **моделе знања** (knowledge models). На пример, колаборативно филтрирање ради искључиво на основу података заједнице и тренутног корисничког профила.

Друга димензија која карактерише хибридне алгоритме је њихов дизајн. Постоје три основна дизајна хибридизације: **монолитни, паралелизовани и хибриди у низу (pipeline)**. Монолитни комбинује више стратегија препоруке у једном алгоритму. Паралелизовани системи раде независно и производе различите листе препорука које се касније комбинују. У архитектури низа, излаз једног система постаје улаз за следећи.



Слика 13. Монолитни, паралелизовани и у низу хибридни дизајни

Пример 7. Монолитни дизајн - Табела 8 приказује оцене неколико корисника за каталог производа. Оцене представљају нееxплицитно посматране повратне информације корисника, попут куповина. Знање о производу ограничено је на жанр ставке. У чисто сарадничком приступу, без узимања у обзир карактеристике производа, и User1 и User2 били би сматрани подједнако сличним Алиси. Међутим, приступ комбинације карактеристика који предлажу Basu и колеге идентификује нове хибридне карактеристике на основу података из заједнице и производа.

Табела 9 приказује хибридно кодирање информација из Табеле 8. Ове карактеристике су изведене према следећим правилима: ако је корисник главном куповао књиге жанра X, постављамо карактеристику "Корисник воли много X књига" на тачно. Слично томе, такође постављамо "Корисник воли неколико X књига" на тачно, захтевајући трећину корисничких куповина. Иако се трансформација на први поглед

чини тривијалном, она ипак показује да нека знања, попут жанра ставке, могу довести до значајних побољшања.

Због једноставности приступа комбинацији карактеристика, многи тренутни системи за препоруку комбинују сарадничке и/или садржајне карактеристике на један или други начин. Међутим, комбинација улазних карактеристика помоћу заснованих на знању приступа остала је већим делом неистражена. Предлажемо да системи засновани на критици, који прикупљају повратне информације корисника у облику ограничења за конкретну ставку, могли бити погодна почетна тачка за будућа истраживања.

Feature	Alice	User1	User2	User3	User4
User likes many *mystery* books	true	true			
User likes some *mystery* books			true	true	
User likes many *romance* books					
User likes some *romance* books			true	true	
User likes many *fiction* books					
User likes some *fiction* books		true	true		true

User	R_{nav}	R_{view}	R_{ctx}	R_{buy}
Alice	n_3, n_4	i_5	k_5	$\emptyset$
User1	n_1, n_5	i_3, i_5	k_5	i_1
User2	n_3, n_4	i_3, i_5, i_7	$\emptyset$	i_3
User3	n_2, n_3, n_4	i_2, i_4, i_5	k_2, k_4	i_4

Табеле 8. и 9. Улазни атрибути и фидбек корисника

Паралелизовани - *Мешовита стратегија* хибридизације комбинује резултате различитих система за препоруку на нивоу корисничког интерфејса, у којем се резултати из различитих техника представљају заједно. Стога је резултат препоруке за корисника *u* и ставка *i* мешовите стратегије хибридизације је сет торки (score, k) за сваког од њених *n* конституишућих препоручивача rec_k :

$$rec_{mixed}(u, i) = \bigcup_{k=1}^{n} \langle rec_k(u, i), k \rangle \quad (15)$$

Приказују се најбоље оцењене ставке из различитих система препорука једне поред друге. Када се различити резултати комбинују у једну целину, потребна је нека форма разрешења конфликта. У домену ТВ апликација, примењују се предефинисана правила предности међу функцијама препоручивача. Описан је и хибридни систем који комбинује резултате из различитих система препорука, предлажући комбинације као што су смештај и активности. За решавање конфликата користи се CSP решавач, осигуравајући да су понуде усклађене са ограничењима, као што је близина активности и смештаја.

Pipelined hybridization design - Каскадни хибриди заснивају се на секвенцијалном редоследу техника, где сваки следећи препоручивач само усавршава препоруке свог претходника. Листа препорука следеће технике ограничена је на ставке које је препоручила и претходна техника. Формално, претпоставимо низ од *n* техника,

где rec_1 представља функцију препоруке прве технике, а rec_n последње. Консеквентно, коначна оцена препоруке за ставку рачуна се помоћу *n*th технике. Међутим, ставка ће бити предложена *k*th техником само ако је *(k − 1)*th техника такође доделила ненулту оцену. Ово важи за све k ≥ 2 по индукцији како је дефинисано у Формули 13.

$$rec_{cascade}(u, i) = rec_n(u, i) \tag{16}$$

где мора важити за свако $2k \geq 2$:

$$rec_k(u, i) = \begin{cases} rec_k(u, i) & : rec_{k-1}(u, i) \neq 0 \\ 0 & : \text{else} \end{cases} \tag{17}$$

У каскадном хибриду, све технике, осим прве, могу само да промене редослед листе препоручених ставки од свог претходника или да искључе ставку постављањем њене употребне вредности на 0. Међутим, не могу увести нове ставке, оне које су већ искључене од стране технике вишег приоритета, на листу препорука. Тако каскадне стратегије имају неповољну особину потенцијалног смањења величине скупа препорука приликом примене сваке додатне технике. Као последица, може доћи до ситуација где каскадни алгоритми не обезбеђују потребан број предлога, смањујући корисност система. Зато се каскадни хибриди могу комбиновати са стратегијом пребацивања у случају када каскадна стратегија не производи довољно препорука. Када се базирају на знању, рекомендери производе листе препорука које су или несортиране или садрже много изједначених ставки, комбиновањем ове са другом техником за сортирање резултата представља природан избор. На пример, постоји ресторански рекомендер базиран на знању који се комбинује са колаборативним алгоритмом за препоруку ресторана. Одступајући од дефиниције каскадног хибрида дате овде, он користи само други рекомендер да „разбије“ изједначене резултате.

Ризици примене система за препоруке

Сви системи препорука деле заједничку карактеристику: да би генерисали персонализоване препоруке, потребне су им информације о атрибутима, захтевима или преференцијама корисника. Обично, што су информације о кориснику детаљније, препоруке су прецизније. Провајдери услуга који управљају системима препорука прикупљају информације где год је то могуће како би обезбедили тачне препоруке. Информације могу бити аутоматски прикупљене или специфично пружене од стране корисника. Аутоматски прикупљене информације су резултат интеракције корисника са системима препорука и доношења одлука на основу препорука. На пример, прегледи страница на *Ebay*-у се користе за аутоматско представљање избора сличних препоручених артикала (recommendations for you). Слично томе, препоручени видео-садржаји на *YouTube*-у су под утицајем недавно прегледаних видеа. На основу куповина других корисника, артикли на *Amazon*-у су праћени понудама за пакете (frequently bought together) или сродним артиклима (customers who bought this item also bought). На основу посећених сајтова, *Google* пружа персонализоване огласе. На основу пријатеља и друштвених интеракција, *Facebook* предлаже нове пријатеље. *LinkedIn*, на основу CV-а корисника и веза, препоручује занимљиве компаније, понуде за посао и вести. Обрнуто, *LinkedIn* такође препоручује особе регрутерима који објављују нове пословне прилике. Многи сајтови за упознавање, као што су *Match.com* или *PAIQ*, препоручују парове својим корисницима. Много је примера таквих система и они ће наставити да постоје у будућности. Корисници такође могу специфично пружити информације, тиме корисници

граде свој лични профил специфицирајући шта им се свиђа и не свиђа, или садржећи опште информације о себи (као што су узраст и пол). На пример, *LastFM* и *YouTube* омогућавају корисницима да наведу своје фаворите. *Facebook* омогућава да се у профилу наведу личне информације као и интересовања.
Међутим, потенцијалне претње приватности корисника често се потцењују. Што су информације о кориснику детаљније, већа је претња за приватност корисника. Да би побољшали своје системе препорука, провајдери услуга прикупљају и консолидују све више информација. На пример, у недавним ажурирањима политике приватности, *Google* је навео да консолидује информације из свих својих услуга у један профил. *Facebook* наставља да шири свој домет на интернету, пружајући могућност да се садржај дели и лајкује.

Да би искористили функционалности система за препоручивање, корисници често морају да деле личне информације, што доводи до компромиса између корисности система и приватности корисника. У овом одељку ће се истражити приватност унутар система за препоручивање и придружене забринутости, с нагласком на заштиту приватности корисника.

Приватност и поверљивост

Приватност је термин са различитим суптилним значењима, углавном повезаним са контролом над личним информацијама на интернету.

Информациона приватност је "право појединца да контролише услове под којима се личне информације – информације које се могу идентификовати са појединцем – прикупљају, откривају или користе." [9]

Ова идеја о информационој приватности је уско повезана са појмом поверљивости из области информационе безбедности, али их не треба мешати. Поверљивост се бави очувањем тајности појединачних информација, док се информациона приватност фокусира на субјекта информација, утицај откривања на појединца и њихово право на контролу и пристанак. Анализа технологија за заштиту приватности ће се концентрисати на спречавање нежељеног откривања и коришћења информација, а не на последичне ефекте на појединца.

Када користе онлајн апликације, корисници деле велике количине информација, било кроз оцене, коментаре, личне профиле или куповине. Приватност осигурава да ове информације остану унутар њиховог намењеног опсега, који је дефинисан величином публике (breadth), обимом дозвољене употребе (depth) и трајањем (lifetime). До кршења приватности долази када се информација премести ван њеног намењеног опсега у било којој од ових димензија (било случајно или намерно). Дакле, **до кршења приватности може доћи када се информација открије страни за коју није била намењена, када се користи за непланирану сврху, или када се чува дуже него што је било предвиђено**.

Кршење приватности може укључивати различите стране (кориснике, провајдера услуга, или спољашње особе) и може бити намеран акт (завиривање, хаковање), или случајан (лоше управљање, задржавање података). У зависности од осетљивости укључених информација, такви инциденти могу имати озбиљне последице. Забринутост постоји због количине (личних) информација које сакупља провајдер услуга и

потенцијално цурење тих информација. Независно од њиховог рада, ми експлицитно идентификујемо забринутости приватности у системима за препоруке и класификујемо их на следећи начин:

- Сакупљање података (енг. Data Collection). Многи корисници нису свесни количине и обима информација које добављач услуга може сакупити, нити шта се може извести из ових информација. То може бити због чињенице да се изјаве о приватности ретко читају, и људи су навикли да обављају активности на интернету. Обично не постоји начин да се одбије такво прикупљање података, осим ако се систем уопште не користи. Пошто праксе сакупљања не одговарају очекивањима корисника, ова забринутост се односи на обим коришћења информација.
- Задржавање података (енг. Data Retention). Информације на интернету често је тешко уклонити, добављач услуга чак може намерно спречити или отежати уклањање података. Ово је зато што постоји комерцијална вредност у информацијама корисника, како за конкурентску предност путем анализе тако и/или продају података. Штавише, информације које су наводно избрисане са једног места могу и даље постојати негде другде у систему, на пример у резервним копијама, где их други могу пронаћи. Забринутост у вези са задржавањем података односи се на намеравани век трајања, пошто информације могу бити доступне дуже него што је намеравано.
- Продаја података (енг. Data Sales). Обиље информација које су сачуване у онлине системима вероватно је од вредности за треће стране и у неким случајевима може бити продато. Оцене корисника, преференције и историје куповине су све потенцијално занимљиве за маркетиншке сврхе. Продаја података обично је у сукобу са очекивањима корисника о приватности. Иако се подаци често анонимизују пре продаје како би се заштитила приватност корисника, поновно идентификовање је претња која се често превиђа или игнорише. На пример, информације које је објавио Netflix као део њихове награде за системе препорука, иако анонимизоване, омогућиле су поновно идентификовање. Истраживачи су повезали анонимизоване записе са јавно доступним записима (као што је IMDb) на основу сличности оцена и времена оцењивања. Ако два записа дају сличну оцену филму у приближно исто време, вероватно су од исте особе. Већи број сличних оцена филмова (по оцени и времену) повећава вероватноћу везе између записа. Ова забринутост се углавном односи на обим коришћења информација.
- Запослени који прегледају приватне информације (енг. Employee Browsing Private Information). Пружаоци услуга имају потпун приступ информацијама и њихови запослени могу то искористити, што је у супротности са очекиваном ширином публике и приватношћу коју је пружалац услуга обећао својим корисницима.
- Препоруке које откривају информације (енг. Recommendations Revealing Information). Препоруке се по дефиницији заснивају на информацијама из система за препоруке. На пример, у колаборативном филтрирању то су оцене свих корисника, или у системима заснованим на знању то је стручно знање. Свака препорука открива мали део приватних информација. Није јасно како велики број препорука утиче на откривање информација, што може открити информације о другим корисницима (компромитујући њихову приватност) или о самом систему за препоруке (што може довести до његовог обртања).
- Дељење уређаја или услуге (енг. Shared Device or Service). Приватност код куће може бити једнако важна као и приватност на интернету. Контрола приватности према породици и пријатељима може бити тешка када се дели уређај попут ТВ

пријемника или рачунара, или пријава на онлине услугу. На пример, жена која жели да сакрије од мужа да му је купила поклон. Ако нема приватан налог, муж може случајно видети њену куповину или добити препоруке засноване на њој.

- Странци прегледају приватне информације (енг. Stranger Views Private Information). Корисници могу погрешно претпоставити да ће неке информације остати ограничене на провајдера услуге или ограничену публику, када у стварности то није тако. Ово може бити због дизајнерских грешака од стране провајдера услуге или због недостатка разумевања или пажње корисника према његовој приватности. Када странац прегледа такве приватне информације, то је у супротности са намераваном ширином публике.

↓ Rec. Sys. / Info. →	Behavioural	Contextual	Domain Knowledge	Item Meta-data	History	Recommendation	Feedback	Social	User Attributes	User Preferences
Collaborative Filtering	•	·	·	·	•	●	·	·	·	●
Content-based	·	·	·	●	•	●	·	·	·	●
Demographic	·	·	●	·	·	●	·	·	●	·
Knowledge-based	•	·	●	•	·	●	•	·	•	●
Context-aware	•	●	•	•	·	·	·	·	•	·
Ensemble	·	·	·	·	·	●	·	·	·	·
Hybrid	·	·	·	·	·	●	·	·	·	·
Social	·	·	·	·	·	·	·	●	·	·

↓ Concern / Info. →	Behavioural	Contextual	Domain Knowledge	Item Meta-data	History	Recommendation	Feedback	Social	User Attributes	User Preferences
Data Collection	●	•	·	·	●	•	●	•	•	•
Data Retention	•	•	·	·	•	•	•	•	•	•
Data Sales	●	●	·	·	●	●	●	●	●	●
Employee	●	●	·	·	●	●	●	●	●	●
Recommendations	·	·	·	·	●	·	•	•	·	●
Shared Service	•	·	·	·	●	●	•	·	·	●
Stranger Views	·	•	·	·	●	•	•	●	●	●

Слика 14. Информације које су присутне у различитим типовима система за препоруке (лево). Забринутост за приватност корисничких информација (десно).

Системи за препоруке, који чувају обимне информације о својим корисницима, представљају главну мету за нападе. Осетљиви подаци могу завршити у погрешним рукама или их могу злоупотребити чак и они који су за то овлашћени. Због велике количине и детаљности података унутар система за препоруке, приватност заслужује озбиљан приступ. Преглед на слици 17 (десно) категорише утицај различитих врста информација у системима за препоруке према забринутости за приватност: висок (●), средњи (•) или низак (·). Знање о домену и метаподаци о ставкама обично имају низак утицај на приватне податке, пошто се односе на садржај а не на кориснике. Међутим, историја корисника и преференције су високо осетљиви због њиховог директног одражавања мишљења корисника о ставкама.

Једно од предложених решења [10] за проблем приватности корисника у системима за препоруке јесте **Федеративно учење** (енг. Federated Learning). Федеративно учење представља нови и веома обећавајући приступ у свету система за препоруке, који значајно доприноси заштити приватности корисника. Овај метод омогућава да се модели препорука тренирају коришћењем интермедијалних параметара, уместо директних података корисника. Тако се осигурава да лични подаци остану локално сачувани на уређајима корисника, чиме се значајно смањује ризик од њиховог непожељног откривања или злоупотребе. У системима федеративног учења за препоруке, познатим као *FedRS*, учесници (који могу бити мобилни уређаји или различите платформе) сарађују у тренирању глобалног модела препорука, размењујући само интермедијалне параметре, а не своје личне податке. Ово не само да штити приватност, већ омогућава и да се модел препорука обучава на богатом скупу података из различитих извора, чиме се побољшава његова тачност и релевантност. За заштиту података у *FedRS* користе се различити механизми, као што су хомоморфно шифровање, дељење тајни и диференцијална приватност. Ови методи омогућавају да се подаци

ефикасно шифрују или анонимизују, што додатно повећава безбедност и приватност корисника. Овај приступ се може применити у разним областима, од е-трговине и социјалних медија до здравственог сектора и персонализованих услуга.

У суштини, федеративно учење у системима за препоруке представља изузетно вредан напредак у области вештачке интелигенције, који пружа баланс између ефикасности у препорукама и строге заштите приватности корисника. Овај метод не само да унапређује перформансе система за препоруке, већ и уводи нову димензију у поштовању и заштити личних података у дигиталном добу.

ГРАФОВСКЕ НЕУРОНСКЕ МРЕЖЕ

Графовске неуронске мреже (енг. Graph Neural Networks) познате и као GNN, попримиле су огроман пораст популарности у последњих неколико година због способности да анализирају податке представљене у облику графова. GNN су коришћене у различитим индустријама, укључујући истраживање друштвених мрежа, претрагу нових лекова, системе препорука и предвиђање густине саобраћаја. Повећана популарност GNN резултовала је растућим интересовањем за ову тему међу научницима и стручњацима. Ове мреже су се показале као ефикасан начин за моделирање сложених структурираних података, као што су друштвене мреже, структуре протеина и обрасци саобраћаја. GNN су нарочито корисне за решавање ситуација у којима се подаци моделирају као граф, где чворови графа представљају ентитете, а ивице односе између тих ентитета. Због способности GNN да обављају резоновање и закључивање користећи структуру графа, идеално су подобне за решавање проблема у којима су односи између ентитета од примарног значаја.

Данас се област графовских неуронских мрежа експоненцијално проширује новим истраживањима и областима примене, а нове идеје и методе се развијају на редовној бази. Ова област је интердисциплинарна и базира се на концептима из различитих области, као што су информатика, математика и статистика. Очекује се да ће релевантност GNN расти како количина структурираних података настави да се повећава. GNN могу да извршавају задатке као што су класификација чвора, предвиђање везе и класификација графа користећи комбинацију атрибута чвора и ивице. Основна идеја иза ових мрежа је да искористе локалне шеме повезивања графа да би извукли информације и користили ово знање за предвиђање. Поред тога, GNN су способне да обрађују велике, сложене графове који садрже милионе или чак милијарде чворова и ивица. Велики графови често се користе да прикажу сложеност односа између различитих ентитета у различитим областима. Употребом GNN, истраживачи и стручњаци могу да изграде моделе за анализу ових графова који су тачнији, могу да се масовно примењују и лакше се тумаче. Ови модели омогућавају издвајање релевантних информација.

Граф

Графови су основна структура података са којима GNN ради и омогућавају представљање сложених односа и зависности између ентитета. Многи проблеми су лако изразиви и имају природне визуелне представе. Данас постоји широк спектар примена теорије графова у стварном животу: дизајнирање породичног стабла, рачунарске мреже, ток рачунања, организацију података, проналажење најкраћег пута на путу, дизајнирање веза круга, парсирање језичког стабла, конструисање молекуларне структуре и многе друге. Публикација коју је написао Ојлер 1736. године, у којој је решио проблем мостова у Кенигсбергу, сматра се годином рођења теорије графова. Важност графова у графовским неуронским мрежама (GNN) не може се оспорити. У многим стварним апликацијама као што су социјалне мреже, системи препорука, откривање лекова и предвиђање саобраћајног тока, подаци могу бити природно представљени као графови. Графови пружају флексибилан и моћан оквир за моделирање таквих података и хватање зависности између ентитета.

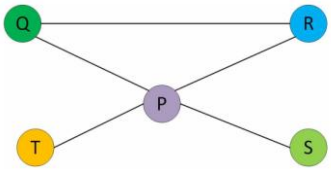
Дефиниција 2. [11] Граф G је уређени пар $(V(G),E(G))$ који се састоји од скупа $V(G)$ чворова и скупа $E(G)$, дисјунктног у односу на $V(G)$, ивица, заједно са функцијом инциденције ψG која свакој ивици G додељује неуређени пар (не обавезно различитих) чворова G. Ако је e ивица, а u и v су чворови тако да $\psi G(e)=\{u,v\}$, онда се каже да e спаја u и v, и чворови u и v се називају крајевима e. Означавамо број чворова и ивица у G са $v(G)$ и $e(G)$; ова два основна параметра се називају редом и величином G, респективно.

Пример 1. За $G=(V(G),E(G))$, где је:

$$V(G)=\{u,v,w,x,y\}$$
$$E(G)=\{a,b,c,d,e,f,g,h\} \quad (18)$$

и функција инциденције ψG је дефинисана као:

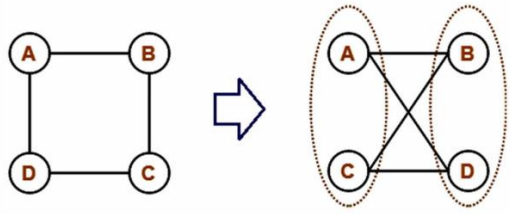
$$\psi G(a)=\{u,v\}\ \psi G(b)=\{u,u\}\ \psi G(c)=\{v,w\}\ \psi G(d)=\{w,x\}$$
$$\psi G(e)=\{v,x\}\ \psi G(f)=\{w,x\}\ \psi G(g)=\{u,x\}\ \psi G(h)=\{x,y\}$$



Слика 15. Граф G=(V,E) где је V={P,Q,R,S,T} и E={{P,Q},{Q,R},{P,R},{P,S},{P,T}}.

Графови су структуре које могу бити усмерене са специфицираним смеровима ивица или неусмерене. Прост граф је неусмерени граф без паралелних ивица и петљи. Нулти граф нема ивица, док мултиграф може имати више ивица између истих чворова. Графови могу бити коначни или бесконачни, повезани, где су свака два чвора повезана путем, или неповезани. Регуларни граф има све чворове са истим степеном. Комплетан граф повезује сваки пар чворова са тачно једном ивицом. Циклусни граф представља граф са једним пуним циклусом. Усмерени графови такође имају улазне и излазне степене за сваки чвор.

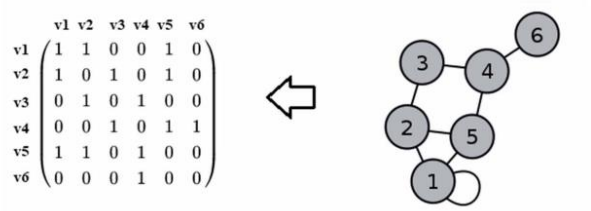
Графови који нас занимају јесу **бипартитни** или **двострани графови**, двострани графови имају широк спектар примене, укључујући откривање рака, рекламирање и системе рангирања у е-трговини, предвиђање преференција за филмове и храну. Граф са две различите половине назива се двострани граф. Граф је двострани ако се његови чворови могу поделити на два различита скупа X и Y. Чворови из скупа X могу се повезати само са чворовима из скупа Y, док чворови који припадају истом скупу нису међусобно повезани.



Слика 16. Бипартитни граф са четири ивице

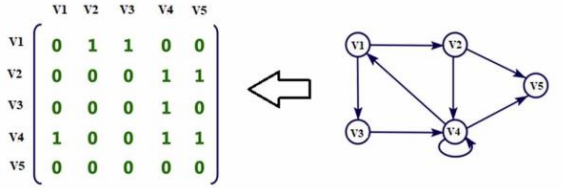
Репрезентација графа - Процес смештања графа у меморију рачунара назива се представљање или репрезентација графа. Овде се разматрају два главна начина за представљање графа.

Матрица суседства (енг. Adjacency matrix)- Матрица A димензија n x n, која је индексирана скупом V, јесте матрица суседства за једначину G = (V, E), а унос за (vi, vj) је дефинисан као: { $A_{vi,vj}$ = 1 ако vi, vj ∈ E, у супротном 0 }
Матрица суседства, позната и као матрица веза, је квадратна матрица са редовима и колонама која служи за представљање основног обележеног крајњег графа, са 0 или 1 на позицији (vi, vj) у зависности од тога да ли су vi и vj суседни или не. Други назив за матрицу суседства је матрица чворова. Ако основни граф не садржи петље, дијагонала матрице чворова треба да садржи 0.
Неусмерени графови су симетрични. Матрица веза је квадратни низ чији редови представљају излазне чворове графа, а колоне улазне чворове графа. Унос 1 представља ивицу графа између два чвора. i-ти ред и j-та колона имају исту вредност. Слика 19 представља матрицу суседства. Ако је граф тежински (енг. weighted), можемо сачувати тежину ивице уместо 1 и 0.



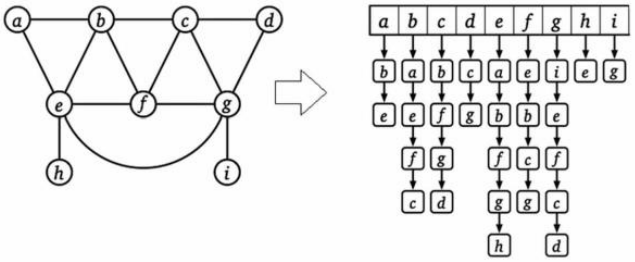
Слика 17. Матрица суседства неусмереног графа

Матрица суседства за усмерени граф - Ивице у усмереном графу означавају конкретну путању која може бити пређена да би се стигло од једног чвора до другог. Ако постоји пут који води од чвора A до чвора B, онда се чвор A сматра почетним чвором, а чвор B завршним чвором. Слика 20 приказује матрицу суседства за усмерени граф.



Слика 18. Матрица суседства усмереног графа

Листе суседства (енг. Adjacency lists) - Листе суседства изражавају графове као повезане листе. Већина графова користи листе суседства. Њихов необичан облик олакшава разликовање који су чворови суседни. Повезане листе омогућавају чворовима графа да се позивају на своје комшије. За сваки чвор у графу, чува се листа суседства која садржи вредност чвора и показивач на следећи суседни чвор одговарајућег чвора. Ако су сви суседни чворови посећени, показивачки део крајњег чвора у листи треба поставити на *NULL*. Слика 16 приказује листу суседства.



Слика 19. Листа суседства

Хетерогени граф - Хетерогени граф је тип графа у коме чворови и ивице могу имати различите типове и атрибуте. Другим речима, сваки чвор и ивица у графу могу припадати различитој категорији или класи и могу имати различите особине или карактеристике. Хетерогени графови се користе за представљање сложених система који укључују више типова објеката и односа, као што су друштвене мреже, системи препорука и знања. У хетерогеном графу, чворови и ивице су обично означени типовима или атрибутима који описују њихове особине. На пример, у друштвеној мрежи, чворови могу бити означени атрибутима као што су старост корисника, пол, занимање и интересовања, док ивице могу бити означене атрибутима као што су пријатељство, праћење или свиђање. У систему препорука, чворови могу бити означени атрибутима као што су преференције корисника, карактеристике производа и оцене, док ивице могу бити означене атрибутима као што су куповина, преглед или клик.

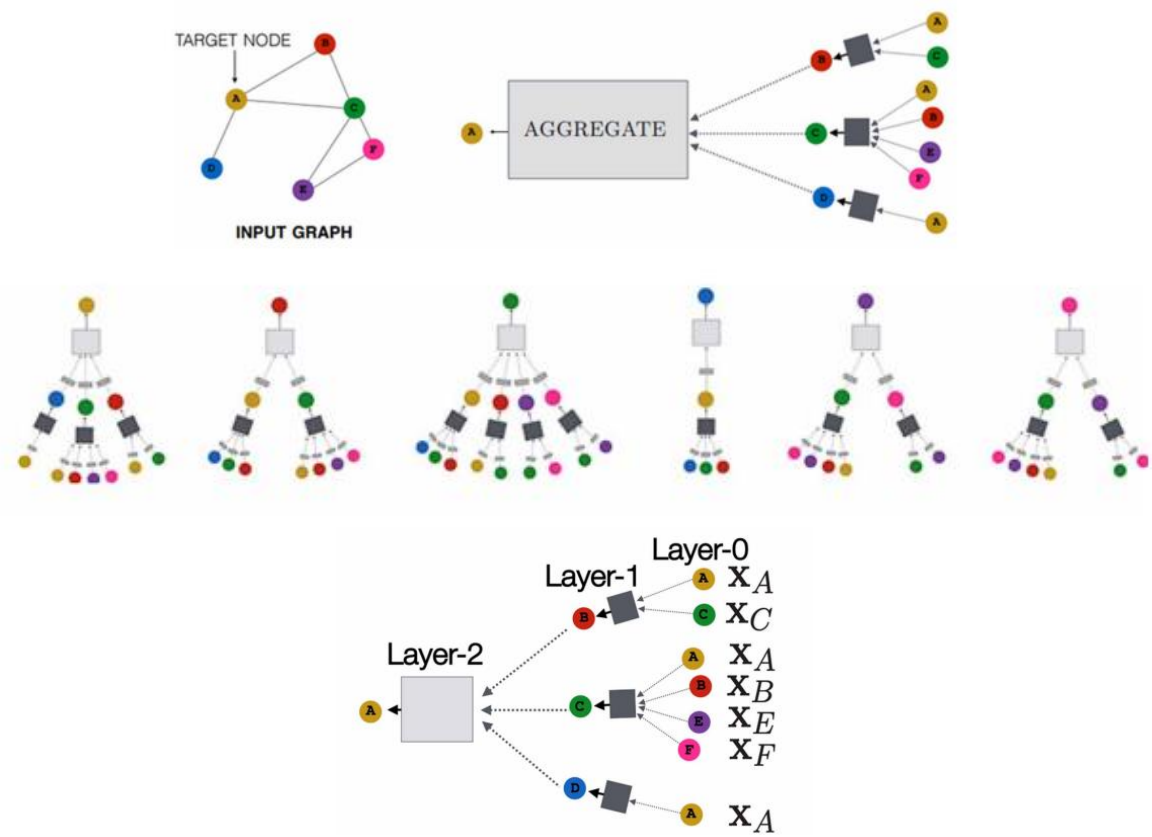
Хетерогени графови имају неколико предности у односу на хомогене графове, где сви чворови и ивице имају исти тип. Прво, могу представљати сложеније односе између објеката, као што су хијерархијски односи, мреже са више односа и контекстуалне зависности. Друго, могу захватити богатије информације о објектима и њиховим особинама, које могу помоћи у побољшању модела машинског учења и анализи података. Коначно, они могу омогућити флексибилније упите и добијање информација из графа, пошто корисници могу навести упите који укључују различите типове чворова и ивица. Хетерогени графови имају неколико примена у машинском учењу и обради података, укључујући системе препорука, усаглашавање ентитета, предвиђање веза и уграђивање графа. Показано је да алгоритми засновани на хетерогеним графовима надмашују традиционалне алгоритме засноване на хомогеним графовима у многим задацима, посебно када се ради са сложеним и разноликим подацима. [12]

Репрезентација графа у простору ниже димензије

Репрезентација графа у простору ниже димензије (енг. Graph Embedding) је техника која се користи у графовским неуронским мрежама за представљање сваког чвора и целог графа као вектора ниске димензије. Ове репрезентације обухватају важне структуралне и семантичке информације о чворовима и ивицама у графу, које се могу користити за широк спектар каснијих задатака, као што су класификација чворова, предвиђање веза и груписање графова. Чвор у графу може се посматрати из два домена: (1) оригинални графовски домен, где су чворови повезани ивицама (или графовска структура), и (2) домен ембединга (Embedding), где је сваки чвор представљен као континуирани вектор. Стога, из ове дводоменске перспективе, уграђивање графа има циљ да преслика сваки чвор из графовског домена у домен ембединга тако да се информације у графовском домену очувају у домену ембединга.

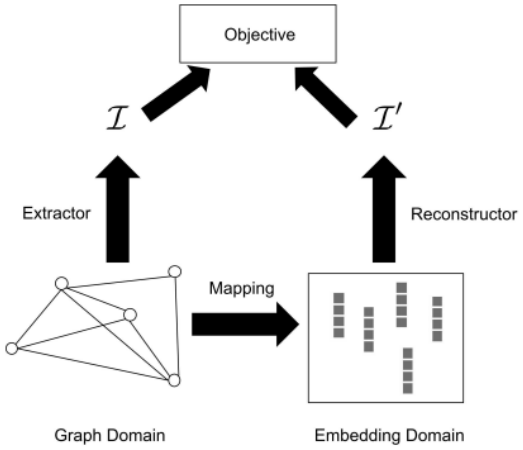
Постоје различити начини за креирање ових репрезентација у GNN-овима, али најчешћи приступ је преко преношења порука (енг. **Message passing**). У овом приступу, сваки чвор шаље и прима поруке од својих суседа, и агрегира ове информације да би ажурирао своју репрезентацију. Поруке обично садрже информације о суседним чворовима и ивицама, као што су њихове карактеристике или тежине, а функције агрегације могу бити једноставне као збир (енг. sum) или максимум (енг. max), или комплексније као што су механизам пажње (енг. **Attention mechanism**) или конволција графа (енг. **Graph convolution).** Како се процес преношења порука наставља кроз више итерација или слојева, репрезентације чворова и графа постају све прецизније и информативније, што омогућава GNN-овима да ухвате комплексне обрасце и зависности о структури и карактеристикама графа. Крајње репрезентације чворова и графа могу се затим користити као улази за касније задатке или као карактеристике за визуализацију и анализу.
Узмимо граф G = (V, A, X) такав да је: V - скуп чворова, A - је матрица суседства, X ∈ $R^{m \times |V|}$ је матрица карактеристика чвора

Објашњење како један чвор сакупља поруке примљене из свог непосредног окружења. Модел сакупља поруке од локалних суседа A (на пример, B, C и D), а поруке које долазе од ових суседа заснивају се на информацијама прикупљеним из њихових респективних околина. Овај процес се наставља док се не сакупе све поруке. Парадигма преношења порука приказана је у варијанти са два слоја која је приказана на слици 22. Вреди напоменути да рачунарска структура GNN-а узима облик *tree* када се креира окружење око циљног чвора.



Слика 20. Генерализована конволуција, агрегација и пример computational графа, за случај мреже са 2 слоја

Постављају се два кључна питања: (1) Које информације треба очувати? и (2) Како очувати те информације? Различити алгоритми за ембединге графа често дају различите одговоре на ова два питања. За прво питање, истраживано је много врста информација, као што су информације о суседству чвора, структурална улога чвора, статус чвора и информације о заједници. Постоје различите методе предложене за одговор на друго питање. Иако се технички детаљи ових метода разликују, већина њих дели исту идеју, која је реконструкција информација из графовског домена које желимо да сачувамо коришћењем представа чворова у домену ембединга. Интуиција је да добре представе чворова треба да буду у стању да реконструишу информације које желимо да очувамо. Стога, пресликавање се може научити минимизирањем грешке у реконструкцији. Уопштени процес ембединга графа се може резимирати са четири кључне компоненте:



Слика 21. Општи принцип за генерисање ембединга графа.

- Функција мапирања (енг. **mapping function**), која мапира чвор из домена графа у домен ембединга.
- Екстрактор информација (енг. **information extractor**), који извлачи кључне информације *I* које желимо да сачувамо из домена графа.
- Реконструктор (енг. **reconstructor**), за конструкцију извучених информација о графу *I* користећи уграђивања из домена уграђивања. Реконструисане информације су означене као *I'*,
- Циљ (енг. **objective**) заснован на извученим информацијама *I* и реконструисаним информацијама *I'*. Обично оптимизујемо циљ да бисмо научили све параметре укључене у мапирање и/или реконструктор.

Ембединг графа за једноставне графове (Simple Graphs) – у овом делу ће бити описани алгоритми за ембединг графа за једноставне графове који су статични, неусмерени, неозначени и хомогени. Организујемо алгоритме према информацијама које покушавају да очувају, укључујући сусретање чворова (енг. node co-occurrence), структуралну улогу, статус чвора и структуру заједнице. Један од најпопуларнијих начина за издвајање сусретања чворова у графу је путем случајних шетњи (енг. **random walks**). Чворови се сматрају сличним ако имају тенденцију да се често сусрећу у овим шетњама. Функција уграђивања је оптимизована тако да научене представе чворова могу реконструисати "сличност" изведену из случајних шетњи. Један репрезентативан алгоритам уграђивања мреже који чува сусретање чворова је *DeepWalk*.

Функција за мапирање - Функција уграђивања $f(v_i)$ може се директно дефинисати коришћењем табеле за претрагу. То значи да добијамо ембединг чвора u_i, на основу његовог индекса i. Конкретно, функција ембединга је имплементирана као:

$$f(v_i)=u_i=e_i^T W \quad (19)$$

где је $e_i \in \{0,1\}^N$ са $N=|V|$ *one-hot encoding* чвора v_i. Посебно, e_i садржи само један елемент $e_i\,[i]=1$ и сви остали елементи су 0. $W^{N\times d}$ су параметри ембединга који се уче, где је d димензија уграђивања. i-ти ред матрице W означава представљање (или ембединг) чвора v_i. Стога, број параметара у функцији ембединга је $N\times d$.

Сусретање засновано на случајним шетњама (енг. Random walks) - Узевши почетни чвор $v^{(0)}$ у графу G, случајно прелазимо на једног од његових суседа. Овај процес понављамо са чвором док не посетимо T чворова. Ова случајна секвенца посећених чворова је случајна шетња дужине T на графу. Формално дефинишемо случајну шетњу на следећи начин:

Дефиниција 3 (Случајна Шетња): Нека $G=\{V,E\}$ означава повезани граф. Разматрамо случајну шетњу која почиње на чвору $v^{(0)} \in V$ на графу G. Претпоставимо да смо у t-ом кораку случајне шетње на чвору $v^{(t)}$ и онда настављамо случајну шетњу бирајући следећи чвор према следећој вероватноћи:

$$p(v^{(t+1)}|v^{(t)}) = \begin{cases} \frac{1}{d(v^{(t)})}, & if\ v^{(t+1)} \in \mathcal{N}(v^{(t)}) \\ 0, & otherwise, \end{cases} \quad (20)$$

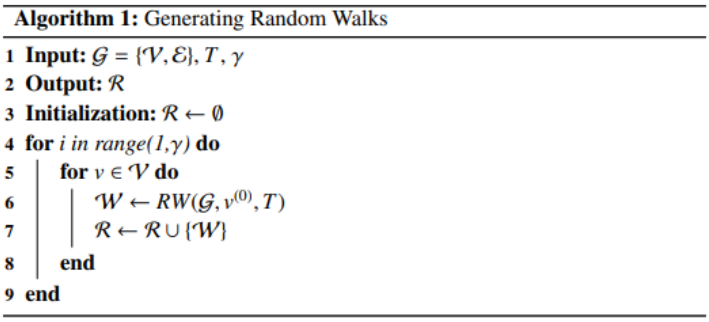
где $d(v^{(t)})$ означава степен чвора $v^{(t)}$ и $N(v^{(t)})$ је скуп суседа чвора $v^{(t)}$. Другим речима, следећи чвор се случајно бира из суседа тренутног чвора пратећи униформну дистрибуцију. Користимо генератор случајних шетњи да сумирамо горњи процес на следећи начин:

$$\mathcal{W} = \mathrm{RW}(\mathcal{G}, v^{(0)}, T), \quad (21)$$

где $W=(v^{(0)},\ldots,v^{(T-1)})$ означава генерисану случајну шетњу, где је $v^{(0)}$ почетни чвор и T је дужина случајне шетње.

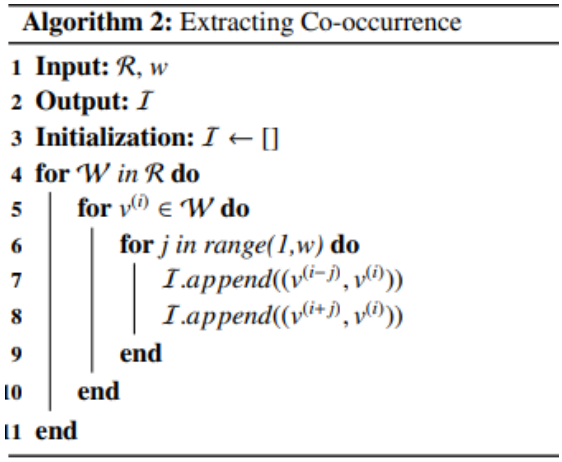
Случајне шетње су коришћене као мера сличности у разним задацима као што су препорука садржаја и детекција заједница. У *DeepWalk* алгоритму, скуп кратких случајних шетњи се генерише из датог графа, и екстракција сусретања чворова се врши из ових случајних шетњи. Детаљно ће бити описан процес генерисања скупа случајних шетњи и екстракцију сусретања из њих.

Да би генерисали случајне шетње које могу захватити информације целог графа, сваки чвор се сматра почетним чвором за генерисање γ случајних шетњи. Стога, укупно постоји $N\cdot\gamma$ случајних шетњи. Овај процес је приказан у Алгоритму 1. Улаз у алгоритам укључује граф G, дужину T случајне шетње и број случајних шетњи γ за сваки почетни чвор. Од линије 4 до линије 8 у Алгоритму 1, генеришемо γ случајних шетњи за сваки чвор у V и додајемо ове случајне шетње у R. На крају, R, који се састоји од $N\cdot\gamma$ генерисаних случајних шетњи, је излаз алгоритма.



Слика 22. Алгоритам случајних шетњи (Random walks) [13]

Ове случајне шетње могу се посматрати као реченице у "вештачком језику" где је скуп чворова V његов речник. Алгоритам Skip-gram у моделирању језика покушава да сачува информације у реченицама каптирајући односе сусретања између речи у тим реченицама. За дату централну реч у реченици, речи које су унутар одређене удаљености w од централне речи се третирају као њен "контекст". Затим се сматра да централна реч сусреће са свим речима у свом контексту. Циљ алгоритма Skip-gram је сачувати такве информације о сусретању. Ови концепти су усвојени за случајне шетње како би се екстраховали односи сусретања између чворова. Конкретно, означавамо сусретање два чвора као торку (v_{con}, v_{cen}), где v_{cen} означава централни чвор, а v_{con} указује на један од његових контекст чворова. Процес издвајања односа сусретања између чворова из случајних шетњи приказан је у Алгоритму 2. За сваку случајну шетњу $W \in R$, итерирамо преко чворова у случајној шетњи (линија 5). За сваки чвор $v^{(i)}$, додамо $(v^{(i-j)}, v^{(i)})$ и $(v^{(i+j)}, v^{(i)})$ у листу сусретања I за $j=1,\ldots,w$ (од линије 6 до линије 9). U случајевима када је $i-j$ или $i+j$ изван опсега случајне шетње, једноставно их игноришемо. За дати централни чвор, третирамо све његове контекст чворове једнако без обзира на удаљеност између њих. Контекст чворови се третирају различито у зависности од њихове удаљености од централног чвора.



Слика 23. Процес издвајања сусретања између чворова у случајним шетњама

Реконструктор и циљ - Са функцијом мапирања и информацијама о сусретању чворова, разматрамо процес реконструкције информација о сусретању коришћењем репрезентација у домену уграђивања. Да бисмо реконструисали информације о сусретању, покушавамо да закључимо вероватноћу посматрања торки у I. За било коју дату торку $(v_{con}, v_{cen}) \in I$, постоје две улоге чворова, тј. централни чвор v_{cen} и контекст чвор v_{con}. Чвор може играти обе улоге, тј. бити централни чвор и контекст чвор других чворова. Стога се користе две функције мапирања да би се генерисале две репрезентације

чвора за сваки чвор који одговара његовим двема улогама. Оне се могу формално изразити као:

$$f_{cen}(v_i) = \mathbf{u}_i = e_i^\top \mathbf{W}_{cen}$$
$$f_{con}(v_i) = \mathbf{v}_i = e_i^\top \mathbf{W}_{con}. \quad (22)$$

За торку (v_{con},v_{cen}), однос сусретања може се објаснити као посматрање v_{con} у контексту централног чвора v_{cen} . Са две функције мапирања f_{cen} и f_{con}, вероватноћа посматрања v_{con} у контексту v_{cen} може се моделовати коришћењем softmax функције као што следи:

$$p(v_{con}|v_{cen}) = \frac{exp(f_{con}(v_{con})^\top f_{cen}(v_{cen}))}{\sum_{v \in V} exp(f_{con}(v) f_{cen}^\top(v_{cen}))}, \quad (23)$$

која се може сматрати реконструисаном информацијом из домена уграђивања за торку (v_{con},v_{cen}). За било коју дату торку (v_{con},v_{cen}), реконструктор *Rec* може вратити вероватноћу у једначини изнад, која је сажета као:

$$Rec((v_{con}, v_{cen})) = p(v_{con}|v_{cen}). \quad (24)$$

Ако можемо тачно закључити оригиналне информације графа I из домена уграђивања, издвојене информације I могу се сматрати добро реконструисаним. Да бисмо постигли циљ, функција *Rec* треба да врати високе вероватноће за издвојене торке у I и ниске вероватноће за случајно генерисане торке. Претпостављамо да су ове торке у сусретању I независне једна од друге као у *Skip-gram* алгоритму. Стога, вероватноћа реконструкције I може се моделовати на следећи начин:

$$I' = Rec(I) = \prod_{(v_{con},v_{cen}) \in I} p(v_{con}|v_{cen}). \quad (25)$$

Могу постојати дупликати торки у I. Да бисмо уклонили ове дупликате у једначини, преформулишемо је на следећи начин:

$$\prod_{(v_{con},v_{cen}) \in set(I)} p(v_{con}|v_{cen})^{\#(v_{con},v_{cen})}, \quad (26)$$

где $set(I)$ означава скуп јединствених торки у I без дупликата и #(v_{con},v_{cen}) је фреквенција торки (v_{con},v_{cen}) у I. Дакле, торке које се чешће појављују у I више доприносе укупној вероватноћи у једначини изнад. Да бисмо осигурали бољу реконструкцију, потребно је научити параметре функција мапирања тако да се једначина може максимизирати. Тако се уграђивања чворова W_{con} и W_{cen} (или параметри две функције мапирања) могу научити минимизирањем следећег циља:

$$\mathcal{L}(\mathbf{W}_{con}, \mathbf{W}_{cen}) = -\sum_{(v_{con},v_{cen}) \in set(I)} \#(v_{con}, v_{cen}) \cdot \log p(v_{con}|v_{cen}), \quad (27)$$

где је циљ негативан логаритам поменуте једначине.

Ембединг графа за комплексне графове [13] **(**енг. Complex graphs**) -** Хетерогени графови – У хетерогеним графовима постоје различити типови чворова. Предложен је приступ за ембединге хетерогених мрежа (енг. Heterogeneous Network Embedding - HNE) како би се различити типови чворова у хетерогеном графу пројектовали у заједнички простор ембединга. Да би се постигао овај циљ, усвојена је посебна функција мапирања за сваки тип. Претпоставља се да су чворови повезани са карактеристикама чворова које могу имати различите облике (нпр. слике или текстови) и димензије. Зато се различити дубоки модели користе за сваки тип чвора да би се одговарајуће карактеристике

мапирале у заједнички простор ембединга. На пример, ако је повезана карактеристика у облику слика, CNN (конволуционе неуронске мреже) се усвајају као функција мапирања. HNE циља на очување парних веза између чворова. Стога, екстрактор у HNE издваја парове чворова са ивицама као информације за реконструкцију, што се може природно означити матрицом суседства A. Дакле, реконструктор треба да опорави матрицу суседства A из ембединга чворова. Специфично, дати пар чворова (v_i,v_j) и њихови ембединзи u_i,u_j научени помоћу функција мапирања, вероватноћа реконструисаног елемента матрице суседности $\tilde{A}_{i,j}=1$ рачуна се на следећи начин:

$$p(\tilde{A}_{i,j}=1)=\sigma(u_i^T u_j), \quad (28)$$

где је σ сигмоидна функција. Кореспондирајуће,

$$p(\tilde{A}_{i,j}=0)=1-\sigma(u_i^T u_j). \quad (29)$$

Циљ је максимизирати вероватноћу тако да реконструисана матрица суседства $\tilde{A}$ буде блиска оригиналној матрици суседства A. Стога, циљ је моделиран помоћу унакрсне ентропије на следећи начин:

$$-\sum_{i,j=1}^{N}\left(\mathbf{A}_{i,j}\log p(\tilde{\mathbf{A}}_{i,j}=1)+(1-\mathbf{A}_{i,j})\log p(\tilde{\mathbf{A}}_{i,j}=0)\right) \quad (30)$$

Функције мапирања могу се научити минимизирањем циља у једначини изнад где се могу добити ембединзи. У хетерогеним графовима, различити типови чворова и ивица носе различита семантичка значења. Стога, за уграђивање хетерогене мреже, треба бринути не само о структуралним корелацијама између чворова већ и о њиховим семантичким корелацијама. ***metapath2vec*** је предложен да ухвати обе корелације између чворова. Функција мапирања у metapath2vec је иста као у *DeepWalk*.

Информациони Екстрактор заснован на мета-путањи (енг. meta-path-based Information Extractor) - Да би се обухватиле и структуралне и семантичке корелације, уведене су случајне шетње засноване на мета-путањама да би се екстраховале информације о сусретању. Специфично, мета-путање се користе да би се ограничиле одлуке случајних шетњи. Прво ће бити уведен концепт мета-путања, а потом описано како дизајнирати случајне шетње засноване на мета-путањи.

Дефиниција 4. (Шема мета-путање) Дат је хетерогени граф G како је дефинисан у Дефиницији 2 шема мета-путање ψ је мета-шаблон у G означен као

$$A_1 \xrightarrow{R_1} A_2 \xrightarrow{R2} \cdots \xrightarrow{R_l} A_{l+1} \quad (31)$$

где $A_i \in T_n$ и $R_i \in T_e$ означавају одређене типове чворова и ивица, респективно. Шема мета-путање дефинише композитну релацију између чворова од типа A_1 до типа A_{l+1} где се релација може означити као $R=R1\circ R2\circ\cdots\circ Rl-1\circ Rl$. Пример шеме мета-путање ψ је мета-путања, где сваки чвор и ивица у путу прате одговарајуће типове у шеми. Шема мета-путање може се користити за вођење случајних шетњи. Случајна шетња заснована на мета-путањи је случајно генерисани пример дате шеме мета-путање ψ. Формална дефиниција случајне шетње засноване на мета-путањи дата је у наставку.

Дефиниција 5. Дата је шема мета-путање ψ:

$$A_1 \xrightarrow{R_1} A_2 \xrightarrow{R2} \cdots \xrightarrow{R_l} A_{l+1} \quad (32)$$

вероватноћа транзиције случајног хода вођеног помоћу ψ може се израчунати као:

$$p(v^{(t+1)}|v^{(t)},\psi)=\begin{cases}\frac{1}{\left|\mathcal{N}_{t+1}^{R_t}(v^{(t)})\right|}, & if\ v^{(t+1)}\in\mathcal{N}_{t+1}^{R_t}(v^{(t)}),\\ 0, & otherwise,\end{cases} \quad (33)$$

где је $v^{(t)}$ чвор типа A_t, што одговара позицији A_t у шеми мета-путање. $N^{Rt}_{t+1}(v^{(t)})$ означава скуп суседа чвора $v^{(t)}$ који имају тип чвора A_{t+1} и везани су са $v^{(t)}$ преко типа ивице R_t. Формално се може дефинисати као:

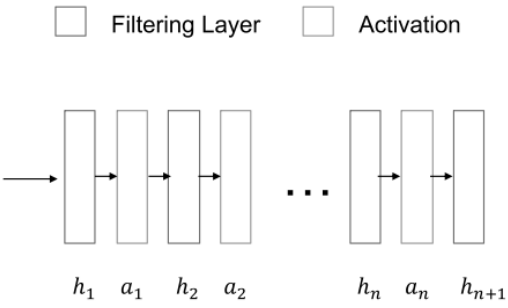
$$\mathcal{N}^{R_t}_{t+1}(v^{(t)}) = \{v_j \mid v_j \in \mathcal{N}(v^{(t)}) \text{ and } \phi_n(v_j) = A_{t+1} \text{ and } \phi_e(v^{(t)}, v_j) = R_t\}, \quad (34)$$

где $\phi_n(v_j)$ је функција за добијање типа чвора v_j а $\phi_e(v^{(t)},v_j)$ је функција за добијање типа ивице $(v^{(t)},v_j)$.

Архитектура GNN

У овом делу биће описане основне архитектуре графовских неуронских мрежа за задатке фокусиране како на чворове тако и на графове у целини. Граф означавамо са $G=\{V, E\}$. Матрица суседности графа са N чворова означена је са A. Повезане карактеристике представљене су као $F \in \mathrm{R}^{N\times d}$. Сваки ред F одговара једном чвору, а d је димензија карактеристика.

Основна архитектура задатака фокусираних на чворове (енг. Node-Focused Tasks)



Слика 24. Основна архитектура задатака фокусираних на чворове

Општа архитектура за задатке фокусиране на чворове може се сматрати композицијом филтрирања графа и нелинеарних активационих слојева. Архитектура GNN са L слојева филтрирања графа и $L-1$ активационих слојева приказан је на слици 26, где $h_i()$ и $\alpha_i()$ означавају i-ти слој филтрирања графа и активациони слој, респективно. Користимо $F^{(i)}$ да означимо излаз i-тог слоја филтрирања графа. Специфично, $F^{(0)}$ је иницијализован као повезани атрибути F. Даље, користимо d_i да означимо димензију излаза i-тог слоја филтрирања графа. Пошто се структура графа не мења, имамо $F^{(i)} \in \mathrm{R}^{N\times di}$. i-ти слој филтрирања графа може се описати као:

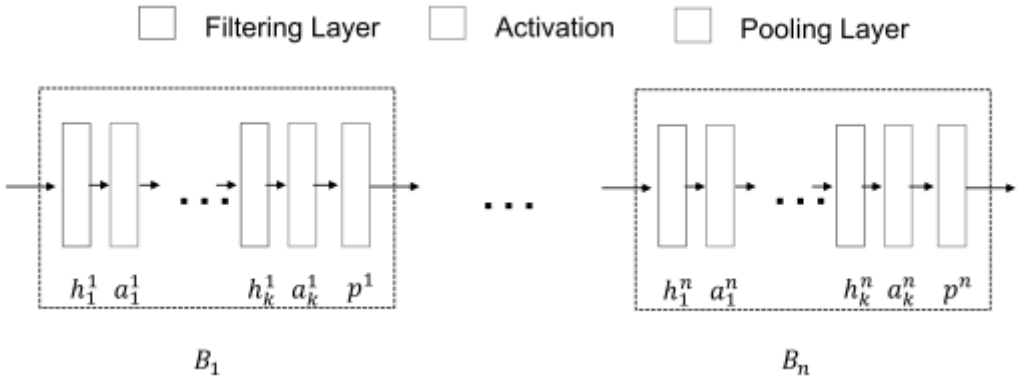
$$F^{(i)} = h_i(A, \alpha_{i-1}(F^{(i-1)})), \quad (35)$$

где је $\alpha_{i-1}()$ функција активације која следи након $(i-1)$-ог слоја филтрирања графа. Крајњи излаз $F(L)$ се користи као улаз за неке специфичне слојеве у складу са *downstream* задацима фокусираним на чворове.

Основна архитектура задатака фокусираних на графове (енг. Graph-Focused Tasks)

Општа архитектура GNN (Графовска Неуронска Мрежа) за задатке фокусиране на граф састоји се од три врсте слојева: слој филтрирања графа, активациони слој и слој

груписања (пулинг) графа. Слој филтрирања графа и активациони слој у архитектури имају функционалности сличне онима у архитектури фокусираној на чворове. Користе се за генерисање бољих атрибута чворова. Слој груписања графа користи се за сажимање атрибута чворова и генерисање атрибута вишег нивоа који могу ухватити информације целог графа. Типично, слој груписања графа следи након серије слојева филтрирања графа и активационих слојева. Након слоја груписања графа генерише се грубљи граф са апстрактнијим и атрибутима чворова вишег нивоа. Ови слојеви могу бити организовани у блок као што је приказано на слици 25, где h_i, α_i и p означавају i-ти слој филтрирања, i-ти активациони слој и слој груписања у овом блоку.



Слика 25. Основна архитектура задатака фокусираних на графове

Улаз у блок је матрица суседства $A^{(ib)}$ и атрибути $F^{(ib)}$ графа $G_{ib} = \{V_{ib}, E_{ib}\}$ и излаз су ново генерисана матрица суседства $A^{(ob)}$ и атрибути $F^{(ob)}$ за грубљи граф $G_{(ob)} = \{V_{ob}, E_{ob}\}$. Процедура израчунавања блока формално је дефинисана као:

$$\begin{aligned} \mathbf{F}^{(i)} &= h_i(\mathbf{A}^{(ib)}, \alpha_{i-1}(\mathbf{F}^{(i-1)})) \quad \text{for} \quad i = 1, \ldots k, \\ \mathbf{A}^{(ob)}, \mathbf{F}^{(ob)} &= p(\mathbf{A}^{(ib)}, \mathbf{F}^{(k)}), \end{aligned} \tag{36}$$

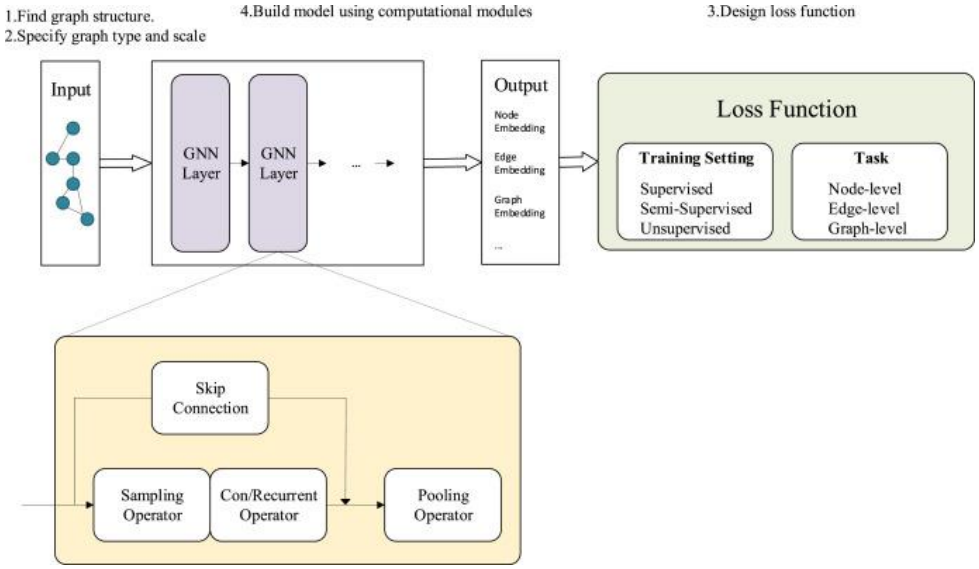
где α_i представља активациону функцију за i !=0 где је α_0 идентитетска функција и $F^{(0)} = F^{ib}$. Можемо сажети горе наведени процес израчунавања блока као:

$$\mathbf{A}^{(ob)}, \mathbf{F}^{(ob)} = B(\mathbf{A}^{(ib)}, \mathbf{F}^{(ib)}). \tag{37}$$

Цела архитектура GNN може се састојати од једног или више ових блокова како је приказано на слици 28. Процес израчунавања архитектуре GNN са L блокова може се формално дефинисати као:

$$\mathbf{A}^{(j)}, \mathbf{F}^{(j)} = B^{(j)}(\mathbf{A}^{(j-1)}, \mathbf{F}^{(j-1)}) \quad \text{for} \quad j = 1, \ldots, L, \tag{38}$$

где су $F^{(0)} = F$ и $A^{(0)} = A$ почетни атрибути чворова и матрица суседства оригиналног графа, респективно. Важно је напоменути да се излаз једног блока користи као улаз за узастопно следећи блок како је приказано у једначини изнад. Када постоји само један блок (или L=1), GNN архитектура се може сматрати равном (енг. flat) јер директно генерише атрибуте на нивоу графа из оригиналног графа. GNN архитектура са слојевима груписања може се посматрати као хијерархијски процес када је L>1, где се атрибути чворова постепено сажимају како би се формирали атрибути графа тако што се накнадно генеришу све грубљи графови.

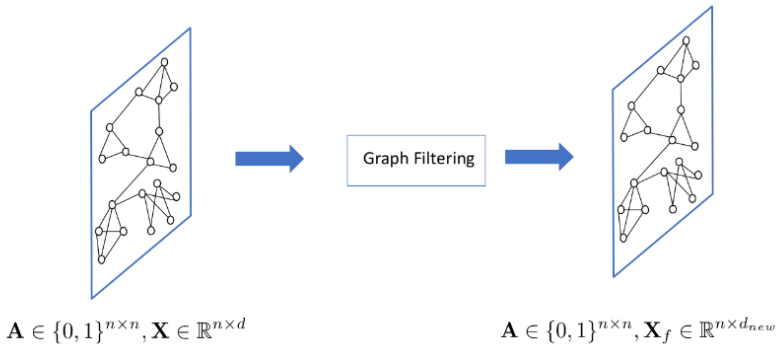


Слика 26. Генерални дизајн GNN модела [14]

GNN се сматра процесом учења репрезентације на графовима. За задатке фокусиране на чворове, GNN циља учење атрибута за сваки чвор, док за задатке фокусиране на граф, циљ је учење репрезентативних атрибута за цео граф. Процес учења атрибута чворова користи улазне атрибуте чворова и структуру графа. Овај процес може се сажети у једначину:

$$F_{of} = h(A, F_{if}), \quad (39)$$

где $A \in R^{N \times N}$ означава матрицу суседства графа са N чворова, а $F_{if} \in R^{N \times dif}$ и $F_{of} \in R^{N \times dof}$ су улазне и излазне матрице атрибута. Операција која узима атрибуте чворова и структуру графа као улаз и генерише нови скуп атрибута чворова назива се **операција филтрирања графа** (graph filtering operation).

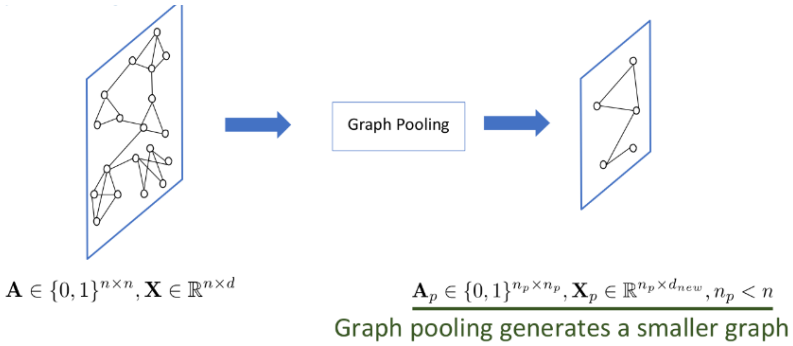


Слика 27. Операција филтрирања графа

За задатке фокусиране на чворове, **операција филтрирања је довољна**, али за задатке фокусиране на граф, потребне су и операције груписања (pooling) да би се генерисали атрибути на нивоу целог графа. Операције груписања на графовима треба да искористе информације о структури графа да би водиле процес груписања. Оне често узимају граф као улаз и производе сажети граф са мање чворова. Кључ операција груписања је генерисање структуре графа (или матрице суседства) и атрибута чворова за сажети граф. Опште, операција груписања графа може се описати једначином:

$$A_{op}, F_{op} = pool(A_{ip}, F_{ip}), \quad (40)$$

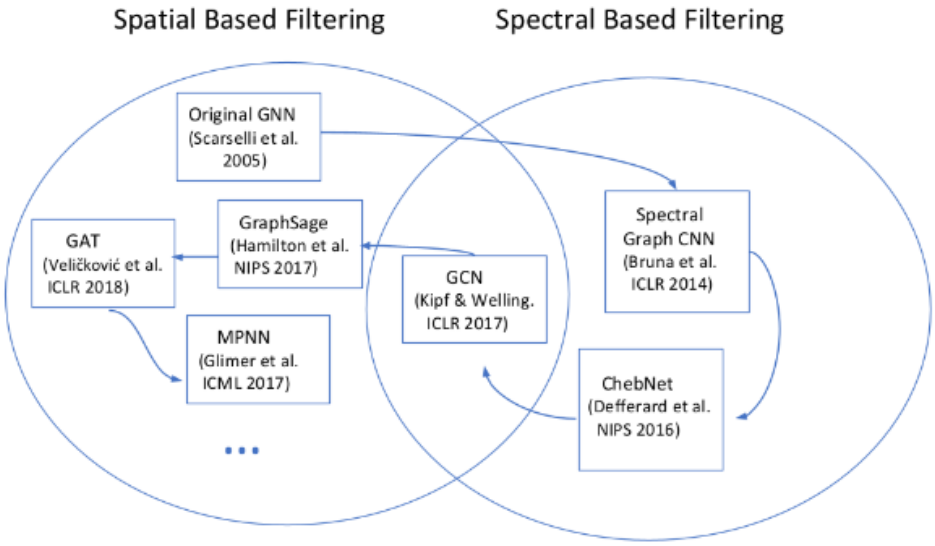
где су A_{ip},Fip и A_{op},F_{op} матрице суседства и атрибута пре и после операције груписања.



Слика 28. Операција груписања графа

Архитектура типичног GNN модела састоји се од операција филтрирања и/или груписања графа. За задатке фокусиране на чворове, GNN користе само операције филтрирања, док за задатке фокусиране на граф, GNN захтевају и операције филтрирања и груписања. Слојеви груписања обично раздвајају слојеве филтрирања у блокове.

Филтрирање графа - Постоје различити приступи за дизајнирање графовских филтера, који се могу грубо поделити на две категорије: (1) **просторно-базирани графички филтери** (енг. spatial-based graph filters) и (2) **спектрално-базирани** графички филтери (енг. spectral-based). Просторно-базирани графички филтери експлицитно користе структуру графа (тј. везе између чворова) да би извршили процес пречишћавања карактеристика у домену графа. За разлику од њих, спектрално-базирани графички филтери користе спектралну теорију графова за дизајнирање операције филтрирања у спектралном домену. Ове две категорије графичких филтера су тесно повезане. Конкретно, неки од спектрално-базираних графичких филтера могу се сматрати просторно-базираним филтерима.



Слика 29. Типови филтрирања графа

Спектрално-базирани графички филтери - Спектрално базирани графички филтери су дизајнирани у спектралном домену сигнала на графу. Идеја спектралног филтрирања графа је модулација фреквенција сигнала на графу тако да се неке

фреквенцијске компоненте задржавају или појачавају, док се друге уклањају или умањују. Дакле, за сигнал на графу $f \in R^N$, прво је потребно применити Фуријеову трансформацију на графу како би се добили Фуријеови коефицијенти графа, а затим модулирати те коефицијенте пре реконструкције сигнала у просторном домену. За сигнал $f \in R^N$ дефинисан на графу G, његова Фуријеова трансформација на графу дефинисана је као: $\hat{f}=U^T f$, где U садржи сопствене векторе Лапласијанове матрице графа G, а $\hat{f}$ су добијени Фуријеови коефицијенти на графу за сигнал f. Ови коефицијенти описују како свака Фуријеова компонента на графу доприноси сигналу f. Специфично, i-ти елемент од $\hat{f}$ одговара i -тој Феријеовој компоненти ui са фреквенцијом λi. λi је сопствена вредност која одговара ui. Да бисмо модулирали фреквенције сигнала f, филтрирамо Фуријеове коефицијенте на следећи начин:

$$\hat{\mathbf{f}}'[i] = \hat{\mathbf{f}}[i] \cdot \gamma(\lambda_i), \text{for} \quad i = 1, \ldots, N, \tag{41}$$

где је $\gamma(\lambda i)$ функција са фреквенцијом λi као улазом, која одређује како треба модулирати одговарајућу фреквенцијску компоненту. Овај процес се може изразити у матричном облику:

$$\hat{\mathbf{f}}' = \gamma(\mathbf{\Lambda}) \cdot \hat{\mathbf{f}} = \gamma(\mathbf{\Lambda}) \cdot \mathbf{U}^\top \mathbf{f}, \tag{42}$$

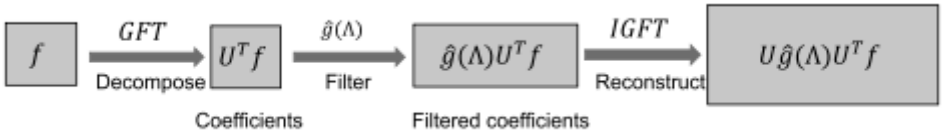
где је Λ дијагонална матрица која се састоји од фреквенција (сопствених вредности Лапласијанове матрице) и $\gamma(\Lambda)$ је примена функције $\gamma()$ на сваки елемент у дијагонали Λ.

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{pmatrix}; \quad \gamma(\mathbf{\Lambda}) = \begin{pmatrix} \gamma(\lambda_1) & & 0 \\ & \ddots & \\ 0 & & \gamma(\lambda_N) \end{pmatrix}.$$

Са филтрираним коефицијентима, сада можемо реконструисати сигнал у домен графа користећи инверзну Фуријеву трансформацију на графу:

$$\mathbf{f}' = \mathbf{U}\hat{\mathbf{f}}' = \mathbf{U} \cdot \gamma(\mathbf{\Lambda}) \cdot \mathbf{U}^\top \mathbf{f}, \tag{43}$$

где је f'' добијени филтрирани сигнал на графу. Процес филтрирања може се сматрати применом оператора $U \cdot \gamma(\Lambda) \cdot U^T$ на улазни сигнал на графу. Ради практичности, функцију $\gamma(\Lambda)$ понекад називамо филтером јер контролише како се свака фреквенцијска компонента сигнала на графу f филтрира.



Слика 30. Процес спектралног филтрирања

Спектралне мреже су редуковале филтер у спектралном домену на дијагоналну матрицу. Као последица тога, могуће је конструисати мрежу која учи конволуцијске филтере за класификацију графова. Недостаци:

- Филтер се примењује на цео граф, тако да не постоји концепт локалитета који имамо у сликама.
- Примена је рачунски неефикасна, посебно за велике графове.

GCN-Filter: Поједностављени *Cheby-Filter* који укључује *1-hop* суседе *Cheby-Filter* укључује *K-hop* окружење чвора приликом израчунавања нових особина за тај чвор. Предложен је поједностављени Cheby-Filter назван GCN-Filter. Поједностављен је од Cheby-Filtera тако што је ред Chebyshev полинома постављен на $K = 1$ и апроксимира се $\lambda\_max \approx 2$. Под овим поједностављењем и апроксимацијом, Cheby-Filter са $K = 1$ може се поједноставити на следећи начин:

$$\begin{aligned}\mathbf{f}' &= \mathbf{U}\gamma(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{f} \\ &= \theta_0\mathbf{U}\mathbf{I}\mathbf{U}^\top\mathbf{f} + \theta_1\mathbf{U}(\mathbf{\Lambda}-\mathbf{I})\mathbf{U}^\top\mathbf{f} \\ &= \theta_0\mathbf{f} - \theta_1(\mathbf{L}-\mathbf{I})\mathbf{f} \\ &= \theta_0\mathbf{f} - \theta_1(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{\frac{1}{2}})\mathbf{f}.\end{aligned} \tag{44}$$

Сходно томе, применом *GCN-Filtera* на графички сигнал *f*, можемо добити излазни сигнал *f '* на следећи начин:

$$\begin{aligned}\mathbf{f}' &= \mathbf{U}\gamma(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{f} \\ &= \theta_0\mathbf{U}\mathbf{I}\mathbf{U}^\top\mathbf{f} + \theta_1\mathbf{U}(\mathbf{\Lambda}-\mathbf{I})\mathbf{U}^\top\mathbf{f} \\ &= \theta_0\mathbf{f} - \theta_1(\mathbf{L}-\mathbf{I})\mathbf{f} \\ &= \theta_0\mathbf{f} - \theta_1(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{\frac{1}{2}})\mathbf{f}.\end{aligned} \tag{45}$$

Треба напоменути да је ова једначина тачна зато што се усваја нормализована Лапласијанова матрица тј. $L = I - D^{-1/2} AD^{-1/2}$. Додатно поједностављење се примењује на ову једначину постављањем $\theta = \theta_0 = -\theta_1$, што доводи до

$$\begin{aligned}\mathbf{f}' &= \theta_0\mathbf{f} - \theta_1(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{\frac{1}{2}})\mathbf{f} \\ &= \theta(\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{f}.\end{aligned} \tag{46}$$

Треба напоменути да матрица $I + D^{-1/2} AD^{-1/2}$ има сопствене вредности у распону *[0, 2]*, што може довести до нумеричке нестабилности када се овај оператор поново примењује на одређени сигнал *f*. Због тога је предложен трик са ренормализацијом за ублажавање овог проблема, који користи $\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$ да замени матрицу $I + D^{-1/2} AD^{-1/2}$ у новој једначини, где је $\tilde{A} = A + I$ и $\tilde{D}_{ii} = \Sigma\tilde{A}_{i,\,j}$. Коначни GCN-Filter након ових поједностављења је дефинисан као

$$\mathbf{f}' = \theta\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{f}. \tag{47}$$

i, j-ти елемент матрице $\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$ је различит од нуле само када су чворови v_i и v_j повезани. За појединачни чвор, овај процес се може сматрати као агрегација информација од његових *1-hop* суседа где се сам чвор такође сматра као свој *1-hop* сусед. Дакле, *GCN-Filter* се такође може сматрати просторно-заснованим (spatial-based) филтером, који укључује само директно повезане суседе приликом ажурирања особина чвора.

GCN су знатно рачунски ефикасније од својих претходника и лакше за кодирање, али имају неколико ограничења:

- Не подржавају директно карактеристике ивица.
- Занемарују концепт порука у графовима. Типично, чворови могу слати поруке (нумеричке векторе) дуж ивица графа.

Просторни приступи (енг**.** spatial-based**)** дефинишу конволуције директно на графу засноване на топологији графа. Обично следе исти образац:

- Вектори карактеристика чворова се трансформишу коришћењем неке врсте пројекције.

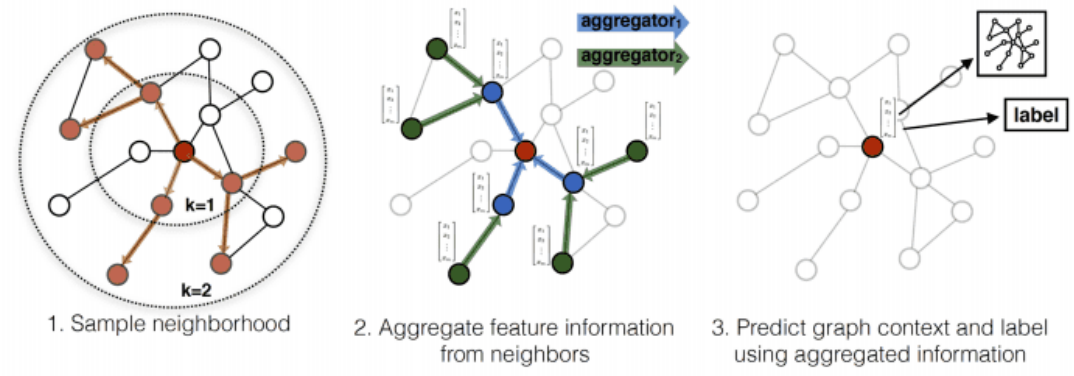
- Агрегирају се пермутационо-инваријантном функцијом.
- Вектор карактеристика сваког чвора се ажурира на основу његових тренутних вредности и агрегиране репрезентације суседства.

GraphSAGE

Модел ***GraphSAGE*** представља просторно заснован филтер који се такође заснива на агрегацији информација из суседних чворова. Процес генерисања нових карактеристика за појединачни чвор v_i може се формулисати на следећи начин:

$$\begin{aligned} \mathcal{N}_S(v_i) &= \text{SAMPLE}(\mathcal{N}(v_i), S) \\ \mathbf{f}'_{\mathcal{N}_S(v_i)} &= \text{AGGREGATE}(\{\mathbf{F}_j, \forall v_j \in \mathcal{N}_S(v_i)\}) \\ \mathbf{F}'_i &= \sigma\left([\mathbf{F}_i, \mathbf{f}'_{\mathcal{N}_S(v_i)}]\Theta\right), \end{aligned} \tag{48}$$

где је *SAMPLE()* функција која узима скуп као улаз и насумично бира S елемената из улаза као излаз, *AGGREGATE()* је функција која комбинује информације из суседних чворова где $f'_{NS(vi)}$ означава излаз функције *AGGREGATE()*, а $[\cdot, \cdot]$ је операција конкатенације. Дакле, за појединачни чвор v_i, филтер у GraphSAGE прво бира S чворова из његових суседних чворова $N(v_i)$ као што је приказано у једначини. Затим, функција *AGGREGATE()* агрегира информације из ових изабраних чворова и генерише карактеристику $f'_{NS(vi)}$ као што је приказано у једначини изнад. На крају, генерисане информације о суседству и старе карактеристике чвора v_i се комбинују да би се генерисале нове карактеристике за чвор v_i како је приказано у једначини.



Слика 31. GraphSage процес

За сваки слој, проширујемо дубину суседства K, што резултира узорковањем карактеристика чвора на удаљености од K скокова. Ово је слично повећању рецептивног поља класичних конволуционих мрежа. Лако је схватити колико је ово рачунски ефикасно у поређењу са коришћењем комплетног суседства. Тиме се завршава процес прослеђивања унапред код *GraphSage*.
Кључни допринос рада *GraphSage* јесу две идеје:

1. Обучавање модела на потпуно ненадгледан (unsupervised) начин. Ово се може урадити коришћењем функције губитка која намеће сличне репрезентације блиским чворовима и различите репрезентације удаљеним чворовима.

2. Такође можемо обучавати на надгледан (supervised) начин користећи етикете и облик крос-ентропије (cross-entropy) за учење репрезентација чворова.

Замршени део је да такође обучавамо функцију агрегације уз наше учиве матрице тежина. Експериментисало са 3 различите функције агрегације: а) **mean агрегатор**, б) **LSTM агрегатор** и ц) **max-pooling** агрегатор. У сва три случаја, функције садрже параметре који се уче током обучавања. На овај начин мрежа ће научити сама себе "правилан" начин агрегације карактеристика са узоркованих (sampled) чворова.

Различите функције *AGGREGATE()* су представљене на следећи начин:

- **Средњи агрегатор** (енг. Mean aggregator). Средњи агрегатор једноставно узима елемент по елемент средњу вредност вектора у $\{F_j, \forall v_j \in N_S(v_i)\}$. Средњи агрегатор овде је веома сличан филтеру у GCN. Када се ради о чвору v_i, оба узимају (тежински) просек суседних чворова као његову нову репрезентацију. Разлика је у томе како се улазна репрезентација F_j чвора v_i укључује у израчунавање. Јасно је да се у *GraphSAGE, Fi* конкатенира са агрегираним информацијама суседа $f'_{NS(vi)}$. Међутим, у GCN-филтеру, чвор v_i се третира на исти начин као и његови суседи и F_i је део процеса тежинског просека.
- ***LSTM* агрегатор**. *LSTM* агрегатор третира скуп узоркованих (енг. sampled) суседних чворова $N_S(v_i)$ чвора v_i као секвенцу и користи *LSTM* архитектуру за обраду секвенце. Излаз последње јединице *LSTM* служи као резултат $f'_{NS(vi)}$. Међутим, не постоји природни редослед међу суседима, због тога је усвојено насумично ређање.
- **Оператор пулинга** (енг. Pooling operator). Оператор пулинга усваја операцију *max pooling* за резимирање информација из суседних чворова. Пре сумирања резултата, улазне карактеристике на сваком чвору се прво трансформишу са слојем неуронске мреже. Процес се може описати на следећи начин:

$$\mathbf{f}'_{N_S(v_i)} = \max(\{\alpha(\mathbf{F}_j\boldsymbol{\Theta}_{\text{pool}}), \forall v_j \in \mathcal{N}_S(v_i)\}), \tag{49}$$

где *max()* означава оператор елемент по елемент максимума, Θ_{pool} означава трансформациону матрицу, а α(), је нелинеарна активациона функција.

GraphSAGE-филтер је просторно локализован јер укључује само суседе на 1-скоку без обзира на коришћени агрегатор. Агрегатор је такође заједнички за све чворове.

GAT

Почињемо са описом једног Graph Attention Layer-а, који је основни слој коришћен у свим GAT (енг. Graph Attention Networks) архитектурама [15]-.

Улаз у овај слој чини сет карактеристика чвора, $h=\{\vec{h}_1,\vec{h}_2,...,\vec{h}_N\}$, где је $h_i \in \mathrm{R}^F$, N је број чворова, а F број карактеристика сваког чвора. Слој производи нови сет карактеристика чвора (потенцијално другачије кардиналности F'), $h'=\{\vec{h}_1',\vec{h}_2',...,\vec{h}_N'\}$, где је $h_i' \in \mathrm{R}^{F'}$, као излаз.

За довољну експресивну моћ у трансформацији улазних карактеристика у виши ниво, потребна је барем једна *learnable* линеарна трансформација. Као први корак, примењује се заједничка линеарна трансформација, параметризована тежинском матрицом $\mathbf{W} \in$

$\mathbf{R}^{F' \times F}$, на сваки чвор. Потом следи ***self-attention*** над чворовима, заједнички механизам пажње $\boldsymbol{a}$: $\mathbf{R}^{F'} \times \mathbf{R}^{F'} \rightarrow \mathbf{R}$ израчунава коефицијенте пажње (енг. *attention coefficients*).

$$e_{ij} = a(\vec{Wh_i}, \vec{Wh_j}), \quad (50)$$

који указују на важност карактеристика чвора j за чвор i. У свом најопштијем облику, модел дозвољава сваком чвору да се фокусира на сваки други чвор, изостављајући све структуралне информације. Уводимо структуру графа у механизам тако што вршимо *masked attention* - израчунавамо e_{ij} само за чворове $j \in N_i$, где је N_i неко суседство (*neighborhood*) чвора i у графу. У овом примеру то ће бити тачно првородни (*first-order*) суседи од i (укључујући i). Да би коефицијенти били лако поредиви међу различитим чворовима, нормализујемо их помоћу softmax функције:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}. \quad (51)$$

У даљим експериментима, механизам пажње a је једнослојна *feedforward* неуронска мрежа, параметризована тежинским вектором $\vec{a} \in \mathrm{R}^{2F'}$, и примењује *LeakyReLU* нелинеарност (са негативним нагибом улаза $\alpha = 0.2$). Коефицијенти израчунати помоћу механизма пажње (Слика 34 (лево)) могу се изразити као:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T [\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)} \quad (52)$$

где T представља транспозицију, а || операцију конкатенације. Једном када се добију, нормализовани коефицијенти пажње користе се за израчунавање линеарне комбинације карактеристика које одговарају истим, како би послужиле као коначне излазне карактеристике за сваки чвор (након потенцијално примењене нелинеарности σ):

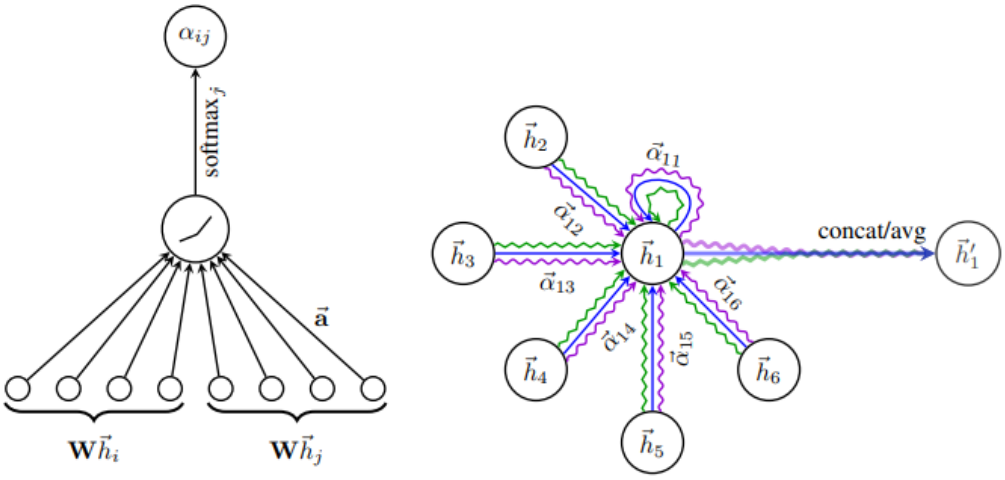
$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\vec{h}_j\right). \quad (53)$$

Да би се стабилизовао процес учења *self-attention*, утврђено је да је коришћење *multi-head attention* корисно за тај проблем. Конкретно, K независних механизама пажње извршава трансформацију приказану изнад, а затим се њихове карактеристике конкатенирају, резултирајући у следећој репрезентацији излазних карактеристика:

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right) \quad (54)$$

где || представља конкатенацију, α^k_{ij}су нормализовани коефицијенти пажње израчунати од стране k-тог механизма пажње (a^k), а W^k је одговарајућа тежинска матрица линеарне трансформације улаза. Напомена: у овом окружењу, коначни враћени излаз, h', ће садржати KF' карактеристика (уместо F') за сваки чвор. Посебно, ако извршимо *multi-head attention* на коначном (предикционом) слоју мреже, конкатенација више није смислена, зато уместо тога користимо *averaging* и одлажемо примену коначне нелинеарности (обично софтмакс или логистички сигмоид за проблеме класификације) до тада:

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right) \quad (55)$$



Слика 32. Механизам пажње(лево) и лустрација multi-head attention(десно)

Слика 32 (лево): Механизам пажње $a(W\vec{h}_i, W\vec{h}_j)$ који користи наш модел, параметризован тежинским вектором $\vec{a} \in \mathrm{R}^{2F'}$, примењује активацију *LeakyReLU*. Слика 32 (десно): Илустрација *multi-head attention* (са K=3 глава) чвора 1 на његовом суседству. Различити стилови и боје стрелица означавају независне израчунавања пажње. Агрегиране карактеристике из сваке главе се конкатенирају или усредњавају (*averaging*) да би се добило $\vec{h}'_1$.

graph attention слој, описан у претходном пододељку, директно решава неколико проблема присутних у претходним приступима моделирања података структурираних у облику графа са неуронским мрежама:

1. **Рачунарска Ефикасност:** Операција само-пажње може се паралелизовати преко свих ивица, а израчунавање излазних особина преко свих чворова. Не захтева скупу декомпозицију нити сличне матричне операције. Сложеност времена једног GAT *attention* чвора за израчунавање F0 особина износи O(|V|FF' + |E|F'), где је F број улазних особина, а |V| и |E| су бројеви чворова и ивица у графу, респективно. Ова сложеност је упоредива са базним методама као што су Графовске Конволуцијске Мреже (GCNs). Примена *multi-head* пажње множи захтеве за складиштењем и параметрима за фактор K, док су рачунања појединачних глава потпуно независна и могу се паралелизовати.
2. **Капацитет Модела и Интерпретативност:** За разлику од GCNs, овај модел омогућава (имплицитно) додељивање различитих важности чворовима истог суседства, чиме се повећава капацитет модела. Даље, анализирање научених *attention* тежина може довести до предности у интерпретативности.
3. **Механизам Пажње и Флексибилност:** Механизам пажње се примењује заједнички на све ивице у графу, те стога не зависи од претходног приступа глобалној структури графа или особинама свих његових чворова (ограничење многих претходних техника). То има неколико пожељних импликација:
    - Граф не мора бити неусмерен (можемо једноставно изоставити рачунање α_{ij} ако ивица j → i није присутна).
    - Чини ову технику директно применљивом на индуктивно учење, укључујући задатке где се модел процењује на графовима који су потпуно непознати током тренинга.
4. **Предности у односу на друге методе:** Недавно објављена индуктивна метода Hamilton и др. (2017) узоркује фиксну величину суседства сваког чвора, да би

одржала конзистентан рачунарски отисак; то јој не дозвољава приступ целокупном суседству током закључивања. Штавише, ова техника постигла је неке од својих најјачих резултата када се користи агрегатор суседства заснован на LSTM. То претпоставља постојање конзистентног секвенцијалног редоследа чворова у суседствима, а аутори су то исправили конзистентним храњењем насумично уређених секвенци у LSTM. Наша техника не пати од било којег од ових проблема - ради са целокупним суседством (уз трошак променљивог рачунарског отиска, који је и даље упоредив са методама попут GCN) и не претпоставља никакав редослед унутар њега.

Предности:

- Омогућавају моделу да се фокусира на специфичне чворове и ивице графа.
- Може се користити да побољша перформансе на задацима везаним за класификацију чворова, графова, ивица…

Мане:

- Може бити рачунски комплексан задатак
- Механизам пажње може бити тежак за имплементирање
- Осетљив на избор хиперпараметара као што је број глава или број скривених слојева
- Може дати лоше резултате за графове са малим бројем чворова и ивица
- Ограничене могућности груписања података: Због техничких ограничења, постоји ограничење у обради више графова одједном.
- Ограничена дубина анализе: Величина дела графа који модел може анализирати ограничена је дубином мреже.
- Проблеми са паралелизацијом: Паралелно обрађивање великог броја ивица може довести до понављања израчунавања због преклапања у графовима.
- Ограничена ефикасност GPU: У одређеним сценаријима, GPU не нуди значајне предности у брзини обраде у односу на CPU.

Transductive

Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	81.7 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	**79.0** ± 0.3%
GAT (ours)	**83.0** ± 0.7%	**72.5** ± 0.7%	**79.0** ± 0.3%

Inductive

Method	PPI
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	**0.973** ± 0.002

Слика 33. Поређење GAT приступа са осталим приступима на специфичним задацима

Резултати упоредних експеримената су сажети на слици 33. За трансдукцијске задатке, извештава се о просечној тачности класификације на тест чворовима након 100 покушаја. Пореди се са већ објављеним резултатима за *state-of-the-art* технике. За Chebyshev филтере, наводе се најбоље перформансе за K = 2 и K = 3. Такође, тестира се GCN модел са 64 скривених карактеристика, користећи ReLU и ELU активацију, и извештава се о бољем резултату (означен као GCN-64∗).
За индуктивни задатак, извештава се о *micro-averaged* F1 скору на чворовима два тест графа, узимајући у обзир 10 покушаја. Упоређује се са објављеним резултатима за друге

технике. Овај приступ је супервизован, упоређује се са супервизованим GraphSAGE приступима. Такође, наводи се најбољи резултат са GraphSAGE, који је постигнут мењајући архитектуру (означен као GraphSAGE∗). Извештава се о резултатима GAT модела са константном пажњом (означен као Const-GAT).
Резултати показују врхунске перформансе на сва четири скупа података. На Cora и Citeseer, GAT побољшава GCN за 1.5% и 1.6%. На PPI скупу, GAT побољшава резултате за 20.5%, доказујући потенцијал за индуктивне поставке. Упоређујући са Const-GAT, GAT показује боље резултате за 3.9%, истичући значај различитих тежина за различите суседе.

Пулинг графа (енг. Graph Pooling)

Филтери за графове усавршавају карактеристике чворова без мењања структуре графа [16]. Након операције филтрирања графа, сваки чвор у графу добија нову репрезентацију карактеристика. Обично су операције филтрирања графа довољне за задатке фокусиране на чворове који искоришћавају репрезентације чворова. Међутим, за задатке фокусиране на графове, потребна је репрезентација целог графа. Да би се добила таква репрезентација, потребно је сажети информације из чворова. Постоје две главне врсте информација које су важне за генерисање репрезентације графа: карактеристике чворова и структура графа. Очекује се да репрезентација графа очува и информације о карактеристикама чворова и информације о структури графа. Слично класичним CNN, предложени су слојеви за пулинг графова како би се генерисале репрезентације на нивоу графа.

Будући да GNN уче векторске репрезентације чворова, за њихову употребу у класификацији графова потребан је пулинг слој који омогућава прелазак са нивоа чвора на ниво графа. Формално, пулинг слој је параметризована функција која пресликава мултискуп вектора, односно научене репрезентације на нивоу чвора, у један вектор, односно репрезентацију на нивоу графа.
Најједноставнији од таквих слојева су збирни (sum), просечни (avg) и минимални (min) или максимални (max) *pooling* слојеви. На пример, дат је граф G и мултискуп
$M = \{\{f(v) \in R^d \mid v \in V\}\}$
репрезентација чворова у графу G, збирни пулинг рачуна

$$f_{\text{pool}}(\mathcal{G}) = \sum_{\mathbf{f}(v) \in M} \mathbf{f}(v), \tag{56}$$

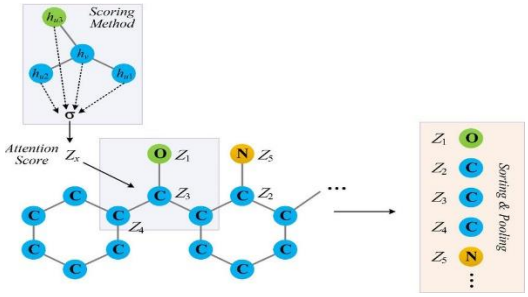
док просечни, минимални, максимални пулинг узимају (по компонентама) просек, минимум, максимум над елементима у M, редом. Ови једноставни пулинг слојеви се и даље користе у многим архитектурама GNN. У ствари, скорији радови показали су да сложенији слојеви, који се ослањају на кластеровање, не нуде емпиријске предности на многим стварним скуповима података, посебно онима из молекуларног домена.

Пулинг слојеви засновани на пажњи (Attention-based pooling layers) -
Једноставно пулинг заснован на пажњи постао је популаран последњих година због своје једноставне имплементације и скалабилности у поређењу са сложенијим алтернативама. За пулинг у GNN, представљен је *SAGPool* слој, скраћено за методу за самопажњу у пулингу графова за GNN, који користи самопажњу (self-attention). Специфично,

израчунат је скор самопажње множењем агрегираних особина произвољног слоја GNN матрицом Θ_{att} у $R^{d\times 1}$, где d означава број компоненти особина чвора. На пример, израчунавање скора самопажње Z(v):

$$\mathbf{Z}(v) = \sigma\left(\mathbf{f}^{t-1}(v)\cdot W_1 + \sum_{w\in N(v)} \mathbf{f}^{t-1}(w)\cdot W_2\right)\cdot \Theta_{\text{att}}. \quad (57)$$

Скор самопажње Z(v) се накнадно користи за избор top-k чворова у графу, изостављајући остале чворове, ефективно прорећујући чворове из графа.



Слика 34. Процес self-attention граф пулинга

Проблеми и области примене GNN

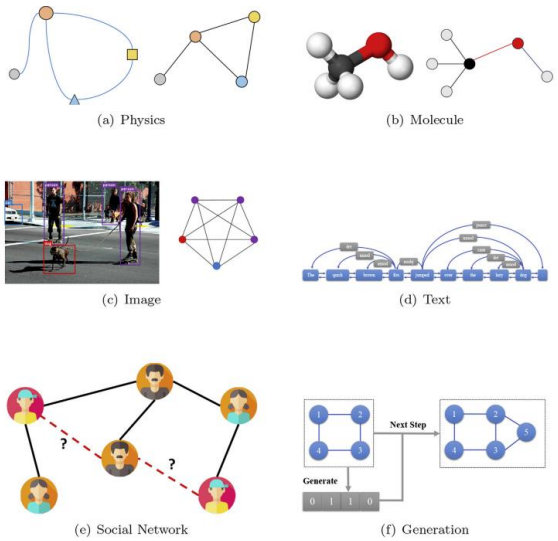
Проблеми које GNN решавају могу се класификовати у следеће категорије:

1. **Класификација Чворова** (eng. Node Classification): Циљ је идентификовати означавање узорака (приказаних као чворови) прегледом ознака њихових суседа. Ова врста проблема се обично учи полу-надзирано (semi-supervised), са само делом графа означеним.
2. **Класификација Графова** (eng. Graph Classification): Задатак је сортирати цео граф у одвојене групе. Фокус се помера на домен графа сличан категоризацији слика. Класификација графова има више примена, укључујући откривање да ли је протеин ензим у биоинформатици, класификацију чланака у обради природног језика и анализи социјалних мрежа.
3. **Визуализација Графова** (eng. Graph Visualization): На пресеку геометријске теорије графова и визуелизације информација, то је област математике и рачунарства. Фокусирана је на визуелни приказ графова који помаже кориснику да разуме графове откривајући структуре и аномалије које могу бити присутне у подацима.
4. **Предвиђање Веза** (eng. Link Prediction): Овде, алгоритам мора да разуме како се ентитети међусобно повезују у графовима и покушава да предвиди да ли ће два ентитета бити повезана. Закључивање социјалних односа или препоручивање потенцијалних пријатеља корисницима је кључно у социјалним мрежама. Такође је примењен на проблеме са системима за препоруке и идентификацији криминалних веза.
5. **Кластеризација Графова** (eng. Graph Clustering): Односи се на кластеризацију података засновану на графу. Кластеризација се на подацима графа врши на два различита начина. Користећи тежине ивица или удаљености ивица, кластеризација чворова покушава да групише чворове графа у густо повезане

кластере. Други начин кластеризације графова то чини тако што третира графове као предмете које треба груписати и групише их на основу сличности.

Области примене графовских неуронских мрежа

1. **Жичане/бежичне комуникационе мреже** (енг. Wired/Wireless Communication Networks)
   - Опис: GNN се користе за оптимизацију протока података и управљање мрежама.
   - Пример: Побољшање квалитета услуге у мобилним мрежама.
2. **Интернет ствари (IoT)** (енг. IoT)
   - Опис: GNN анализирају мреже уређаја за аутоматизацију и унапређење IoT система.
   - Пример: Управљање енергијом у паметним зградама.
3. **Биоинформатика** (енг. Bioinformatics)
   - Опис: Анализа биолошких мрежа и структура на молекуларном нивоу.
   - Пример: Истраживање интеракција међу протеинима.
4. **Физика** (енг. Physics)
   - Опис: Моделирање и анализа физичких система помоћу GNN.
   - Пример: Симулације сложених физичких процеса.
5. **Обрада природног језика** (енг. Natural Language Processing)
   - Опис: Разумевање и генерисање језичких структура помоћу GNN.
   - Пример: Аутоматско сумирање текстова или превођење.
6. **Компјутерски вид** (енг. Computer Vision)
   - Опис: Интерпретација слика и видео садржаја користећи GNN.
   - Пример: Анализа социјалних интеракција на видео снимцима.
7. **Графови знања**(енг. Knowledge Graphs)
   - Опис: Организација и анализа великих скупова података и знања.
   - Пример: Семантичко претраживање и везивање информација у великим базама података.
8. **Системи за препоруке**(енг. Recommendation Systems)
   - Опис: Генерисање персонализованих препорука на основу мреже корисника и производа.
   - Пример: Предлагање филмова на стриминг сервисима.

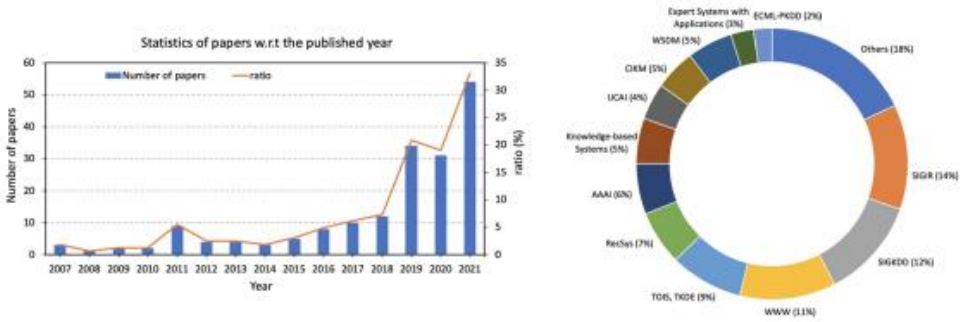


Слика 35. Области примене GNN

ГРАФОВСКЕ НЕУРОНСКЕ МРЕЖЕ И СИСТЕМИ ЗА ПРЕПОРУКЕ

Напредак у приступима базираним на графовима показао је њихову ефикасност у моделирању корисничких преференција и карактеристика предмета за системе за препоруке. Већина података у системима за препоруке се може организовати у графове где су различити објекти (нпр. корисници, предмети и атрибути) експлицитно или имплицитно повезани и утичу једни на друге путем различитих веза. Таква организација заснована на графовима доноси користи искоришћавању потенцијалних својстава у техникама учења графова (нпр. случајне шетње и умрежавање графова) како би се обогатиле репрезентације чворова корисника и предмета, што је суштински фактор за успешне препоруке.

У протеклој деценији, системи за препоруке су се брзо развијали од традиционалних факторизационих приступа до напредних модела заснованих на дубоким неуронским мрежама. Посебно, GNN базирани системи за препоруке су постигли *state-of-the-art* статус у многим аспектима, укључујући различите фазе препоруке, сценарије, циљеве и апликације.



Слика 36. Статистика објављених радова на тему примене GNN мрежа у системима за препоруке

Успех GNN базираних система за препоруке може се објаснити из следећа три угла:

- **Структурални подаци** (енг. Structural data) - Подаци прикупљени са онлајн платформи долазе у разним облицима, укључујући интеракцију корисник-предмет (оцена, клик, куповина итд.), профил корисника (пол, године, приходи итд.), атрибуте предмета (бренд, категорија, цена итд.), итд. Традиционални системи за препоруке нису у стању да искористе те вишеструке облике података, и обично се фокусирају на један или неколико одређених извора података, што доводи до субоптималних перформанси пошто се игнорише пуно информација. Изражавањем свих података као чворова и ивица на графу, GNN пружа јединствен начин да се искористе доступни подаци. У међувремену, GNN показује снажну моћ у учењу репрезентација, и тако се могу добити висококвалитетни ембединзи за кориснике, предмете и остале особине, што је од круцијалног значаја за перформансе препорука
- **Повезаност вишег реда** (енг. High-order connectivity) - Тачност препорука зависи од установљивања сличности између корисника и предмета, а таква сличност треба да буде одражена у простору научених ембединга. Конкретно, научено уграђивање за корисника је слично уграђивањима предмета са којима је корисник интераговао. Штавише, ти предмети са којима су интераговали други корисници

са сличним преференцијама такође су релевантни за корисника, што је познато као ефекат колаборативног филтрирања, и то је од велике важности за тачност препорука. У традиционалним приступима, ефекат колаборативног филтрирања се само имплицитно хвата пошто су подаци о обуци углавном записи интеракција које садрже само директно повезане предмете. Другим речима, у обзир се узима само повезаност првог реда. Одсуство повезаности вишег реда може у великој мери оштетити перформансе препорука. Насупрот томе, модели базирани на GNN могу ефикасно установити повезаност високог нивоа. Конкретно, ефекат колаборативног филтрирања се може природно изразити као *multi-hop* суседност на графу, и он се укључује у научене репрезентације кроз пропагацију и агрегацију ембединга.

- **Сигнал супервизије** (енг. Supervision signal) - Сигнали надзора су обично ретки у прикупљеним подацима, док модел базиран на GNN може искористити полу-надзиране сигнале у процесу учења репрезентација да би ублажио овај проблем. Узмимо за пример платформу *E-Commerce,* циљно понашање, куповина, је прилично ретко у поређењу са другим понашањима. Стога, системи за препоруке који користе само циљно понашање могу имати лоше перформансе. Модели базирани на GNN могу ефикасно укључити вишеструка нециљна понашања, као што су претрага и додавање у корпу, кодирајући полу-надзиране сигнале преко графа, што може значајно побољшати перформансе препорука. У међувремену, сигнали само-надзора се такође могу искористити дизајнирањем помоћних задатака на графу, што даље побољшава перформансе препорука.

Дефиниција проблема

За дати извор података X, који обично обухвата скуп корисника U и скуп ставки I, за сваког корисника $u \in U$, проблем препоруке уопштено се може видети као функција мапирања $Y=\text{argmax}_u f(U,I)$, која генерише одговарајуће резултате препорука из I који су интересантни за корисника u. Међутим, не постоји формална дефиниција GLRS-а због различитих имплементација разних модела на различитим скуповима података са специфичним карактеристикама. Графови се могу изградити на основу улазних извора података, као што су интеракције корисника и ставки, као и друге помоћне информације. На пример, узимајући у обзир граф $G=(V, E)$, где чворови у V могу представљати кориснике, ставке и друге именоване ентитете, док ивице у E могу представљати куповине, кликове, социјалне односе и друге везе међу ентитетима. Формулишемо проблем из опште перспективе. Конкретно, за улазни извор података X, желимо да пронађемо мапирање $M(X)\rightarrow G$, које се користи као улаз за генерисање одговарајућих резултата препорука Y моделовањем својстава графа као главног начина, допуњеног другим помоћним карактеристикама графова:

$$Y = argmax_u f(\mathcal{M}(\mathcal{X}) \rightarrow \mathcal{G}|\Theta) \tag{58}$$

где G може бити различитих типова, на пример, хомогени, k-партитни, комплексни хетерогени граф итд..., у зависности од специфичних сценарија препоруке, док Y може бити различитих облика, на пример, оцене, могуће везе, класификације или рангиране листе. θ је скуп параметара модела који треба оптимизовати током обуке модела. [17]

Изазови примене GNN-ова на системе за препоруке

Постоје четири критична изазова.

- Како конструисати одговарајуће графове за специфичне задатке?
- Како дизајнирати механизам пропагације и агрегације информација?
- Како оптимизовати модел?
- Како обезбедити ефикасност тренирања и извођења модела?

Конструкција графа

Очигледно, први корак у примени графичких неуронских мрежа је конструисање графова. То је у двоструком смислу: конструисање улазних података као структурираних података графа; реорганизација циља препоруке као задатка на графу. Узимајући задатак стандардног колаборативног филтрирања као пример, улазни подаци су посматрани подаци о интеракцијама корисника и предмета, а излаз су предвиђања недостајућих интеракција корисника и предмета. Стога, може се конструисати бипартитни граф са корисницима/предметима као чворовима и интеракцијама као ивицама. Осим тога, задатак КФ се претвара у предвиђање веза корисник-предмет на графу. Међутим, изазов је конструисати графове који могу добро да се носе са задатком. То треба пажљиво спровести с обзиром на следеће аспекте.

- **Чворови** (енг. Nodes). Један од главних циљева учења са графичким неуронским мрежама је додељивање репрезентација чворовима. То за последицу има да дефиниција чворова у великој мери одређује размеру модела GNN, од којих већину параметара заузимају ембединзи чворова на нивоу-0. Важно је напоменути да се ембединзи ивица обично или не узимају у обзир или се рачунају на основу уграђивања чворова. С друге стране, изазован је и проблем одређивања да ли разликовати различите врсте чворова. На пример, у задатку колаборативног филтрирања, чворови корисника и предмета могу се моделовати различито или сматрати истом врстом чворова. Још једна изазовна тачка је обрада конкретних улазних података као што су нумеричке карактеристике попут цена предмета, које су увек континуиране бројке. Да би се ове карактеристике представиле у графу, једно од могућих решења је да се дискретизују у категоријске, које се онда могу представити као чворови.
- **Ивице** (енг. Edges). Дефиниција ивица веома утиче на квалитет графа у даљој пропагацији и агрегацији, заједно са оптимизацијом модела. У неким тривијалним задацима, улазни подаци система за препоруке могу се сматрати врстом релационих података, као што су интеракције корисник-предмет или друштвене везе корисник-корисник. У неким сложеним задацима, други односи такође могу бити представљени као ивице. На пример, у препоруци пакета, пакет се састоји од неколико предмета. Тада ивица која повезује пакет и предмет може одражавати однос припадности. При конструисању графа, добри дизајни ивица треба да размотре густину графа. Превише густ граф значи да постоје чворови са изузетно високим степенима. То ће учинити да се пропагација ембединга спроводи над огромним бројем суседа. То ће даље учинити пропагирано уграђивање неразликујућим и бескорисним. Да би се обрадиле превише густе ивице, узорковање, филтрирање или обрезивање на графовима су обећавајућа решења. Превише редак граф такође ће резултирати лошом употребом пропагације уграђивања јер ће се пропагација спроводити само на малом делу чворова.

Дизајн мреже

Слој пропагације чини GNN другачијим од традиционалних метода учења графа. Што се тиче пропагације, критично је како изабрати пут за моделирање повезаности високог реда у системима препорука. Осим тога, пропагација такође може бити параметарска, која додељује различите тежине различитим чворовима. На пример, пропагација уграђивања предмета на чвор корисника у графу интеракција корисник-предмет осликава ефекат КФ базиран на предметима. Тежине се односе на различиту важност историјски интерактивних предмета.

У пропагацији, постоје и различити избори функција агрегације, укључујући међусобно сабирање, LSTM, максимално, минимално итд. Пошто не постоји јединствен избор који може најбоље да се изведе међу свим задацима препоруке или различитим скуповима података, витално је дизајнирати специфичан и одговарајући. Осим тога, различити избори пропагације/агрегације доста утичу на ефикасност израчунавања. На пример, међусобно сабирање се широко користи у моделима препорука заснованим на GNN, јер се може ефикасно израчунати, посебно за граф који садржи чворове високог степена, као што су врло популарни предмети (који могу повезати многе кориснике). Такође, слојеви пропагације/агрегације могу се нагомилавати да би помогли чворовима да приступе суседима већих скокова. Превише плитки слојеви чине да се структура графа високог реда не може добро моделовати, а превише дубоки чине да уграђивање чвора буде превише изглачано. Било који од ова два случаја довешће до лошег учинка препоруке.

Оптимизација модела

Да би се оптимизовали модели препорука базирани на графичким неуронским мрежама, традиционалне функције губитка у системима препорука увек се окрећу губицима у учењу графа. На пример, *logloss* у оптимизацији може се сматрати *point-wise* губитком за предвиђање. Слично томе, *BPR* губитак се обично усваја у задатку предвиђања везе на графовима. Још један аспект је узорковање (*sampling*) података. У препоруци заснованој на GNN, за узорковање позитивних или негативних предмета, начин узорковања може у великој мери зависити од структуре графа. На пример, у друштвеној препоруци, извођење случајне шетње по графу може генерисати слабо позитивне предмете (као што су предмети са којима су интераговали пријатељи).
Осим тога, понекад препорука заснована на GNN може укључивати више задатака, као што су задаци предвиђања веза на различитим врстама ивица. Тада у таквом случају, како уравнотежити сваки задатак и учинити да се међусобно појачавају представља изазов.

Ефикасност Рачунања

У стварном свету, системи за препоруке би требали да се тренирају/изводе ефикасно. Зато, да би се осигурала применљива вредност модела препорука базираних на GNN, њихова ефикасност рачунања треба се озбиљно размотрити. Комплексне матричне операције укључене су у сваки слој *GCN*, посебно за спектралне моделе GNN као што је *GCN*. Са вишеслојним гомилањем *GCN* слојева, трошкови рачунања се даље повећавају. Због тога, просторни модели GNN као што је *PinSage* могу бити лакши за имплементацију у великим индустријским апликацијама. Уз *sampling* међу суседима или

pruned структуру графа, ефикасност увек може бити очувана све док можемо поднети пад у перформансама препоруке.

Предобрада података за GNN-базиране системе препорука

Предобрада података је један од круцијалних корака у изградњи било ког модела машинског учења, укључујући GNN-базиране системе препорука. У овом одељку биће дискутовани различити кораци предобраде података укључени у изградњу GNN-базираног система препорука.

1. **Чишћење и трансформација података** (енг. Data Cleaning and Transformation): Први корак у предобради података је да се очисте и трансформишу сирови подаци у формат који је погодан за GNN-базиране системе препорука. Ово подразумева уклањање недостајућих вредности, трансформацију категоричких променљивих у нумеричке променљиве, и нормализацију нумеричких променљивих.
2. **Конструкција графа** (енг. Graph Construction): GNN-базирани системи препорука раде на граф-структурираним подацима, где су корисници и производи представљени као чворови у графу, а њихове интеракције као ивице. Други корак у предобради података је конструкција графа из очишћених података. Граф се може конструисати користећи различите приступе, као што су бипартитни графови корисник-производ или графови ко-појављивања корисник-корисник и производ-производ.
3. **Инжењеринг карактеристика чворова и ивица** (енг. Node and Edge Feature Engineering): Следећи корак је инжењеринг карактеристика за чворове и ивице у графу. Ово подразумева дефинисање карактеристика за чворове и ивице на основу доступних података, као што су демографске карактеристике корисника, атрибути производа и интеракције између корисника и производа. Карактеристике се могу користити за учење ембединга чворова и ивица користећи GNN-ове.
4. **Раздвајање скупа за обуку, валидацију и тестирање** (енг. Train-Validation-Test Split): Следећи корак је раздвајање графа на скупове за обуку, валидацију и тестирање. Скуп за обуку се користи за учење модела GNN, скуп за валидацију за подешавање хиперпараметара, а скуп за тестирање за процену перформанси модела.
5. **Негативно узорковање** (енг. Negative Sampling): У системима препорука, број негативних интеракција (тј. интеракција између корисника и производа које се нису догодиле) је много већи од броја позитивних интеракција. Негативно узорковање је техника која се користи за решавање неуравнотежености класа тако што се случајно узоркују негативне интеракције за обуку модела.
6. **Аугментација података** (енг. Data Augmentation): Аугментација података је техника која се користи за повећање величине скупа података за обуку генерисањем синтетичких примера. У системима препорука, аугментација података се може користити за симулирање различитих интеракција корисника, као што је додавање шума постојећим интеракцијама или стварање нових интеракција између сличних производа.

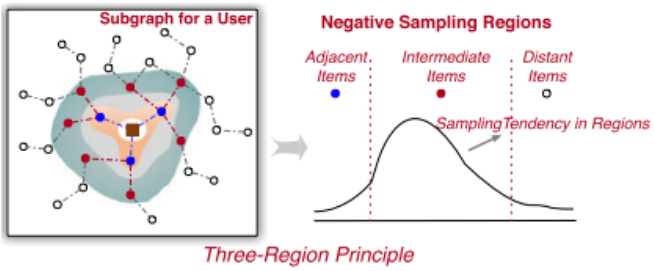
Предобрада података је кључан корак у изградњи GNN-базираних система препорука. Подразумева чишћење и трансформацију сирових података, конструкцију графа, инжењеринг карактеристика за чворове и ивице, раздвајање графа на скупове за

обуку, валидацију и тестирање, извршење негативног узорковања, и коришћење аугментације података за повећање величине скупа података за обуку.

Негативно узорковање (енг. Negative Sampling)

Истраживања из протеклих година подстакла су еволуцију система за препоруке, који су се развили од колаборативног филтрирања до графовских препорука. Кључна тачка је учење квалитетних ембединга и процена вероватноће интеракције корисника и предмета, што је широко примењено у онлајн куповини, друштвеним мрежама и оглашавању. Брзи развој GNN технологије фундаментална је покретачка снага успона графовских препорука, указујући на потенцијал да постане кључна технологија следеће генерације препорука, посебно за веб апликације великих размера.
Међутим, постоји кључни изазов у томе што су у корисник-предмет графовима посматрани само позитивни парови, док други предмети нису повезани с корисницима и могу се сматрати неопаженим везама. Штавише, број глобално неопажених веза је обично огроман и рачунање свих неопажених парова је неизводљиво. Негативно узорковање је витална техника за решавање овог питања. Негативно узорковање је широко прихваћено, стратегија узорковања укључује само избор малог дела негативних веза из региона глобално неопажених веза и обучавање модела да раздвоји те негативне везе од позитивних. Стратегија негативног узорковања убрзава процес тренинга и смањује рачунарску сложеност, омогућавајући препоруке засноване на великим графовима. Осим тога, резултати неколико студија показују да квалитет негативних веза утиче на квалитете ембединга корисника/предмета и ефикасност задатка препоруке. Уобичајено, класична стратегија је коришћење униформне дистрибуције за негативно узорковање. Да би се побољшао квалитет негативних веза, студије су покушале да осмисле нове дистрибуције негативног узорковања како би узорковале тзв. *hard* негативне везе на основу текућег модела у којима би модел разликовао разлике између позитивних и негативних парова с већом финоћом.

На пример, резултати експеримената са *LightGCN* (енг. Light Graph Convolutional Network) показују да перформансе почињу да опадају након што достигну врхунац на другом слоју, када се број слојева повећава. Ово указује на корисност укључивања првог и другог реда суседа у ембединге, али такође и на проблеме прекомерног изглађивања при коришћењу суседа вишег реда. Ови проблеми неизбежни су са повећањем броја слојева мреже, али се могу ублажити дизајнирањем ефикаснијег метода негативног узорковања (енг. negative sampling). Пропагација кроз суседе већег опсега доводи до погоршања перформанси, што сугерише да је таква информација штетна за перформансе препорука. Стога, негативне везе треба узорковати из специфичне регије, а не из глобално неопажене регије. Предложен је Трорегионални Принцип (енг. **The Three-Region Principle**)за узорковање негативних веза из међурегије, пружајући принцип за одређивање кандидатске негативне регије. Такође представљен је метод негативног узорковања RecNS (енг. **Recommendation Negative Sampling**) за дизајнирање дистрибуције узорковања негативних веза из кандидатске регије. Опсежни експериментални резултати показују супериорност RecNS-а у односу на постојеће стратегије негативног узорковања, с побољшањима као што је повећање од 10.47% за PinSage (енг. Pinterest Sage), 6.02% за NGCF (енг. Neural Graph Collaborative Filtering) и 8.20% за LightGCN у погледу Recall@20 на Alibaba (енг. Alibaba) скупу података.



Слика 37. Принцип три регије [18]

Усмерени на особину механизма ширења GNN-а у задатку препоруке засноване на графу, предложен је принцип поделе на три региона за негативно узорковање везан за било ког корисника *u*, где су ставке подељене у три категорије:

1. Суседни регион. Ставке у суседном региону се обично користе за ширење информација о карактеристикама централног корисника *u* у препорукама заснованим на графу. Оне представљају позитивне преференције корисника и не би требало да буду узорковане као негативне.
2. Интермедијални регион. Ставке у интермедијалном региону сматрају се тешким (hard) ставкама сличним позитивним које могу донети више информација за обуку модела. Оне су умерено удаљене од корисника и могу утицати на перформансе препорука. Њихово узорковање као негативних може допринети квалитету узорковања.
3. Удаљени регион. Ставке из удаљеног региона су значајно одвојене од интересовања корисника и често се не узимају у обзир за интеракцију, зато њихово претерано негативно узорковање није продуктивно.

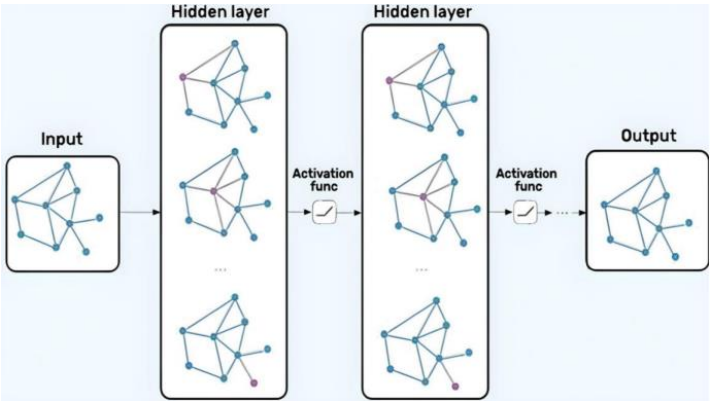
После утврђивања овог принципа, потребно је установити критеријум за раздвајање ових региона. Користимо претраживање по ширини у слојевима (LBFS) за одређивање персонализованих скупова ставки. Ставке које се налазе у k-hop суседству корисника u сматрамо суседним ставкама, а оне изван k-хоп суседства су удаљене ставке. Ставке које припадају k-хоп суседству корисника у сматрају се интермедијалним. K-хоп суседство K дефинисано је као K = A

Структура GNN за систем препорука

Структура графичке неуронске мреже (GNN) за систем препорука може варирати у зависности од специфичних захтева апликације. Међутим, типична структура GNN-а за системе препорука може се поделити на следеће компоненте:

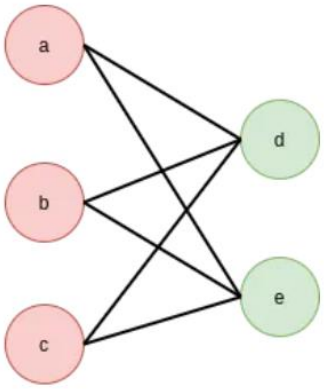
1. **Улазни слој**: Улазни слој GNN-а узима као улаз карактеристике чворова и за кориснике и за производе. Ове карактеристике могу укључивати категоријске карактеристике попут категорије производа, бренда и локације корисника, као и нумеричке карактеристике попут цене производа и старости корисника.
2. **Конволуциони слојеви**: Конволуциони слојеви GNN-а врше пренос порука између чворова у графу. Порука која се преноси између чворова може бити функција карактеристика оба чвора, као и њиховог односа у графу. У случају система препорука, конволуциони слојеви могу ухватити сличност између корисника и производа на основу њихових претходних интеракција и карактеристика.

3. **Пулинг слојеви**: Пулинг слојеви GNN-а агрегирају информације из конволуционих слојева широм графа. Ово се може урадити рачунањем просека или максимума карактеристика чворова у околини чворова. Пулинг слојеви се могу користити и за извршење екстракције карактеристика на нивоу графа.
4. **Излазни слој**: Излазни слој GNN-а предвиђа вероватноћу интеракције корисника са одређеним производом. Ово се може урадити рачунањем резултата сличности између корисника и производа на основу њихових научених репрезентација. Излазни слој се такође може користити за генерисање рангиране листе препоручених производа за датог корисника.



Слика 38. Генерална структура GNN

Уопштено говорећи, структура GNN-а за системе препорука подразумева учење репрезентација корисника и производа на основу њихових интеракција и карактеристика у бипартитном графу (слика 39). Ово омогућава генерисање персонализованих препорука за кориснике на основу њиховог прошлог понашања и преференција.



Слика 39. Бипартитни граф [19]

Функције губитака

Веома битан параметар током тренинга неуронске мреже свакако јесте функција губитка, од ње нам зависи како ће наш модел учити и у ком смеру ће ићи учење, па је због тога избор ове функције круцијалан корак ка изградњи квалитетног система за

препоруке. Три функције које ће бити примењене у раду су BCE (Binary Cross Entropy Loss), Crossentropy Loss и BPR (Bayesian Personalized Ranking Loss).

- BCE се примењује у сценаријима бинарне класификације где постоје само две класе. Ова функција је једноставна за разумевање и имплементацију, и идеална је за моделе који требају да одлуче између две могућности, као што су да/не или позитивно/негативно одлуке. Она је нарочито ефективна када су класе добро балансиране. Међутим, када се суочава са вишекласним задацима, BCE губи своју ефикасност због ограничења на две класе.

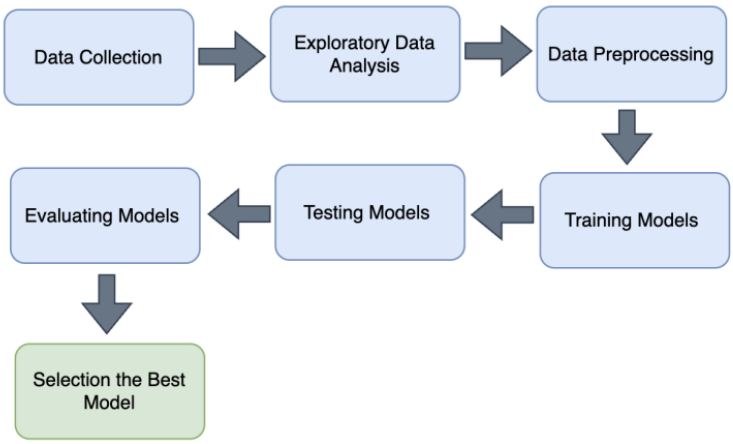
- С друге стране, Crossentropy loss нуди више сложености и адаптивности. Она је дизајнирана за рад са вишекласним проблемима, где може постојати више од две класе. Ова функција је важна у областима где је потребно финије разликовање између различитих класа, као што су сценарији у којима модел треба да препозна и категоризује више различитих типова објеката или феномена. Међутим, са овом сложеношћу долази и потенцијални ризик од overfitting-a, као и потреба за више података и снажнијим моделом.

- BPR (Bayesian Personalized Ranking) loss функција је специјализована за задатке рангирања, нарочито у области система за препоруке. Њена примена је најефикаснија у ситуацијама где је потребно уређивати предмете (као што су производи, филмови, књиге и сл.) у складу са персонализованим преференцијама корисника. Она функционише на јединствен начин који се битно разликује од традиционалних метода класификације попут BCE и Crossentropy. BPR модел ради тако што узима у обзир парове ставки: једна ставка коју је корисник пожелео (позитиван пример) и једна коју није (негативан пример). Циљ функције је да максимизује разлику у предвиђеној вредности између ових парова ставки. То значи да модел тежи да предвиди већу вероватноћу за ставке које корисник воли у односу на оне које не воли, чиме се постиже рангирање. Процес учења у BPR моделу се врши тако што се константно упоређују позитивни и негативни примери, при чему модел учи да прави разлику између ставки које су више или мање пожељне за корисника. Кроз овај процес, модел постепено усавршава своју способност да предвиди персонализовани редослед ставки који најбоље одговара сваком појединачном кориснику. У контексту препоручивачких система, BPR је посебно вредан јер нуди могућност за стварање високо персонализованих препорука. Уместо да се фокусира само на то да ли је нека ставка релевантна или не, BPR разматра како различите ставке треба да буду рангиране у односу једна на другу, што је често кључни аспект у ефикасним препоручивачким системима.

BCE и Crossentropy loss функције су два различита приступа која се користе у области машинског учења, а посебно у контексту класификације. Премда обе имају заједнички циљ - минимизацију грешке између предвиђених и стварних вредности - постоје значајне разлике у њиховој примени и подручју ефикасности. У светлу ових разматрања, избор између BCE, Crossentropy и BPR у великој мери зависи од специфичности задатка система за препоруке. Док BCE и Crossentropy нуде солидне перформансе у одређеним контекстима, BPR се често истиче као најпогоднији за комплексне задатке рангирања, што је чест случај у напредним системима за препоруке.

Архитектура GNN модела за систем препорука

Архитектура GNN модела за препоруке производа укључује конструкцију графа, инжењеринг карактеристика чворова и ивица, коришћење слојева GNN за учење уметања чворова и коришћење излазног слоја за генерисање препорука за сваког корисника. Предложено је неколико варијација архитектуре GNN модела, свака са својим предностима и ограничењима. У овом раду обучаваће се и упоређивати неколико различитих GNN модела, тако да ће општа схема изгледати као што је приказано на слици 40.



Слика 40. Процес изградње система препорука коришћењем GNN [20]

Евалуација система за препоруке заснованог на графовским неуронским мрежама

Евалуација система за препоруке заснованих на GNN је од суштинског значаја за одређивање њихове ефикасности и ефекта у препоручивању производа корисницима. У овом делу ћемо обрадити мерила и технике које се користе за оцењивање ефикасности система за препоруке заснованих на GNN.

1. **Мерила за евалуацију:** За оцењивање ефикасности система за препоруке користи се неколико мерила, укључујући прецизност, *recall* , F1-резултат и просечну прецизност (MAP). Прецизност изражава удео препоручених производа који су релевантни за корисника, док *recall* показује удео релевантних производа који су препоручени. F1-резултат је хармонијска средина прецизности и *recall*-а. MAP представља меру просечне прецизности на различитим нивоима опозива. Осим тога, често коришћено мерило за оцењивање ефикасности система за препоруке је тачност у најбољих K (top-K accuracy), која мери проценат тест примера у којима се тачна препорука налази међу K најбоље предвиђених препорука [26]. Ова мерила могу се користити за оцењивање тачности, релевантности и разноврсности препорука које генеришу системи за препоруке засновани на GNN.
2. **Стратегија поделе**: За оцењивање ефикасности система за препоруке заснованих на GNN, подаци се обично деле на скупове за обуку, валидацију и тестирање.

Скуп за обуку се користи за обучавање модела GNN, скуп за валидацију за прилагођавање хиперпараметара, а скуп за тестирање за оцењивање ефикасности модела на подацима који раније нису виђени. Различите стратегије поделе могу се применити, као што су случајна подела, временска подела или подела заснована на корисницима, у зависности од карактеристика скупа података.

3. **Студија аблације** (енг. Ablation Study): Студија аблације је техника која се користи за процену доприноса сваке компоненте модела GNN укупним перформансама. Ово укључује испитивање утицаја уклањања или модификовања одређених слојева или операција у моделу на његове крајње перформансе.
4. **Упоредне студије**: Упоредне студије се користе за оцењивање перформанси система за препоруке заснованих на GNN у односу на друге врсте система за препоруке, као што су матрична факторизација, колаборативно филтрирање и дубоке неуронске мреже. Ово помаже у идентификовању предности и недостатака GNN у контексту препоручивања.
5. **Квалитативна анализа**: Поред квантитативних мерила, квалитативна анализа се такође може користити за оцењивање ефикасности система за препоруке заснованих на GNN. Ово укључује анализу карактеристика препоручених производа, разноврсности препорука и способност модела да објасни своје препоруке.

Системи за препоруке засновани на GNN пружају обећавајуће резултате у области препоручивања производа. Евалуација ових система је кључна за разумевање њихове ефикасности и унапређење њихових перформанси. Кроз примену различитих мерила и техника евалуације, истраживачи и практичари могу боље разумети како системи за препоруке засновани на GNN функционишу и како их може унапредити за пружање бољих и релевантнијих препорука корисницима.

ИМПЛЕМЕНТАЦИЈА СИСТЕМА ЗА ПРЕПОРУКЕ УЗ ПОМОЋ ГРАФОВСКИХ НЕУРОНСКИХ МРЕЖА

Као што је у теоретском делу рада описано, графовске неуронске мреже представљају иновативан приступ решавању проблема препоруке у савременим системима за препоруке. Како је време одмицало, системи за препоруке су еволуирали, од колаборативног филтрирања, преко препоруке засноване на садржају и моделу, до модела заснованих на знању и на крају хибридних модела који имају карактеристике претходно наведених модела. Графовске неуронске мреже представљају хибридни приступ који комбинује особине основних модела препоруке како би се добио компететиван, тачан и исплатљивији модел, обзиром да у оваквим системима минимални напредак у креирању модела може довести до огромних, пре свега, финансијских добитака, који су компанијама најбитнији, ова област ће у будућности итетако још бити истраживана и системи ће постајати све квалитетнији и тачнији. За сада GNN представљају *state-of-the-art* решење па је у складу са тим свако истраживање од помоћи и свако побољшање модела могуће. Велике компаније се нису још у потпуности пребациле и имплементирале овакве системе, међутим, ове компаније активно раде на унапређивању својих система за препоруке, а GNN представљају област која се истражује и обећава, тако да су све претпоставке да ће у будућности овакви системи засновани на GNN бити честа појава код најмоћнијих играча што се тиче система за препоруке.

Технологије које ће бити коришћене у практичном делу рада су следеће: Visual Studio Code, Python 3.11, Jupyter Notebook екстензија за VS Code (.ipynb), популарну библиотеку PyTorch која је изабрана због предности у раду са графовским подацима, затим PyTorch Geometric (PyG) библиотека која представља основу за рад са GNN и она је надоградња PyTorch и лако се интегрира са различитим алатима које PyTorch пружа. У нашем задатку битна је због могућности рада са хетерогеним графовима што је њена предност у односу на конкурентне библиотеке, такође садржи и имплементацију захтеваних GNN-ова CAGE и GAT. Библиотека networkx се користи са циљем визуелизације графова. Додатне пратеће библиотеке за рад са подацима и моделима су: ast, copy, pickle, random, pandas, matplotlib, wordcloud, textblob, numpy, seaborn, gensim, nltk, sklearn, scipy… Рачунар на коме су извршене све евалуације и тренинзи има 8GB RAM меморије и процесор марке Intel генерације 3, 2.00GHz, графичка картица није употребљива за тренинг због тога што је старије генерације тако да ће се све извршавати на процесору и доступној RAM меморији.

Идеја

Идеја практичног дела јесте да се имплементирају најпопуларнији модели графовских неуронских мрежа (SAGE и GAT), затим да се исти примене на предикцију веза између чворова у графу, а да се затим евалуирају резултати и да упоредна анализа ефикасности, како модела графовских неуронских мрежа тако и најпопуларнијег приступа (Колаборативно филтрирање и факторизација матрице) за решавање проблема система за препоруке. Што на крају треба резултирати показивањем потенцијала GNN за решавање овакве врсте проблема.

Пре самог имплементирања ових мрежа и евалуације система потребно је испратити одређени скуп корака како би се добили подаци погодни за рад са овим мрежама. Проналажење скупа података одговарајућег за овај подухват, затим Препроцесирање података што уз коректну обраду подразумева и одабир релевантних фичера које је могуће искористити за тражени задатак, представља први корак. Након тога, потребно је превести податке у облик погодан за рад са графовским неуронским мрежама, што подразумева превођење скупа података у облик графа који одговара доступним подацима. Подела података на тренинг, тест и валидациони скуп представља додатни изазов код задатака овог типа, јер постоје многи параметри које потребно правилно подесити како би неуронска мрежа функционисала исправно. Након припреме података, креира се графовска неуронска мрежа, план је да се имплементирају два типа оваквих модела и то SAGE и GAT, који представљају два најпопуларнија приступа, и над њима изврши тренинг и процена ефикасности оваквог приступа. Након тренинга и подешавања хиперпараметара биће одрађена евалуација рада ових модела и модела колаборативног филтрирања као и пример препорука одређеним корисницима.

Слика 41. Процес имплементације практичног дела рада

Скуп података

Током тражења скупа података у обзир су узети различити услови, подаци су морали да буду погодни за задатак система за препоруке што подразумева постојање веза између корисника и производа, затим постојање оцена за производе, а онда и различитих фичера којима би модел графовске неуронске мреже могао бити „нахрањен“ како би научио репрезентације корисника и производа и на основу сличних карактеристика повезао исте. Затим је било потребно да скуп има велики број инстанци података како би био погодан и за колаборативно филтрирање и за GNN, поред тога вођено је рачуна о томе да нема много недостајућих података. Коришћени скуп података за овај задатак јесте мешавина два скупа података које је поделио Амазон и то управо за системе за препоруке под називом “Amazon product data” (https://snap.stanford.edu/data/amazon/productGraph/), један од скупова јесте “Amazon

review data - Tools and Home Improvement", а други је "Amazon metadata - Tools and Home Improvement". Како изгледају ови скупови:

{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano. He is having a wonderful time playing these old hymns. The music is at times hard to read because we think the book was published for singing from more than playing from. Great purchase though!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}

reviewerID - ID корисника, asin - ID производа, reviewerName – име корисника, helpful – корисност рејтинга корисника, reviewText – текст оцене, overall – оцена производа од стране корисника, summary – сажетак текста о оцени, unixReviewTime – време оцењивања (unix време), reviewTime – време остављања оцене.

{
  "asin": "0000031852",
  "title": "Girls Ballet Tutu Zebra Hot Pink",
  "price": 3.17,
  "imUrl": "http://ecx.images-amazon.com/images/I/51fAmVkTbyL._SY300_.jpg",
  "related":
  {
    "also_bought": ["B00JHONN1S", "B002R0F7FE", "B00E1YRI4C", "B008UBQZKU", "B00D103F8U", "B007R2RM8W"],
    "also_viewed": ["B002BZX8Z6", "B00B608000", "B008F0SMUC", "B00BFXLZ8M"],
    "bought_together": ["B002BZX8Z6"]
  },
  "salesRank": {"Toys & Games": 211836},
  "brand": "Coxlures",
  "categories": [["Sports & Outdoors", "Other Sports", "Dance"]]
}

asin - ID производа, title – име производа, price – цена у доларима, imUrl - *url* слике производа, related – повезани производи (такође купио, такође погледао, купио заједно, купио након оцењивања), salesRank – информације о производу, brand – назив бренда, categories – листа категорија којима припада производ.

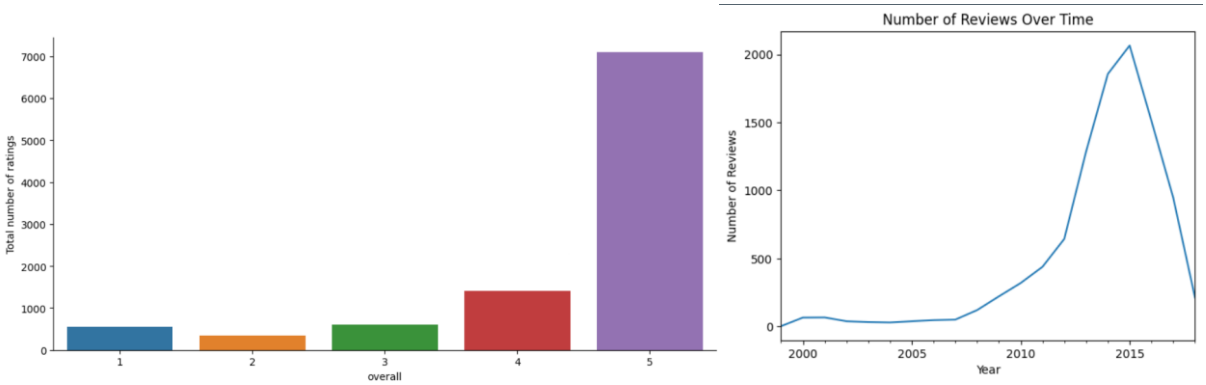
Ови скупови података су само део огромног Амазон скупа и представљају категорију која је везана за кућне алате и уређење куће, један скуп представља скуп остављених оцена од стране корисника за производе, док други скуп представља податке о производима, обзиром на то да радимо са графовским неуронским мрежама и да нам више информација о корисницима и производима директно пропорционално омогућава боље тренирање модела, и на постојање оваква два скупа, јасно је да је спајање ова два скупа, пошто су повезани идентификатором производа може допринети бољем тренирању модела. Па је сходно томе у наставку извршено спајање ова два сета података.

Скупови података заузимају доста меморије и сачувани су у .json формату, величина сета са оценама је 3.97 GB, има 9 015 203 инстанци, што говори да има толико оцена од стране корисника, док сет података са производима садржи 571 982 инстанци тј. информација о производима, док је његова величина 1 GB. Са овом количином података може се рећи да је могуће ефикасно тренирати како модел за GNN тако и за колаборативно филтрирање. Због великих количина података, како би се подаци ова два скупа обрадили, потребно је да се они поделе у мање скупове и одради појединачна обрада подскупова података. У наставку ће бити описан процес препроцесирања и предобраде података, а затим и спајања поменутих скупова.

Препроцесирање и предобрада података

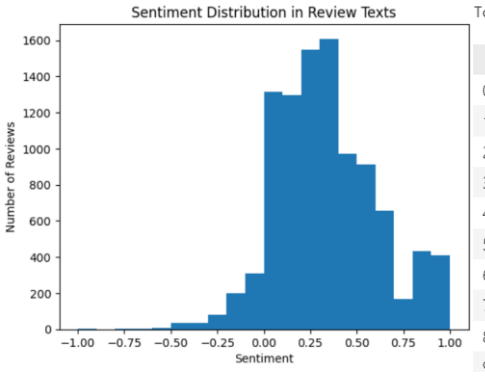
Препроцесирање скупа података са оценама за почетак подразумева поделу овог скупа података на мање скупове – *chunk*-ове из простог разлога што је количина података огромна и не може се учитати у меморију одједном, због тога ће анализа података у наставку бити извршена над малим скупом података од 10000 инстанци што је довољно да се репрезентују основне карактеристике података.

Резултати анализе података кажу следеће:
Count of unique Users : 9797, Count of unique Products : 88 – Број јединствених корисника је много већи од броја производа, што значи да има много различитих коментара за сваки производ, док нема много информација о преференцијама корисника ка производима, обзиром на то да је веома мали број њих дао више од једне рецензије.



Слика 42. Оцена – број оцена(лево), Година – број оцена(десно)

На слици 42(лево) можемо видети да је најпопуларнија оцена 5, док су остале оцене у много мањем броју, што је и очекиван резултат за *E-Commerce* продају, ово даје назнаке да ће код модела колаборативног филтрирања бити веома сличан резултат препоруке за многе кориснике и да ће матрица сличности можда чак давати резултате сличне средњој вредности оцене производа. На истој слици десно може се видети однос броја рецензија са годинама који иде узлазном путањом са годинама.



Top 10 most popular products

	asin	count
0	193362275X	3725
1	7300265137	1226
2	7109036146	544
3	1940019532	386
4	0982085028	275
5	750823765X	222
6	1933622741	211
7	7502175741	194
8	7502775447	171
9	B000022X0	160

Top 10 most unpopular products

asin	
7115138931	1
1590651529	1
7302094888	1
B000022314	2
B0000222UV	2
9178880661	2
7109074919	2
7223020318	3
1934047295	3
7806865365	3

Слика 43. Дистрибуција сентимената(лево), популарност производа(десно)

На слици 43(лево) је приказана дистрибуција сентимената у скупу података када је у питању опис оцене, распрострањеност сентимената не прати оцене које су дали корисници, огроман део сентимената је неутралан или половично позитиван, што не осликава стање са оценама(са слике 42), ово говори да оцене можда нису право мерило и да остали атрибути скупа могу равноправно да утичу на систем препорука, стога их треба укључити у евалуацију. На истој слици, десно, се може видети популарност производа, где можемо приметити да пар производа има огроман број рецензија у односу на остале, док с друге стране постоје производи који имају врло мало рецензија, ова небалансираност може довести до лоших препорука зато што ће најпопуларнији производи бацити у сенку мање популарне, треба се размишљати о решењу овог проблема, а да се не направи негативан утицај ни на популарне ни на не тако популарне производе. Колаборативно филтрирање доказано има проблема са оваквом поседнутошћу и поделом, од GNN се очекује да се успешно бори са овим проблемом.

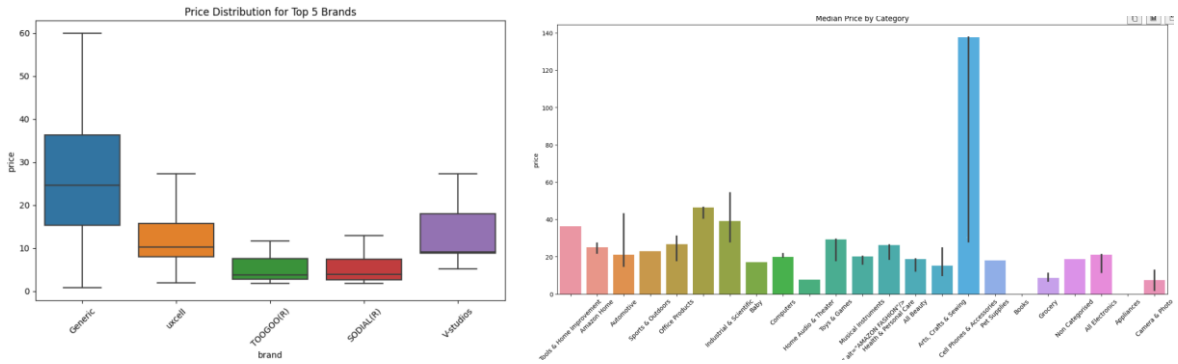
Из овог скупа су избачене колоне: 'vote', 'image' и 'style' пре свега због тога што има доста недостајућих података у њима, а затим и због тога што нам слика не би била од користи и правила би озбиљан временски изазов. Затим је енкодирана колона 'verified'. Главни проблем у наставку представљају текстуални подаци, због тога што графовске неуронске мреже раде са ембединзима, тако да се ови текстуални подаци морају превести у векторе, тј. да се токенизују, токенизација је урађена уз помоћ *Word2Vec* и то над атрибутима 'summary', 'reviewText' и 'reviewerName', са дужином вектора 100. Додате су колоне број прегледа производа и просечна оцена. Тако да је коначни изглед скупа након обраде података о оценама као на следећој слици:

	overall	verified	reviewerID	asin	reviewTime_year	reviewTime_month	reviewTime_day
0	3	1	A11F6V4DD7WND4	0972970169	2008	1	7

num_reviews	avg_overall	summary_embedding	reviewText_embedding	reviewerName_embedding
5	4.0	[0.03523069, 0.11031662, 0.0314211, -0.04870499, 0.012789654, -0.25267738, 0.108048305, 0.545657	[0.32878143, 0.5443114, -0.0144567955, -0.41948524, -0.1004955, -0.101737164, 0.60988843, 0.2369...	[-0.008190265, -0.005267412, -0.0041470574, -0.00022117546, 0.0034758472, -0.00065248506, -0.008...

Слика 44. "Amazon review data - Tools and Home Improvement" након обраде

Следеће је препроцесирање скупа везаног за производе, такође и овај скуп је превелик тако да се у наставку ради са једним издвојеним *chunk*-ом, који се састоји од 11535 инстанци. У старту је избачен велики број колона због великих недостатака у подацима, избачене су колоне 'date', 'imageURL', 'imageURLHighRes', 'similar_item','category', 'tech1', 'fit', 'tech2', 'rank', 'details', 'also_buy', 'also_view', у складу са тим остало је врло мало колона које опусију производ и то 'title', 'brand', 'feature', 'main_cat', 'price' и 'asin'.



Слика 45. Разноликост цене за најпопуларније брендове(лево) и средња цена по категорији(десно)

У овом сету је четири атрибута('title', 'brand', 'main_cat', 'feature_str') која требају да се преведу у векторе, што је и учињено на исти начин као и код података за оцене, изглед овог сета на крају предобраде је следећи:

	price	asin	title_embedding	brand_embedding
0	1.94	B01FOD3PG8	[-0.9921781, 0.17185117, 0.31034502, -0.053865295, 0.4419937, -0.87737817, 0.2402262, 0.9273665,...	[-0.0034941304, 0.005707793, -0.0029161104, -0.000762, 0.005399071, -0.035023566, -0.012122216, ...

main_cat_embedding	feature_str_embedding
[-0.09605328, 0.13511288, 0.024353098, 0.10977504, 0.12583356, -0.10256021, -0.045893088, 0.2308...	[-0.16374987, -0.112800665, 0.4182995, 0.1792441, 0.34141585, -0.18056883, 0.059733417, 0.763017...

Слика 46. "Amazon metadata - Tools and Home Improvement" након обраде

Након предобраде и препроцесирања основних скупова, креиран је коначан скуп података који ће бити коришћен за креирање графа за тренинг. Атрибути које садржи су 'overall','verified','reviewerID','asin','reviewTime_year','reviewTime_month', 'reviewTime_day','num_reviews','avg_overall','summary_embedding', 'reviewText_embedding','reviewerName_embedding','price','title_embedding', 'brand_embedding', 'main_cat_embedding', 'feature_str_embedding'.

	overall	verified	reviewerID	asin	reviewTime_year	reviewTime_month	reviewTime_day	num_reviews	avg_overall	summary_embedding
0	3.0	1	A11F6V4DD7WND4	0972970169	2008	1	7	5	4.0	[-0.0349934027, 0.3010466993, 0.2923053205, -0.2102584094, -0.4241042137, -0.0882628933, -0.0622854941, 0.1726019382,

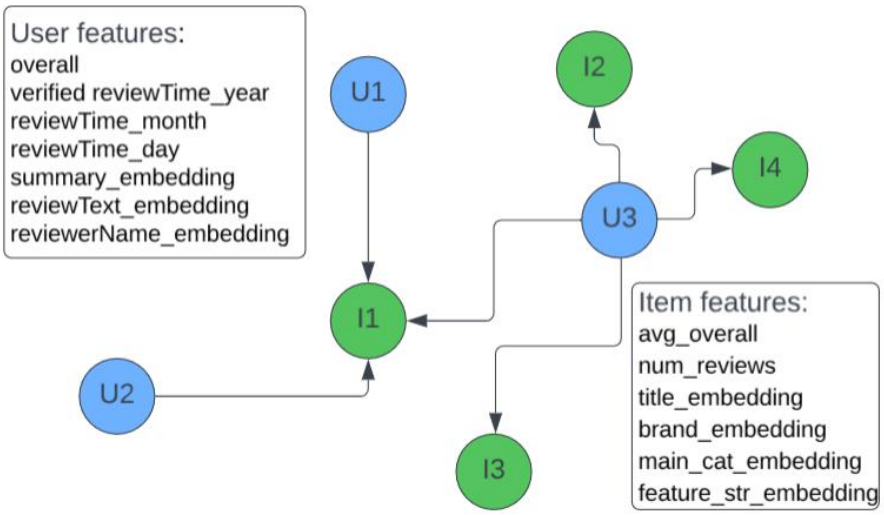
reviewText_embedding	reviewerName_embedding	price	title_embedding	brand_embedding	main_cat_embedding	feature_str_embedding
[0.1435533613, 0.2974196076, 0.0406210013, 0.3089621663, -0.1878142804, 1.0334198475, 0.5054050088, -0.6605418921,	[-0.018527884, -0.0252203979, 0.0289514419, -0.0192238018, -0.1012029648, -0.1017871425, 0.0602640063, 0.0942470729, 0.0256298799, -0.0573205054.	24.95	[0.4795634, 0.26978546, 0.41530558, 0.6573644, -0.057249546, -1.5672796, 1.0454497, 0.68714136, -0.91670555,	[-0.0077716955, 0.18858612, 0.036968283, -0.01429993, -0.2067404, -0.10090074, -0.22654955, 0.57370096, 0.14598724.	[0.0069790934, -0.0002095115, -0.00794701, 0.008890595, -0.008551635, 0.005559889, 0.0064678504, 0.0009757793, -0.008692562.	[0.19309111, -0.45927882, -1.6106459, 0.6315689, -0.3419104, -0.020687653, -0.96790636, -0.061966542, -0.6577551, 0.1824594,

Слика 47. Изглед коначног скупа за обраду

На слици 47 је приказан формат коначног скупа података припремљеног за рад са графовима и графовским неуронским мрежама.

Креирање и визуелизација графа

У теоријском делу су објашњене разне врсте графова, али је акценат стављен на хетерогене и бипартитне графове, из разлога што се управо такав граф формира у овом случају. Зашто? Због тога што ће се граф састојати од две врсте чворова, а то су чворови или *nodes* за кориснике и чворови за производе, то објашњава хетерогени граф, даље, чворови једног типа ће искључиво бити повезани са чворовима другог типа, док неће бити конекција између чворова истог типа, то објашњава појам бипартитни, коначно, наш циљ је креирање **хетерогеног бипартитног усмереног графа.** Оно што повезује чворове јесу везе или *edges*, оба типа чворова имаће своје атрибуте, које ће GNN узимати у обзир током рачунања ембединга за предикцију веза између чворова. Структура графа који треба бити креиран је следећа:



Слика 48. Структура хетерогеног бипартитног усмереног графа

На слици 48 може се видети структура жељеног графа, са именима атрибута који ће бити приписани одређеним чворовима у нашем случају, скуп $U=\{u_1, u_2,\ldots,u_n\}$ представља низ јединствених корисника, док је $I=\{i_1, i_2,\ldots,i_m\}$ низ јединствених производа. Гране, тј. везе између корисника и производа се представљају као уређени парови корисник-производ $E=\{\{u_1, i_1\}, \{u_2, i_1\}, \{u_3, i_1\}, \{u_3, i_2\}, \{u_3, i_3\}, \{u_3, i_4\},\ldots, \{u_n, i_m\}\}$. Потребно је припремити податке за овакву конструкцију графа, пре свега, јединствени идентификатори корисника и производа нису континуалне вредности већ су изведени из стрингова, што значи да морамо да их преведемо у континуалне бројне вредности које могу да се процесирају, а да затим креирамо везе између парова које се налазе у подацима, међутим, морају се запамтити мапирања како би касније преко идентификатора могли да прегледамо податке. [21] Поступак је следећи:

```
unique_reviewer_id = df['reviewerID'].unique()
mapped_reviewer_id = pd.DataFrame(data={
   'reviewerID': unique_reviewer_id,
   'mappedID': pd.RangeIndex(len(unique_reviewer_id)),
})
unique_product_id = df['asin'].unique()
mapped_product_id = pd.DataFrame(data={
   'asin': unique_product_id,
   'mappedID': pd.RangeIndex(len(unique_product_id)),
})

mapped_reviewer_id.to_csv('reviewer_mapping.csv', index=False)
mapped_product_id.to_csv('product_mapping.csv', index=False)
```

Прво су издвојени јединствени идентификатори, а затим мапирани у континуалне бројне вредности и од њих креиран фрејм података, затим су сачувана мапирања како би се касније могла учитати и применити за декодирање добијених резултата како би били читљиви од стране корисника/програмера.
Обзиром да се у GNN тензори користе за репрезентацију чворова и ивица, следећи корак јесте замена мапираних идентификатора у главном сету података и њихово превођење у тензорски облик, то је учињено овако:

```
reviews_reviewer_id = pd.merge(df['reviewerID'], mapped_reviewer_id,
                               left_on='reviewerID', right_on='reviewerID', how='left')
reviews_reviewer_id_tensor =
torch.from_numpy(reviews_reviewer_id['mappedID'].values)

ratings_product_id = pd.merge(df['asin'], mapped_product_id,
                              left_on='asin', right_on='asin', how='left')
ratings_product_id_tensor =
torch.from_numpy(ratings_product_id['mappedID'].values)
```

Након превођења у тензорски облик, идентификатори корисника и производа респективно су репрезентовани на следећи начин:

```
tensor([    0,     1,     2,  ..., 49334, 49335, 49335])
tensor([  0,   0,   0,  ..., 772, 772, 772])
```

Следећи корак јесте креирање ивица, тј. уређених парова корисник-производ, популарни назив за формат који се добија јесте COO (Coordinate) и он је уобичајен у PyTorch Geometric библиотеци.

```
edge_index_reviewer_to_product = torch.stack([reviews_reviewer_id_tensor,
ratings_product_id_tensor], dim=0)
edge_number = edge_index_reviewer_to_product.size()[1]
assert edge_index_reviewer_to_product.size() == (2, edge_number)
```

Stacking ових тензора хоризонтално(дуж димензије 0), креира 2D тензор где свака колона представља једну ивицу графа, први ред је извор(корисник тј. рецензент), други ред је циљ(производ). Следеће две линије представљају проверу да ли је тензор у тачном формату, тј. да ли су индекси ивица исправно форматирани. Прва димензија треба да буде 2(за извор и циљ), друга димензија треба да одговара броју ивица. Сврха овог кода је да конструише репрезентацију веза (ивица) у графу за коришћење у GNN. Свака ивица повезује рецензента с производом, што је корисно за анализе као што су препоруке производа, анализа сентимента рецензија и слично. COO формат је ефикасан начин за представљање ретких графова, што је често случај у реалним апликацијама.

Након припреме, коначно можемо креирати потребан хетерогени граф. PyTorch Geometric библиотека је од скоро омогућила нови тип графова, и то управо хетерогених, тако да ће граф бити креиран на тај начин.

```
from torch_geometric.data import HeteroData
data["reviewer"].node_id = torch.arange(len(mapped_reviewer_id))
data["product"].node_id = torch.arange(len(mapped_product_id))
```

Овај део кода креира празан HeteroData објекат из PyTorch Geometric за складиштење података хетерогеног графа и додељује јединствене целобројне идентификаторе чворовима за 'reviewer' и 'product', користећи torch.arange да креира тензоре са низом целих бројева, што представља кључни корак у припреми података за анализе и обраду користећи графовске неуронске мреже у PyTorch Geometric. Затим следи припрема атрибута чворова за додељивање, на следећи начин:

```
features = ['overall', 'verified', 'reviewTime_year', 'reviewTime_month',
'reviewTime_day',
        'summary_embedding', 'reviewText_embedding', 'reviewerName_embedding']
feature_tensors = []
for feature in features:
  if df[feature].dtype == object:  # Handling embedding features
    feature_tensor = torch.tensor(df[feature].tolist(), dtype=torch.float32)
  else:  # Handling scalar features
    feature_tensor = torch.tensor(df[feature].values,
dtype=torch.float32).unsqueeze(1)
  feature_tensors.append(feature_tensor)
reviewer_features = torch.cat(feature_tensors, dim=1)

product_data = df.groupby('asin').agg({
  'overall': 'mean',
  'reviewerID': 'size',
```

```
    'title_embedding': 'first',
    'brand_embedding': 'first',
    'main_cat_embedding': 'first',
    'feature_str_embedding': 'first'
}).reset_index()
product_data.rename(columns={'overall': 'avg_overall', 'reviewerID': 'num_reviews'},
inplace=True)
product_features_list = [
    torch.tensor(product_data[['avg_overall',                    'num_reviews']].values,
dtype=torch.float32),
    torch.tensor(product_data['title_embedding'].tolist(), dtype=torch.float32),
    torch.tensor(product_data['brand_embedding'].tolist(), dtype=torch.float32),
    torch.tensor(product_data['main_cat_embedding'].tolist(), dtype=torch.float32),
    torch.tensor(product_data['feature_str_embedding'].tolist(), dtype=torch.float32)
]
product_features = torch.cat(product_features_list, dim=1)

data['reviewer'].x = reviewer_features[reviews_reviewer_id_tensor]
data['product'].x = product_features[ratings_product_id_tensor]
```

У овом делу кода прво се одређује листа атрибута који ће бити укључени у тензор особина за рецензенте. Ова листа укључује различите особине као што су 'overall', 'verified', 'reviewTime_year', итд., па све до уграђених особина (embeddings) попут 'summary_embedding', 'reviewText_embedding', 'reviewerName_embedding'. Затим се креирају тензори особина за сваку особину из ове листе. Ако је особина уграђени тип (embedding), листа уграђених особина се конвертује у тензор. Ако је особина скаларна вредност, вредности се такође конвертују у тензор и додаје се нова оса да би се направио 2D тензор. Сви ови тензори се затим хоризонтално конкатенирају да би се креирао јединствени тензор особина за рецензенте. За чворове 'product' у графу, одређене особине се агрегирају на нивоу производа. То укључује израчунавање просечних 'overall' оцена и бројање рецензија. Такође се укључују различите уграђене особине асоциране са сваким производом. Слично као за рецензенте, креирају се тензори особина за 'product' чворове и ови тензори се такође хоризонтално конкатенирају. На крају, креирани тензори особина се додељују одговарајућим чворовима у HeteroData објекту. Ово омогућава да сваки чвор у графу има свој скуп особина који се може користити за даље обраде и анализе у GNN.

```
data["reviewer", "reviews", "product"].edge_index =
                                                edge_index_reviewer_to_product

from torch_geometric.transforms import ToUndirected
data = ToUndirected()(data)
print(data)
```

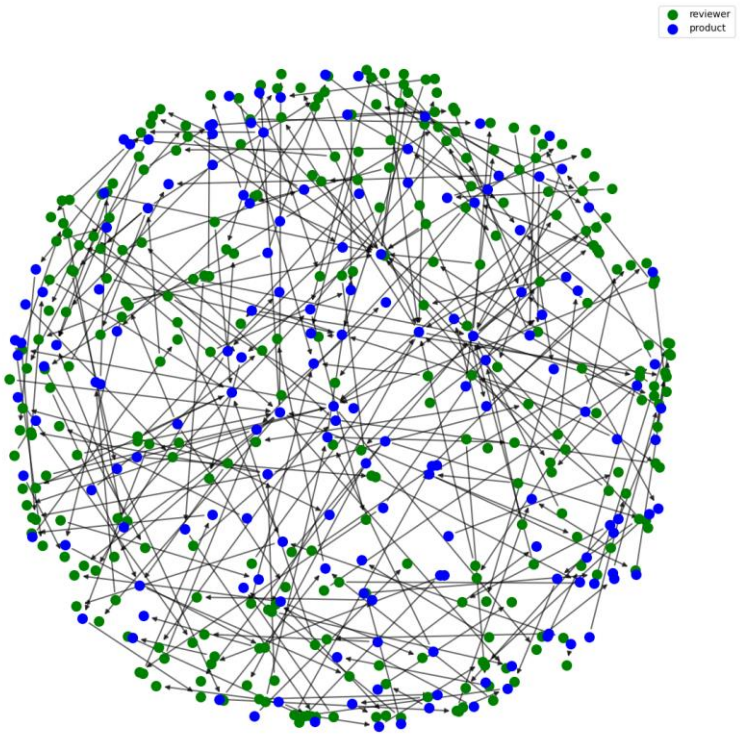
У овом делу кода обрађују се ивице у HeteroData објекту који представља граф у PyTorch Geometric. Прво се додају индекси ивица у HeteroData објекат да би се представиле везе између чворова 'reviewer' и 'product'. То се ради тако што се edge_index тензор, који представља ове везе, додељује атрибуту edge_index у HeteroData објекту. На овај начин се дефинишу ивице које повезују рецензенте с производима у графу. Даље, ради омогућавања двосмерног преноса порука у графовским неуронским мрежама

(GNN), додају се обрнуте ивице које иду од 'product' до 'reviewer' чворова. Ово се постиже коришћењем ToUndirected трансформације из PyTorch Geometric. ToUndirected трансформација модификује HeteroData објекат тако што додаје обрнуте ивице, омогућавајући да информације теку у оба смера између чворова. Ово је значајно у многим апликацијама GNN-а, јер мора да постоји унакрсна комуникација између чворова како би радили механизми за прослеђивање информација и порука. Хетерогени граф са 100000 инстанци након креирања је представљен на слици 50.

```
HeteroData(
  reviewer={
    node_id=[49336],
    x=[100000, 305],
  },
  product={
    node_id=[773],
    x=[773, 402],
  },
  (reviewer, reviews, product)={ edge_index=[2, 100000] },
  (product, rev_reviews, reviewer)={ edge_index=[2, 100000] }
)
```

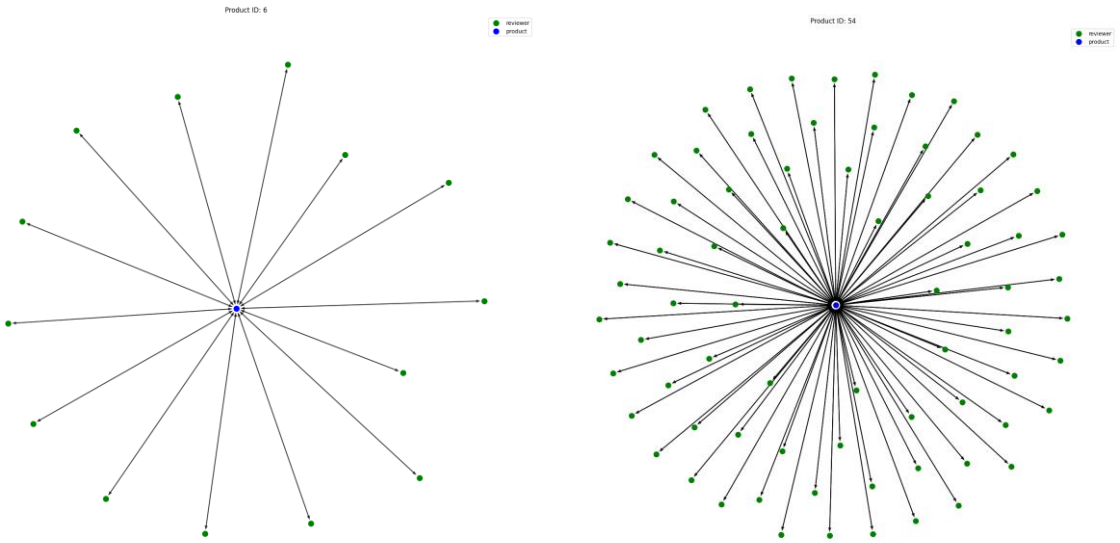
Слика 49. Изглед креираног хетерогеног бипартитног усмереног графа

За сликовити приказ графа, PyTorch Geometric нема дефинисану библиотеку, али се граф може приказати уз помоћ networkx библиотеке, наравно пре тога је потребно конвертовати граф у облик погодан за ову библиотеку, након конверзије граф се може сликовито представити, наравно, због количине података у графу, тј. броја чворова и ивица биће приказан мали део огромног графа:

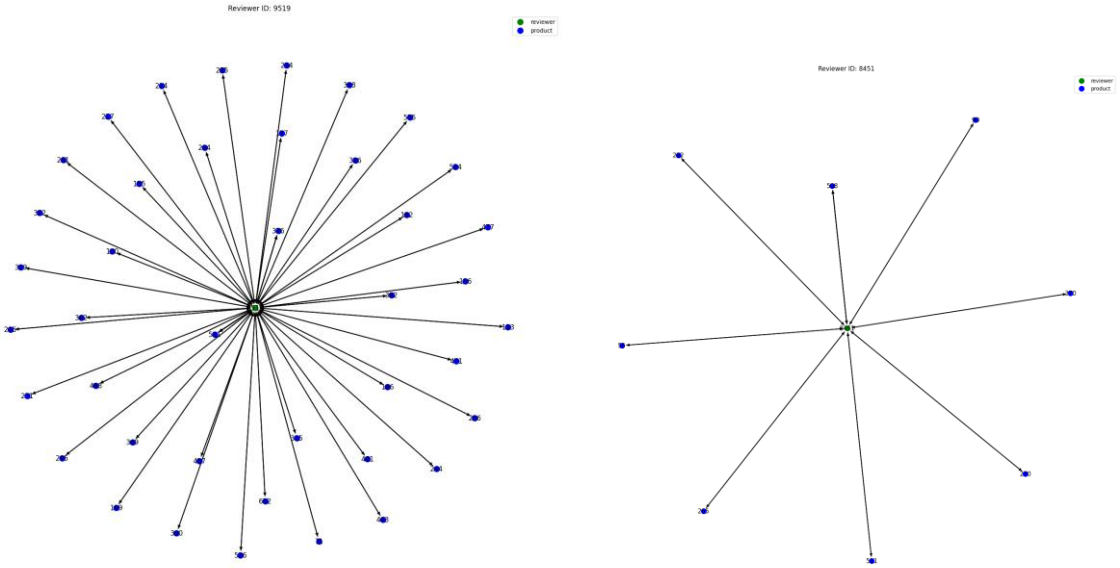


Слика 50. Визуелни приказ добијеног графа

Такође, у наставку ће бити приказани графови појединих чворова за рецензенте и производе, као на пример:



Слика 51. Приказ конекција насумичних чворова производа



Слика 52, Приказ конекција насумичних чворова рецензената(корисника)

На сликама 51 и 52 могу се видети конекције између насумичних чворова, са ових слика можемо закључити да је граф успешно креиран и да има доста конекција и што се тиче производа и што се тиче корисника, након овог корака креирања графа, може се кренути у припрему података за тренинг, креирање GNN и сам тренинг.

Подела података на тренинг, тест и валидациони скуп

Веома битан корак у поступку креирања система за препоруке, из простог разлога што је доста параметара потребно подесити како би модел био успешан, код поделе

података за графовске неуронске мреже. Обзиром на то да радимо са графовима, се поставља питање како ће потпуно повезан граф бити подељен на више делова тако да конекције не буду поремећене и оштећене, долази се до закључка да ће неке везе међу подацима засигурно бити раскинуте када су у питању тренинг подаци и да ће то можда утицати на рад модела, међутим PyTorch Geometric има имплементирану функцију за поделу података на тренинг, тест и валидациони скуп за проблеме предвиђања линкова, зове се RandomLinkSplit и у наставку ће бити исписана примена ове функције за поделу података, такође ће бити описан ефекат разних подешавања хиперпараметара ове функције.

```
transform_parameters = {
   "num_val":0.1,
   "num_test":0.2,
   "is_undirected":False,
   "disjoint_train_ratio":0.3,
   "neg_sampling_ratio":1.0,
   "add_negative_train_samples":True,
   "split_labels ":False,
   "edge_types":("reviewer", "reviews", "product"),
   "rev_edge_types":("product", "rev_reviews", "reviewer"),
}

transform = T.RandomLinkSplit(
   **transform_parameters
)
train_data, val_data, test_data = transform(data)
```

Исправно подешавање параметара изнад умногоме утиче на квалитет модела, па ће због тога сваки параметар бити засебно објашњен. "**num_val**" и "**num_test**", вредности додељене овим параметрима говоре о томе колико ће веза из графа бити додељену тест и валидационом скупу података. Затим, параметар "**is_undirected**" Ако је постављен на True, претпоставља се да је граф неусмерен и позитивни и негативни узорци неће довести до цурења повезаности ивица преко различитих подела. Ово утиче само на поделу графа, подаци о ознакама неће бити враћени као неусмерени. Ова опција се занемарује за бипартитне типове ивица или када edge_type != rev_edge_type. У нашем случају, обзиром да су нам везе бипартитне, можемо занемарити овај параметар. Параметар "**disjoint_train_ratio**", веома битан параметар, јер служи за контролу надзора над моделом током тренинга, ако је овај параметар постављен на 0.0, све везе из тренинг скупа ће размењивати поруке и неће постојати упоредне везе које би осигуравале упоређивање са тачним везама, тако да овај параметар мора бити подешен на вредност изнад нуле и он представља проценат тренинг података који се користи за надзор током тренинга, вредности су између 0.0 и 1.0, што представља проценат података који се користе за надзор. Проблем са класификацијом ивица, јесте тај, што ми за пример имамо само позитивне ивице, тј. само ивице које стварно постоје, и како би предвидели које ивице ће постојати у будућности, или за дате валидационе податке, немамо супротну страну, па је због тога потребно креирати ивице које ће током тренинга представљати супротну страну и омогућити класификацију ивица, поставља се одређени број негативних непостојећих ивица, углавном се додаје број негативних ивица једнак броју позитивних како би постојао баланс, овај параметар је "**neg_sampling_ratio**". "**add_negative_train_samples**" је параметар којим се одлучује да ли ће негативни узорци

веза бити додати сету података током тренинга или ће бити додати током поделе података, због истраживања, овај параметар мора бити подешен на позитивну вредност из простог разлога да би могли да упоредимо рад различитих модела над истим генерисаним подацима, што подразумева и исте негативне узорке. "**edge_types**":("reviewer", "reviews", "product") параметар који дефинише тип веза међу чворовима, а има и супротан параметар који је "rev_edge_types", због двоструке директне усмерености графа.

Обзиром на то да је задатак који се решава широког обима, да неки чворови имају велики број суседа, да је меморијски немогуће истренирати цео граф једном итерацијом, подаци се морају разложити на бечеве и морају се поделити конекције неких чворова, како због ефикаснијег тренинга и балансираности популарности производа тако и због количине података којом се располаже(милиони инстанци). LinkNeighborLoader ефикасно решава ове изазове кроз узорковање суседства (neighbor sampling), где се за сваки чвор узоркује одређен број суседа, дефинисан у loader_configs. Ово не само да побољшава рачунарску ефикасност и оптимизује употребу меморије, већ и омогућава учитавање података у серијама (batch processing) са опцијом мешања података (shuffle), што је кључно за ефикасно тренирање GNN-а и смањује ризик од преприлагођавања. Такође, LinkNeighborLoader омогућава генерисање негативних узорака (negative sampling који смо ми већ урадили па нема потребе и овде) потребних у задацима предвиђања веза, и прилагођава се специфичним потребама различитих GNN архитектура, чиме се постиже боља генерализација и прецизност модела.

```
from torch_geometric.loader import LinkNeighborLoader
loader_configs = {
   "Train_Loader_Config": {
      "num_neighbors": [20, 20],
      "batch_size": 512,
      "shuffle": True
   }
}
edge_label_index = train_data["reviewer", "reviews", "product"].edge_label_index
edge_label = train_data["reviewer", "reviews", "product"].edge_label
train_loader = LinkNeighborLoader(
   data=train_data,
   num_neighbors=loader_configs["Train_Loader_Config"]["num_neighbors"],
   #neg_sampling_ratio=1.0,
   edge_label_index=(("reviewer", "reviews", "product"), edge_label_index),
   edge_label=edge_label,
   batch_size=loader_configs["Train_Loader_Config"]["batch_size"],
   shuffle=loader_configs["Train_Loader_Config"]["shuffle"],
)
```

Имплементација графовске неуронске мреже

Графовска неуронска мрежа заузима средишњи део практичног дела рада, представља најбитнији део и одабрани начин решавања задатка, компликована имплементација која захтева опсежно истраживање и доста труда како би била

функционална. Кренућемо од основне поставке и конструкције, до ситнијих детаља мреже који ју је чине функционалном.

```
from torch_geometric.nn import SAGEConv, GATConv, to_hetero

class GNN(torch.nn.Module):
    def __init__(self, hidden_channels, conv_type=None):
        super().__init__()
        if conv_type == 'sage':
            self.conv1 = SAGEConv(hidden_channels, hidden_channels)
            self.conv2 = SAGEConv(hidden_channels, hidden_channels)
            #self.conv3 = SAGEConv(hidden_channels, hidden_channels)  # Third layer
        elif conv_type == 'gat':
            self.conv1 = GATConv(hidden_channels, hidden_channels,
                                 add_self_loops=False)
            self.conv2 = GATConv(hidden_channels, hidden_channels,
                                 add_self_loops=False)
            #self.conv3 = GATConv(hidden_channels, hidden_channels,
                                 add_self_loops=False)  # Third layer
        else:
            raise ValueError("Unknown conv_type")
    def forward(self, x: Tensor, edge_index: Tensor) -> Tensor:
        x = F.relu(self.conv1(x, edge_index))
        #x = F.dropout(x, training=self.training)
        #x = F.relu(self.conv2(x, edge_index))
        x = self.conv2(x, edge_index)
        #return torch.sigmoid(x)
        return x
```

Код изнад представља основни облик графовске неуронске мреже са применом два филтера за класификацију SAGEConv и GATConv. У зависности од изабраног conv_type, иницијализују се одговарајући конволутивни слојеви. SAGEConv слојеви се користе у GraphSAGE архитектури, која агрегира информације из суседних чворова користећи процедуру узорковања и агрегације. GATConv слојеви, са друге стране, примењују механизам пажње за тежинско узорковање информација из суседства, што омогућава моделу да се фокусира на важне чворове у околини. Овај приступ омогућава велику флексибилност у дизајнирању модела, јер се може лако променити тип конволуције или додати додатни слојеви. Такође, јасно одвајање слојева омогућава једноставније експериментисање и тестирање различитих архитектура.
О чему се ради у GNN која је имплементирана? GNN је креирана од два или три конволуциона слоја која уз помоћ функције forward или прослеђивања међусобно комуницирају и самим тим представљају дубину неуронске мреже, јер оба типа слојева функционишу прослеђивањем порука између чворова. Метод forward дефинише како се улазни подаци пропуштају кроз модел. Он прима тензоре чворова (x) и индексе ивица (edge_index). У овом методу, улазни подаци се прво пропуштају кроз први конволутивни слој, затим се примењује функција активације ReLU, која се користи после сваког конволутивног слоја да се уведе нелинеарност и побољша способност модела да учи сложене обрасце у графовским подацима, подаци се пропуштају кроз други(и трећи) конволутивни слој. Излаз функције forward у имплементираној GNN класи је трансформисани скуп особина чворова након што су пропуштени кроз конволутивне

слојеве. Конкретно, овај излаз представља нове ембединге (нове репрезентације) чворова графа након што су обрађени уз помоћ конволуција и функције активације ReLU.
Идеја код ових мрежа јесте прослеђивање порука, тј. размена информација и упоређивање репрезентација чворова како би се добиле повезаности између разних чворова и предвиделе везе између чворова. Зато је неопходан класификатор како би модел могао да се креира, класификатор дефинише начин на који се повезују репрезентације чворова у графу и како се рачунају њихове сличности.

```
from torch.nn import LSTM

class Classifier(torch.nn.Module):
    def __init__(self, hidden_channels, classifier_type='mean'):
        super().__init__()
        self.classifier_type = classifier_type
        if classifier_type == 'lstm':
            self.lstm = LSTM(input_size=hidden_channels, hidden_size=hidden_channels,
                                                                    batch_first=True)

    def forward(self, x_reviewer: Tensor, x_product: Tensor, edge_label_index: Tensor)
                                                                        -> Tensor:
        edge_feat_reviewer = x_reviewer[edge_label_index[0]]
        edge_feat_product = x_product[edge_label_index[1]]
        edge_features = edge_feat_reviewer * edge_feat_product  # Element-wise
multiplication

        if self.classifier_type == 'lstm':
            # LSTM processing
            edge_features = edge_features.unsqueeze(1)  # Add seq_len dimension
            lstm_out, _ = self.lstm(edge_features)
            lstm_out = lstm_out.squeeze(1)  # Remove seq_len dimension
            return lstm_out.mean(dim=-1)
        else:
            # Mean operation
            return edge_features.mean(dim=-1)
```

Класификатор у оквиру графовских неуронских мрежа (GNN) је креиран као класа која наслеђује torch.nn.Module, што омогућава лаку интеграцију и коришћење у PyTorch екосистему. У његовој основи, класификатор обрађује улазне податке - особине чворова (тензори x_reviewer и x_product) и индексе ивица (edge_label_index). На основу изабраног classifier_type, који може бити 'lstm' или 'mean', класификатор динамички одређује начин обраде ових података. За classifier_type 'lstm', користи се LSTM слој за детаљну и динамичку обраду особина ивица. Ово укључује проширење димензија улазних података, њихову обраду кроз LSTM слој, и даље средње вредновање излаза. С друге стране, за 'mean', класификатор једноставно „упросечава“ особине ивица.

Избор између LSTM и mean класификатора омогућава моделу да се прилагоди различитим врстама података и аналитичким задацима. LSTM агрегатор третира скуп узоркованих (енг. sampled) суседних чворова као секвенцу и користи LSTM архитектуру за обраду секвенце. Излаз последње јединице LSTM служи као резултат. Међутим, не постоји природни редослед међу суседима, због тога је усвојено насумично ређање. Ово

може значајно побољшати предвиђања у сложеним графовима. С друге стране, mean класификатор пружа бржу и једноставнију алтернативу, која је погодна за задатке где временске зависности нису изражене или где је потребна брза агрегација особина ивица. Интеграција ова два класификатора у модел графовских неуронских мрежа доноси значајну флексибилност и моћ у анализи и предвиђању података у графовима. Овај приступ омогућава експериментисање са различитим методама обраде података, оптимизујући модел за специфичне потребе и изазове који се јављају у раду са графовским подацима.

Коначно, након креирања класификатора и структуре неуронске мреже, примењујемо ове две класе за креирање коначног модела који ће бити коришћен за тренинг и над подацима и за креирање самих вероватноћа веза између чворова.

```
class Model(torch.nn.Module):
    def __init__(self, hidden_channels, conv_type=None, classifier_type='mean'):
        super().__init__()
        self.reviewer_emb = torch.nn.Embedding(data["reviewer"].num_nodes,
                                               hidden_channels)
        self.product_emb = torch.nn.Embedding(data["product"].num_nodes,
                                              hidden_channels)
        self.gnn = GNN(hidden_channels, conv_type=conv_type)
        self.gnn = to_hetero(self.gnn, metadata=data.metadata())
        self.classifier = Classifier(hidden_channels=hidden_channels,
                                     classifier_type=classifier_type)

    def forward(self, data: HeteroData) -> Tensor:
        x_dict = {
          "reviewer": self.reviewer_emb(data["reviewer"].node_id),
          #"product": self.product_lin(data["product"].x) +
self.product_emb(data["product"].node_id),
          "product": self.product_emb(data["product"].node_id),
        }
        # x_dict holds feature matrices of all node types
        # edge_index_dict holds all edge indices of all edge types
        x_dict = self.gnn(x_dict, data.edge_index_dict)
        pred = self.classifier(
            x_dict["reviewer"],
            x_dict["product"],
            data["reviewer", "reviews", "product"].edge_label_index,
        )
        return pred
```

У основи овог модела за графовске неуронске мреже стоји класа Model, која је кључна за интеграцију различитих компоненти неопходних за ефикасну обраду и анализу графовских података. Ова класа је структурирана као проширење torch.nn.Module, што јој омогућава да се лако уклопи у екосистем PyTorch-а и да буде флексибилна за различите апликације. Иницијализација укључује два ембединг слоја, reviewer_emb и product_emb, који претварају индексе чворова у густе векторске репрезентације. Ово је кључно за кодирање информација о рецензентима и производима у формату погодном за неуронске мреже. gnn компонента представља основни

графовски конволутивни модул. Врста конволуције (нпр. GraphSAGE или GAT) се бира на основу параметра conv_type. Овај модул је одговоран за обраду графа на начин који учи његову структуру и везе између чворова. Коришћењем to_hetero, GNN модул се трансформише у хетерогену форму која омогућава рад са графовима који имају више типова чворова и ивица, што је чест случај у реалним подацима. Компонента classifier се иницијализује са изабраним типом класификације ('lstm' или 'mean') и одговорна је за крајњу предикцију, процесуирајући информације издвојене од стране GNN модула.

У forward методу, модел прво примењује ембединг на чворове корисника и производа. Ове репрезентације се затим прослеђују кроз GNN модул који користи информације о ивицама (edge_index_dict) да би агрегирао и ухватио комплексне обрасце у графу. Након GNN обраде, добијене репрезентације се користе у класификатору за крајњу предикцију. Класификатор разматра карактеристике ивица, које су изведене из комбинованих особина чворова, и доноси предикцију на основу изабраног метода (LSTM или mean).

```
# For SAGE
#model_sage = Model(hidden_channels=128, conv_type='sage', classifier_type='lstm')
# For GAT
#model_gat = Model(hidden_channels=128, conv_type='gat', classifier_type='mean')
```

Класа Model представља основу за комплексну анализу и обраду графовских података, обједињујући различите елементе у један интегрисан систем. Са способношћу да обрађује хетерогене графове и примењује различите типове конволуција и класификација, овај модел нуди изузетну флексибилност и моћ у решавању разноликих задатака у графовској анализи, од предвиђања веза до класификације чворова. Ово га чини снажним алатом у истраживањима која укључују комплексне графовске структуре и податке. Код изнад представља пример креирања модела са различитим конфигурацијама.

Тренинг

Како би тренинг био могућ, потребно је дефинисати функцију за тренинг, када се ради са PyTorch-ем, идеја у овом пројекту јесте да се ова функција креира тако да буде погодна за истраживање, што подразумева адаптацију на разне функције губитака, хиперпараметара, приступе евалуацији рада над тренинг и тест подацима, праћење ефикасности рада и наравно визуелном приказу у напредовању тренинга. Функција је веома комплексна и има све потребне карактеристике прилагодљивости истраживању и експериментисању. У наставку су делови кода из train_model методе.

```
model_type = hyperparams['model_type']
    print(f"Training {model_type} model on device: '{hyperparams['device']}'")
    model = model.to(hyperparams['device'])
    optimizer = torch.optim.Adam(model.parameters(),
                                 lr=hyperparams['learning_rate'])

    # Define scheduler if using GATConv
    if model_type == 'GATConv':
        scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
```

```
        step_size=hyperparams['lr_decay_step'], gamma=hyperparams['lr_decay_factor'])
    best_val_loss = float('inf')
    best_epoch = -1
    epochs_without_improvement = 0

    train_losses = []
    test_losses = []
    loss_function_name = hyperparams.get('loss_function', 'bce')  # Default to 'bce' if
not provided

    if loss_function_name == 'crossentropy':
        loss_fn = F.binary_cross_entropy_with_logits
    elif loss_function_name == 'bce':
        loss_fn =  torch.nn.BCELoss()
    else:
        loss_fn = torch.nn.BCELoss()
```

У припремној фази тренинга модела у PyTorch-у, важно је изабрати прави тип модела из параметара hyperparams. Ови модели се потом постављају на одговарајући хардверски уређај (CPU или GPU) ради оптимизације перформанси. За оптимизацију се користи Adam оптимизатор због његове адаптивне способности промене брзине учења, а у случају GATConv модела користи се и scheduler који смањује брзину учења за фино подешавање модела. Кључни аспект тренинга је избор функције губитка, где се користе BCE (Binary Cross-Entropy) за бинарну класификацију и кросентропијски губитак са логитима за ситуације где излаз модела није ограничен сигмоидном функцијом. Ови приступи су значајни јер омогућавају моделу да учи минимизирањем разлике између предвиђања и стварних вредности, али се разликују у начину обраде излазних података модела, чинећи основу за успешан тренинг модела. Такође се користи *bpr* губитак.

```
def bpr_loss(user_interacted_scores, user_non_interacted_scores):
    # BPR loss calculation
    score_diff = user_interacted_scores - user_non_interacted_scores
    loss = -torch.mean(torch.log(torch.sigmoid(score_diff)))
    return loss
```

Функција bpr_loss рачуна губитак користећи Бајесовски персонални рангирање (Bayesian Personalized Ranking) тако што одузима резултате за неинтерактивне кориснике од резултата за интерактивне кориснике и примењује логистичку сигмоидну функцију на разлику, а затим израчунава средњу вредност негативног логаритма тих резултата.

```
for epoch in range(1, hyperparams['num_epochs'] + 1):
        model.train()
        total_loss = total_examples = 0
        for sampled_data in tqdm.tqdm(train_loader):
            optimizer.zero_grad()
            sampled_data.to(hyperparams['device'])
            if loss_function_name == 'bpr':
                pred = model(sampled_data)  # Model predictions for all edges
                edge_labels = sampled_data['reviewer', 'reviews', 'product'].edge_label
```

```
            # Ensure equal sampling of positive and negative samples
            positive_indices = torch.where(edge_labels == 1)[0]
            negative_indices = torch.where(edge_labels == 0)[0]

            num_samples = min(len(positive_indices), len(negative_indices))
            if num_samples > 0:
                negative_indices =
        negative_indices[torch.randperm(len(negative_indices))[:num_samples]]
                positive_indices =
        positive_indices[torch.randperm(len(positive_indices))[:num_samples]]

                positive_pred = pred[positive_indices]
                negative_pred = pred[negative_indices]

                loss = bpr_loss(positive_pred, negative_pred)
            else:
                continue
        else:
            pred = torch.sigmoid(model(sampled_data)) if loss_function_name !=
                            'crossentropy' else model(sampled_data)
            ground_truth = sampled_data["reviewer", "reviews", "product"].edge_label
            loss = loss_fn(pred, ground_truth)
        loss.backward()
        optimizer.step()
        total_loss += float(loss) * pred.numel()
        total_examples += pred.numel()

    avg_train_loss = total_loss / total_examples
    train_losses.append(avg_train_loss)  # Append to train losses

    model.eval()
    avg_test_loss = total_test_loss / total_examples
    test_losses.append(avg_test_loss)
    print(f"{model_type} - Epoch: {epoch:03d}, Train Loss: {avg_train_loss:.4f}
                                        Test Loss: {avg_test_loss:.4f}")
```

Процес тренинга модела у PyTorch-у обухвата низ итеративних корака који се одвијају кроз више епоха, где свака епоха представља један пролазак кроз цео тренинг скуп података. Тренинг се одвија у оквиру for петље која броји епохе, почевши од 1 па све до максималног броја епоха дефинисаног у hyperparams['num_epochs']. На почетку сваке епохе, модел се поставља у тренинг мод (model.train()), што омогућава активацију специфичних функција попут dropout-а који се користе само током тренинга. Затим, петља пролази кроз све податке у train_loader, који доставља узорке података у серијама. За сваки узорак података, прво се обавља ресетовање градијената оптимизатора optimizer.zero_grad(), које је потребно за правилно ажурирање тежина модела. Подаци се пребацују на одговарајући уређај hyperparams['device'], а затим се обрађују кроз модел. У зависности од типа функције губитка (нпр. 'crossentropy' или 'bce', 'bpr'), примењује се одговарајућа активациона функција на излаз модела. Након добијања предвиђања од модела, израчунава се губитак у поређењу са правим вредностима (ground_truth). Код бпр губитка је битан фактор балансираност позитивних и негативних примера веза, па се

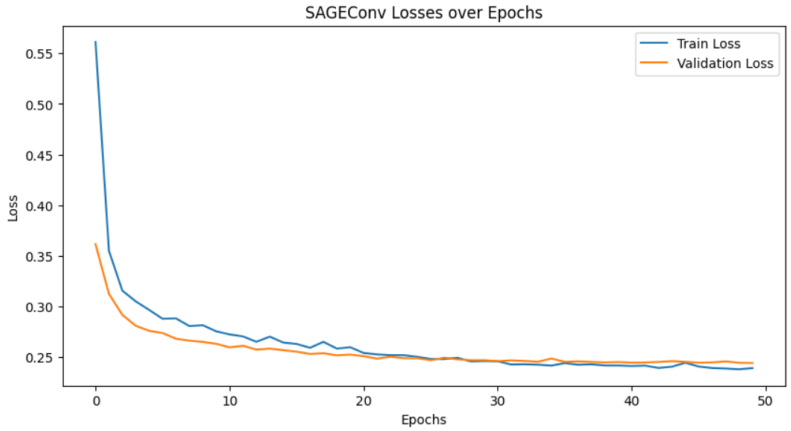
због тога укључује логика балансирања ових веза. Овај губитак се затим користи у процесу backpropagation-a (loss.backward()), након чега се оптимизатор користи за ажурирање тежина модела (optimizer.step()). Укупан губитак и број примера се акумулирају ради каснијег израчунавања просечног губитка за тренинг.

Након завршетка тренинг фазе у току једне епохе, модел се пребацује у мод за тестирање (model.eval()), који онемогућава функције као што су dropout. У овој фази, користећи torch.no_grad() да се спречи израчунавање градијената, модел обрађује податке из test_loader. Губитак израчунат на тест скупу такође се акумулира за израчунавање просечног губитка на тесту. Ови резултати се користе за праћење перформанси модела кроз време и за идентификацију најбољег модела на основу тест губитка. Овај приступ омогућава темељно и ефикасно обучавање модела, где се непрестано прати и усавршава перформанса модела кроз више итерација, омогућавајући моделу да се прилагоди и оптимизује за задати задатак. Примењује се логика раног заустављања и чувања најбољег модела. Ако је губитак на тест скупу мањи од претходно најбољег, то стање модела се чува. У супротном, ако нема побољшања у губитку кроз дефинисан број епоха (patience), тренинг се зауставља како би се спречило преприлагођавање. Након сваке епохе, за GATConv моделе, ажурира се scheduler. На крају, приказују се графикони губитака за тренинг и тест, а модел се враћа на стање из најбоље епохе, осигуравајући оптималну перформансу модела.

Пример позива тренинг функције и креирања модела:

```
hyperparams_sage = {
    "model_type": 'SAGEConv',  # Include model_type here
    "device": torch.device('cuda' if torch.cuda.is_available() else 'cpu'),
    "learning_rate": 0.001,
    "num_epochs": 150,
    "patience": 20,
    "loss_function": 'crossentropy',
}
model_sage = Model(hidden_channels=64, conv_type='sage', classifier_type='mean')
best_epoch_sage, best_val_loss_sage = train_model(model_sage, train_loader, test_loader, hyperparams_sage)
print(f"The best model was from epoch {best_epoch_sage} with a validation loss of {best_val_loss_sage:.4f}")
```

Изглед резултата тренинга за специфичну конфигурацију:



Слика 53. Резултат тренинга GNN модела

Следећи корак јесте евалуација тренинга над валидационим подацима, валидациони подаци се учитавају преко loader-а као и тренинг и тест подаци, с тим што ће поента бити да се уместо над бечевима евалуира модел, да се евалуира над целим валидационим скупом. Тај скуп представља sampled_data и над њим ће се вршити евалуације.

```
def prepare_data(sampled_data, model, device):
    model = model.to(device)
    model.eval()

    with torch.no_grad():
        pred = model(sampled_data)

    ground_truth_labels = sampled_data["reviewer", "reviews", "product"].edge_label
    probabilities = torch.sigmoid(pred)
    binary_predictions = (probabilities >= 0.5).float()

    return ground_truth_labels, probabilities, binary_predictions
```

Можда и најбитнија функција што се тиче евалуације, из простог разлога што је пре свега потребно усвојити начин на који ће се добијени резултати посматрати. Да би уопште упоређивали податке потребно је извући праве конекције између чворова, док ће се предвиђене конекције узимати као резултат обраде излаза сигмоидалне функције која даје вредности између 0 и 1, сматра се да су вредности за које је добијен резултат вероватноће (probabilities >= 0.5) које су веће или једнаке од 0.5 након сигмоидалне функције, вредности позитивних конекција, тачније предиктованих конекција и за узорковане и реалне податке. Док ће ground_truth_labels служити за упоредно анализу са предвиђеним вредностима.

```
def get_positive_negative_indices(ground_truth_labels):
    positive_edge_indices = (ground_truth_labels == 1).nonzero(as_tuple=True)[0]
    negative_edge_indices = (ground_truth_labels == 0).nonzero(as_tuple=True)[0]
    return positive_edge_indices, negative_edge_indices
```

Осим што модел предвиђа постојеће конекције, он предвиђа и узорковане конекције, тако да ће током евалуације бити потребно раздвојити предикције за непостојеће везе и за стварно постојеће, функција get_positive_negative_indices враћа податак о везама које су позитивне и негативне и тиме помаже при евалуацији.

За основну евалуацију модела, током истраживања са различитим хиперпараметрима, лос функцијама и моделима, биће коришћени основни параметри и функције за евалуацију, како би добили најбољи модел GNN са подешавањима која дају најбоље резултате, након проналаска најбољег модела ће се приступити упоредној анализи и стварном коришћењу препорука истренираног модела. Функције које ће се користити су:

- **plot_confusion_matrix_for_positive_edges**: Визуализује матрицу забуне за позитивне ивице, омогућавајући боље разумевање перформанси модела у класификацији позитивних ивица.

- **evaluate_models_auc**: Рачуна подручје испод ROC криве (AUC) за моделе, што је мера квалитета модела у разликовању између класа.
- **evaluate_accuracy**: Израчунава тачност модела одвојено за позитивне и негативне ивице, пружајући детаљнију анализу перформанси.
- **count_predictions**: Броји колико пута је модел предвидео одређену класу (1 или 0) за позитивне и негативне ивице, што помаже у разумевању склоности модела ка одређеној класи.
- **evaluate_precision_recall**: Израчунава прецизност и опозив за моделе, што су важне метрике за оцену квалитета класификације, посебно у ситуацијама неуређене дистрибуције класа.

Проналажење најефикаснијег модела

Како би пронашли најефикасније поставке и модел, потребно је да то испитамо мануелно испробавањем различитих комбинација, ово представља временски најзахтевнији корак и због тога је битно да све претходно описане функције буду ваљано имплементиране како се не би појавила грешка у самом процесу тренинга са различитим конфигурацијама. Битно је поменути да су упитни параметри специфични за системе за препоруке и да су нијансе те које ће одлучити који модел ћемо користити у наставку.

Параметри који су од интереса јесу лос функција, тип класификатора и наравно тип графовске неуронске мреже. Па ће у складу са тим бити испитано која мрежа се боље понаша над нашим подацима да ли је то 'SAGEConv' или 'GATConv'. Према истраживањима, обзиром на свеопшту употребу, која је на страни SAGE модела, посвећено је више пажње овом приступу, па се и очекују бољи резултати. Међутим ГАТ мреже су донеле револуцију у GNN тако да постоји могућност квалитетнијих предикција. У наставку ће сваки резултат бити испитан и прокоментарисан, биће дато виђење предности и мана ових модела. Што се тиче лос функција, за проблем препорука су најчешће коришћене следеће три: 'bce', 'crossentropy' и 'bpr', биће примењена свака од ових функција у комбинацији свих осталих параметара, тако да ћемо на крају издвојити најбољу за наш проблем, очекује се да 'bpr' губитак покаже најбоље резултате. Најпопуларнији типови класификатора код GNN јесу управо 'mean' и 'lstm', тако да ће и они бити укључени у избор најбољег. У наставку ће бити приказани резултати тренинга над 10000 инстанци података, где је подела на тренинг, тест и валидационе податке следећа: 70% за тренинг податке, 20% за тест и 10% за валидацију. Наравно, приказани резултати тренинга ће бити резултати примене модела над валидационим скупом података, који се састоји од 1000 позитивних и 1000 негативних веза међу чворовима.

Прво ћемо се осврнути на различите класификаторе:

Model	Classifier Type	Loss Function	Best Epoch	Best Validation Loss	Validation AUC	Accuracy for Positive Edges	Accuracy for Negative Edges	Precision	Recall	F1 Score
SAGEConv	mean	bce	101	0.183205	0.8590	0.828	0.893	0.885561	0.828	0.855814
GATConv	mean	bce	28	0.330976	0.8075	0.816	0.837	0.833504	0.816	0.824659
SAGEConv	mean	crossentropy	70	0.192621	0.8490	0.811	0.884	0.874865	0.811	0.841723
GATConv	mean	crossentropy	33	0.308855	0.8290	0.808	0.858	0.850526	0.808	0.828718
SAGEConv	mean	bpr	58	0.189143	0.8610	0.831	0.900	0.892589	0.831	0.860694
GATConv	mean	bpr	37	0.298253	0.8235	0.813	0.837	0.832992	0.813	0.822874

Слика 54. Резултати тренинга за класификатор 'mean'

На слици 54 је приказ основних резултата основних валидационих функција у случају примене класификатора илити агрегатора 'mean' при прослеђивању порука између суседних чворова. Најбољи резултат испод ROC криве даје САГЕ модел, са тачношћу од 86.1% овај модел даје најбоље резултате и за позитивне и за негативне ивице, што га чини најбољим избором у комбинацији са 'mean' класификатором. Може се приметити да је комбинација са 'bpr' функцијом губитка, као што је очекивано за проблем препорука дала најбољи резултат, за овај резултат је био потребан оптималан број епоха. Са слике се такође може видети да је за све комбинације параметара резултат над негативним везама бољи него над позитивним, што је и очекивано у овом случају и што показује квалитет предвиђања за позитивне конекције међу чворовима које минимално заостаје. Занимљиво је за приметити да 'bce' и 'crossentropy' функције губитка, које имају исту структуру, само је код друге додат корак више и подаци су прецизнији, да ГАТ модел показује зависност од прецизнијих података за евалуацију, док, обрнуто пропорционално САГЕ модел даје боље резултате за заокружене вредности које се прослеђују током комуникације, што овај модел чини прилагодљивијим. Закључак је да је у овом случају са овим типом класификације, тј. агрегације комбинација SAGE-mean-bpr најбољи избор. Што је у великој мери очекивано, са разликом што је код САГЕ модела очекиван бољи резултат са lstm приступом агрегације, у наставку ћемо видети да ли је теорија оправдана у пракси.

Model	Classifier Type	Loss Function	Best Epoch	Best Validation Loss	Validation AUC	Accuracy for Positive Edges	Accuracy for Negative Edges	Precision	Recall	F1 Score
SAGEConv	lstm	bce	53	0.201073	0.8595	0.832	0.887	0.880423	0.832	0.855527
GATConv	lstm	bce	56	0.320359	0.8365	0.801	0.855	0.846723	0.801	0.823227
SAGEConv	lstm	crossentropy	101	0.180258	0.8500	0.795	0.895	0.883333	0.795	0.836842
GATConv	lstm	crossentropy	59	0.310715	0.8310	0.796	0.879	0.868048	0.796	0.830464
SAGEConv	lstm	bpr	56	0.194844	0.8640	0.837	0.895	0.888535	0.837	0.861998
GATConv	lstm	bpr	34	0.336984	0.8290	0.755	0.875	0.857955	0.755	0.803191

Слика 55. Резултати тренинга за 'lstm' класификатор

При анализи представљених резултата, уочава се да 'lstm' класификатор уопштено надмаши 'mean' класификатор у контексту вредности испод ROC криве у свим тренинзима. Међутим, у детаљнијем прегледу постаје јасно да је 'lstm' класификатор превасходно ефикаснији у предвиђању непостојећих веза, док се у предвиђању постојећих веза - основном задатку - показује као знатно слабија опција у поређењу са 'mean' класификатором. Остале мере као што су прецизност и *recall* нуде мешовите резултате, те стога не пружају јасан увид у то који модел са којим параметрима нуди боље укупне перформансе. Ипак, комбинација SAGE-lstm-bpr се истиче као оптималан избор за 'lstm' класификатор, не само због повећане тачности предвиђања за позитивне везе, већ и због смањене тачности предвиђања за негативне везе, што указује на складан однос и представља идеалну опцију међу разматраним комбинацијама параметара.

Model	Classifier Type	Loss Function	Best Epoch	Best Validation Loss	Validation AUC	Accuracy for Positive Edges	Accuracy for Negative Edges	Precision	Recall	F1 Score
SAGEConv	mean	bce	101	0.183205	0.8590	0.828	0.893	0.885561	0.828	0.855814
GATConv	mean	bce	28	0.330976	0.8075	0.816	0.837	0.833504	0.816	0.824659
SAGEConv	lstm	bce	53	0.201073	0.8595	0.832	0.887	0.880423	0.832	0.855527
GATConv	lstm	bce	56	0.320359	0.8365	0.801	0.855	0.846723	0.801	0.823227

Слика 56. Приказ резултата у односу на 'bce' функцију губитака

'bce' даје добре резултате у случају САГЕ модела и то су једни од најбалансиранијих резултата, међутим кад је у питању ГАТ модел, јасно је да је и поред промене функције губитака 'mean' тип класификатора са овим типом модела даје најлошије резултате. 'bce' функција губитака са свим типовима модела и класификатора даје просечне резултате и представља добар избор у случају тренинга, поготово из разлога што за различите комбинације даје велику прецизност за позитивне везе, наравно, очекивања су да 'crossentropy' да боље резултате од ове функције губитака, што ћемо проверити у наставку.

Model	Classifier Type	Loss Function	Best Epoch	Best Validation Loss	Validation AUC	Accuracy for Positive Edges	Accuracy for Negative Edges	Precision	Recall	F1 Score
SAGEConv	mean	crossentropy	70	0.192621	0.849	0.811	0.884	0.874865	0.811	0.841723
GATConv	mean	crossentropy	33	0.308855	0.829	0.808	0.858	0.850526	0.808	0.828718
SAGEConv	lstm	crossentropy	101	0.180258	0.850	0.795	0.895	0.883333	0.795	0.836842
GATConv	lstm	crossentropy	59	0.310715	0.831	0.796	0.879	0.868048	0.796	0.830464

Слика 57. Приказ резултата у односу на 'crossentropy' функцију губитака

Очекивања нису оправдана и 'crossentropy' функција губитака даје најлошије резултате предикције по скоро свим параметрима. Међутим, треба се приметити да поред тога што даје незнатно слабије резултате ова функција губитка представља стабилан избор за тренинг података када је у питању проблем препоруке. Следећи је теоретски фаворит, 'bpr'. Према очекивањима, 'bpr' функција губитака даје најбоље резултате, али само за САГЕ модел. Закључак је да ако бирамо стабилну функцију губитака, бирамо 'crossentropy', ако нам је битна прецизност предвиђања позитивних веза бирамо 'bce' и ако бирамо највећу тачност по цену потенцијалног фијаска, бирамо 'bpr' функцију губитака.

Model	Classifier Type	Loss Function	Best Epoch	Best Validation Loss	Validation AUC	Accuracy for Positive Edges	Accuracy for Negative Edges	Precision	Recall	F1 Score
SAGEConv	mean	bpr	58	0.189143	0.8610	0.831	0.900	0.892589	0.831	0.860694
GATConv	mean	bpr	37	0.298253	0.8235	0.813	0.837	0.832992	0.813	0.822874
SAGEConv	lstm	bpr	56	0.194844	0.8640	0.837	0.895	0.888535	0.837	0.861998
GATConv	lstm	bpr	34	0.336984	0.8290	0.755	0.875	0.857955	0.755	0.803191

Слика 58. Приказ резултата у односу на 'bpr' функцију губитака

Model	Classifier Type	Loss Function	Best Epoch	Best Validation Loss	Validation AUC	Accuracy for Positive Edges	Accuracy for Negative Edges	Precision	Recall	F1 Score
SAGEConv	lstm	bpr	56	0.194844	0.8640	0.837	0.895	0.888535	0.837	0.861998
SAGEConv	mean	bpr	58	0.189143	0.8610	0.831	0.900	0.892589	0.831	0.860694
SAGEConv	lstm	bce	53	0.201073	0.8595	0.832	0.887	0.880423	0.832	0.855527
SAGEConv	mean	bce	101	0.183205	0.8590	0.828	0.893	0.885561	0.828	0.855814
SAGEConv	lstm	crossentropy	101	0.180258	0.8500	0.795	0.895	0.883333	0.795	0.836842
SAGEConv	mean	crossentropy	70	0.192621	0.8490	0.811	0.884	0.874865	0.811	0.841723
GATConv	lstm	bce	56	0.320359	0.8365	0.801	0.855	0.846723	0.801	0.823227
GATConv	lstm	crossentropy	59	0.310715	0.8310	0.796	0.879	0.868048	0.796	0.830464
GATConv	mean	crossentropy	33	0.308855	0.8290	0.808	0.858	0.850526	0.808	0.828718
GATConv	lstm	bpr	34	0.336984	0.8290	0.755	0.875	0.857955	0.755	0.803191
GATConv	mean	bpr	37	0.298253	0.8235	0.813	0.837	0.832992	0.813	0.822874
GATConv	mean	bce	28	0.330976	0.8075	0.816	0.837	0.833504	0.816	0.824659

Слика 59. Резултати тренинга

На слици изнад приказани су резултати примењених тренинг модела са различитим конфигурацијама над валидационим подацима сортирани по вредностима испод ROC криве. Као што се може приметити модел графовске неуронске мреже САГЕ даје боље резултате од ГАТ модела за све комбинације основних параметара, тако да је за наш скуп података, САГЕ убедљиво бољи избор. Када је у питању тип класификатора чини се да оба типа које примењујемо поред различитог начина функционисања, дају сличне резултате, док код САГЕ модела постоје незнатне разлике у типовима класификатора, код ГАТ модела је приметно бољи избор lstm класификатор. Функција губитака се код САГЕ модела сама издвојила и 'bpr' представља најбољи избор док су остале две склоне и *overfiting*-у, с друге стране код ГАТ модела најбоље резултате дају функције губитака засноване на бинарној кросентропији. Закључак је да код нашег проблема требамо користити **САГЕ модел** у комбинацији са **'bpr'** функцијом губитака, док је избор класификатора мање битан фактор и може се одабрати било који, с тим да се треба узети у обзир да је теоретски гледано **'lstm'** поузданији класификатор, што се може и видети из приложеног.

У наставку ће бити дати резултати исте врсте испитивања, само са већом количином података, тачније са 100000 инстанци података, а онда и анализа утицаја количине података на разне комбинације хиперпараметара. Наравно, временски захтевнији задатак, што је једна од мана графовских неуронских мрежа, јер за веће количине података треба много више времена.

Model	Classifier Type	Loss Function	Best Epoch	Best Validation Loss	Validation AUC	Accuracy for Positive Edges	Accuracy for Negative Edges	Precision	Recall	F1 Score
SAGEConv	lstm	bpr	50	0.244171	0.82415	0.8035	0.8436	0.837066	0.8035	0.819940
SAGEConv	lstm	bce	103	0.418516	0.82090	0.8121	0.8343	0.830538	0.8121	0.821215
SAGEConv	lstm	crossentropy	70	0.417983	0.81675	0.8139	0.8182	0.817415	0.8139	0.815654
SAGEConv	mean	bpr	97	0.030421	0.81080	0.7475	0.8763	0.858012	0.7475	0.798953
SAGEConv	mean	crossentropy	20	0.129686	0.80270	0.7774	0.8264	0.817455	0.7774	0.796925
SAGEConv	mean	bce	24	0.130804	0.79855	0.7820	0.8155	0.809105	0.7820	0.795322
GATConv	lstm	crossentropy	25	0.495159	0.78525	0.9051	0.6708	0.733290	0.9051	0.810187
GATConv	mean	crossentropy	96	0.323124	0.78425	0.7679	0.7944	0.788803	0.7679	0.778211
GATConv	lstm	bpr	37	0.353741	0.77830	0.8742	0.6823	0.733451	0.8742	0.797664
GATConv	mean	bce	110	0.322787	0.76800	0.7528	0.7975	0.788025	0.7528	0.770010
GATConv	lstm	bce	71	0.483676	0.76380	0.7457	0.7846	0.775882	0.7457	0.760492
GATConv	mean	bpr	104	0.165012	0.64005	0.9908	0.2850	0.580842	0.9908	0.732353

Слика 60. Резултати тренинга над 100000 инстанци података

За тренинг свих комбинација специфичних хиперпараметара у случају са 100000 инстанци било је потребно 1568 минута, што је мало више од 26 сати или више од једног дана, по моделу је потребно око 130 минута, наравно са доступним хардверским ресурсима. Приметно је да су резултати над великом количином података приближно слични као резултати са мањом количином, метрике и хиперпараметри слично, или исто, директно пропорционално прате резултати мањег скупа, за најбољу комбинацију је поново добијена комбинација **САГЕ модела, 'bpr'** функције губитака и **'lstm'** класификатора, што оправдава коришћење подскупа великог скупа података за истраживање. САГЕ модел се издвојио и у овом случају и тотално искључио гат модел из евалуације, али ГАТ је у овом случају за специфичну комбинацију параметара дао изузетне резултате за позитивне реалне везе, тако да ће то бити додатно испитано. 'lstm' класификатор се такође поново издвојио и то за више случајева комбинација, што важи и за 'bpr' и 'bce' функције губитака**.** Међутим, са већом количином података, очекивани

су иоле бољи резултати због веће распрострањености веза међу подацима, али то се није десило, разлог томе је објашњив, повећана количина података поред више веза, доноси и више комшија сваком чвору, због тога је промењен само један фактор у односу на мању количину података, а то је број комшија који се узимају у обзир током тренинга модела, за сваки чвор, када би био дозвољен максималан број, вероватно никада не би дочекали извршење програма, због тога је број комшија у случају већег скупа смањен на [30, 20], што доводи до смањења перформанси, исти принцип је примењен на мањи скуп података и резултати су такође били за нијансу лошији, што такође објашњава ситуацију.

Коначни закључак јесте да САГЕ модел, 'bpr' функција губитака и 'lstm' класификатор дају најбоље резултате за дати проблем, тако да ће у наставку ови параметри бити коришћени за тренинг над свим скуповима података.

Валидација најефикаснијег модела

Након што смо одлучили који модел, са којим карактеристикама ће бити коришћен за наш систем препорука, можемо приступити испитивању тачности препорука и дубљој анализи добијених резултата за комбинацију САГЕ модела, 'bpr' функције губитака и 'lstm' класификатора.

Model	Classifier Type	Loss Function	Best Epoch	Best Validation Loss	Validation AUC	Accuracy for Positive Edges	Accuracy for Negative Edges	Precision	Recall	F1 Score
SAGEConv	lstm	bpr	56	0.194844	0.8640	0.837	0.895	0.888535	0.837	0.861998

Masked From	Masked To	Node From	Node To	Probability	True Label	Prediction
363	8	1888	8	0.99425	Positive	Positive
835	12	4329	12	0.83908	Positive	Positive
1127	31	5908	31	0.90526	Positive	Positive
1551	47	8228	47	0.61225	Positive	Positive
1761	62	9072	62	0.99095	Positive	Positive
896	12	4636	12	0.93621	Positive	Positive
169	43	798	43	0.00000	Negative	Negative
350	7	1765	7	0.00000	Negative	Negative
680	28	3611	28	0.00000	Negative	Negative
454	54	2365	54	0.56629	Negative	Positive
1817	51	9611	51	0.10254	Negative	Negative

Слика 61. Пример односа стварних веза и резултата предикције

На слици 61 можемо видети неке од предиктованих веза упоређене са стварним везама, приметно је да има и лоших и добрих предикција, али да су исправне предикције у великој већини, као што и саме евалуационе метрике кажу. Приметно је да када дође до погрешне предикције, обзиром на то да је метрика за избор >=0.5, након излаза из сигмоидне функције, углавном у питању вредност која је много већа од негативне границе којој треба да припада или много мања од позитивне границе којој треба да припада, што показује да модел у неким случајевима знатно греши, али и оправдава вредност постављене границе предикције.

Product ID P	Count_positive_real	Product ID N	Count_negative_real	Product ID P	Count_positive	Product ID N	Count_Negative
8.0	371.0	11	25	8.0	378.0	9	28
33.0	113.0	43	22	33.0	121.0	20	26
20.0	47.0	49	22	61.0	44.0	11	25
61.0	43.0	47	22	62.0	43.0	47	24
12.0	36.0	32	22	12.0	35.0	46	24
62.0	34.0	29	21	60.0	34.0	3	24
60.0	32.0	53	21	20.0	32.0	51	23
1.0	27.0	51	20	55.0	25.0	43	23
9.0	25.0	46	20	1.0	23.0	10	23
55.0	21.0	10	19	36.0	23.0	32	22

Слика 62. Популарност производа у реалним(лево) и предиктованим(десно) везама

Узорковане, случајне везе су балансираније од реалних, небалансираност реалних веза је намерно остављена због реалне ситуације где постоје производи који су заиста веома популарнији од других, међутим, наш задатак јесте предикција потенцијалних веза, која је одрађена одлично, већина најпопуларнијих производа је препоручена скоро исти број пута као и у реалној ситуацији, ово нам говори много тога о квалитету резултата модела, иако су најпопуларнији производи небалансирани у броју, модел није упао у замку различите популарности производа и није дао већу шансу популарнијим производима, него сасвим реалну, приметно је да су популарни производи углавном препоручени више пута него у реалној ситуацији, али тај број је занемарљив. Такође, то значи да су мање популарни производи добили шансу да буду препоручени одговарајућим корисницима, овим је доказано да **небалансираност броја препорука не утиче на предикције од стране GNN модела**. Ситуација је иста и код постојећих и узоркованих веза.

Упоредна анализа класичног КФ и GNN базираног система препорука

Као што је већ наведено у теоријском делу, колаборативно филтрирање у свом основном облику има доста недостатака, све методе касније креиране су покушаји да се ти недостаци надокнаде, GNN представљају једно од хибридних решења и у великој мери допуњују недостатке система за препоруке заснованих на КФ, проблеми које би требало прокоментарисати су: хладан почетак (Cold Start), скалабилност (Scalability), реткост (Sparsity), синонимија (Synonymy), сиве овце (Gray Sheep), црне овце (Black Sheep), Shilling напад (Shilling Attack), проклетство димензионалности (Curse of Dimensionality). За почетак, имплементиран је основни систем колаборативног филтрирања, над 10000 инстанци података, а затим и над 100000 инстанци, коришћени су подаци који су коришћени у имплементацији задатка са GNN, у наставку ће бити приказана и укратко објашњена имплементација, а затим ће бити описана предност GNN за сваки засебан проблем.

Подаци који се користе у овом случају се састоје само од 3 колоне, колоне о корисницима(рецензентима), производима и оцени коју су оставили корисници. То је по дефиницији костур сета података за КФ, на следећој слици биће приказане основне карактеристике.

rating	userId	productId
3.000	0	0
5.000	1	0
4.000	2	0
4.000	3	0
4.000	4	0

UserId	# ratings
9519	94
9420	80
9163	74
11012	64
10554	34
9403	30
9086	30
17449	28
8502	24
11937	24

```
Count of observations in each ratings:

5.000    70449
4.000    15219
3.000     5730
1.000     5288
2.000     3314
```

```
Count of unique Users     : 49336  Number of rows    : 100000
Count of unique Products : 773    Number of columns : 3
```

```
# unique USERS who have rated 3 or more products : 3211
# unique USERS dropped      : 46125
# unique ITEMS remaining    : 607
# unique ITEMS dropped      : 166

Final length of the dataset : 15721
```

```
Duplicates found
Shape of final_ratings_matrix          : (3211, 607)
Total observed ratings in the dataset  : 7461
Total ratings possible for the dataset : 1949077
Density of the dataset                 : 0.38%
```

Слика 63. Карактеристике података коришћених за КФ имплементацију

Приказане карактеристике представљају основне информације о сету података који је коришћен за имплементацију примера КФ, има око 50000 различитих корисника и око 800 производа, најзаступљеније оцене су 4 и 5, док су остале равномерно распоређене, што је и очекивано. Како би систем био иоле ефикасан, а заснива се на поседнутости реткопоседнуте матрице, потребно је обрадити га на тај начин што ће из скупа података бити избачени јединствени корисници који немају већи број конекција ка производима, у овом случају тај број је постављен на три, у реалним случајевима тај број се уобичајено поставља на 50, међутим, обзиром да наши корисници немају велики број конекција, изабран је број 3 како би могли да прикажемо рад оваквог система.

```
user_item = ratings.pivot(index='userId', columns='productId', values='rating').fillna(0)
print('Shape of User-Item sparse matrix:', user_item.shape)
user_item.head()
user_similarity = cosine_similarity(user_item)
np.fill_diagonal(user_similarity, 0)
user_similarity_df = pd.DataFrame(user_similarity,index=user_item.index,
        columns=user_item.index)
print("User similarity")
user_similarity_df.head()
```

Овај код служи за израду и анализу корисник-производ ретке матрице и сличности између корисника. user_item = ratings.pivot(index='userId', columns='productId', values='rating').fillna(0): Преводи почетни скуп података ratings у реткопоседнуту матрицу где редови представљају кориснике (userId), колоне представљају производе (productId), а вредности у матрици су оцене (rating) које корисници дају производима. Недостајуће вредности се попуњавају нулама(у нашем случају не постоје недостајуће вредности). Сличности између корисника су рачунате користећи косинусну сличност између корисника на основу реткопоседнуте матрице user_item. Креирана је нова матрица user_similarity која садржи сличности између свака два корисника. Након тога, дијагонални елементи матрице user_similarity су попуњени нулама, пошто је корисник увек сличан сам себи и такве сличности нису битне у анализи. Коначно, резултати

сличности су преведени у *DataFrame* облик под именом user_similarity_df, где су кориснички идентификатори додељени као индекси и колоне матрице, омогућавајући лакшу визуелну анализу и преглед сличности између корисника.

Shape of User-Item sparse matrix: (3211, 607)

productId	8	9	10	17	18	19	20	27	33	34	...	755	756	758	760	761	762	764	766	768	769
userId																					
997	3.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1785	5.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2265	5.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2269	5.000	5.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2397	4.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

5 rows × 607 columns

User similarity

userId	997	1785	2265	2269	2397	2446	3342	3440	3469	3698	...	48170	48276	48312	48319	48357	48371	48413	48414	48449	48566
userId																					
997	0.000	0.128	0.137	0.157	0.109	0.137	0.128	0.083	0.137	0.174	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1785	0.128	0.000	0.355	0.408	0.284	0.355	0.333	0.214	0.355	0.451	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2265	0.137	0.355	0.000	0.435	0.303	0.379	0.355	0.229	0.379	0.481	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2269	0.157	0.408	0.435	0.000	0.348	0.870	0.408	0.263	0.435	0.552	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2397	0.109	0.284	0.303	0.348	0.000	0.303	0.284	0.354	0.303	0.384	...	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

5 rows × 3211 columns

Слика 64. Матрица корисник-производ(горе), матрица сличности корисника(доле)

На основу ових матрица се рачунају предикције и вероватноћа постојања везе између одређених корисника и производа, за то су имплементиране следеће функције које на излазу требају да дају резултате у опсегу између 0 и 1, за шта ће бити коришћена сигмоидна функција, као и код GNN.

```
def predict(ratings, similarity, type='user'):
    mean_user_rating = ratings.mean(axis=1)
    ratings_diff = (ratings - mean_user_rating[:, np.newaxis])
    pred = mean_user_rating[:, np.newaxis] + similarity.dot(ratings_diff) /
np.array([np.abs(similarity).sum(axis=1)]).T
def predict_probability(ratings, similarity, type='user'):
    predicted_ratings = predict(ratings, similarity, type)
    # Apply the sigmoid function
    probability_pred = 1 / (1 + np.exp(-predicted_ratings))
    return probability_pred
def recommend_items(userId, preds_df, top_n):
    sorted_user_predictions = preds_df.loc[userId].sort_values(ascending=False)  #
                                                        sorted_user_predictions
    recommendations = sorted_user_predictions.rename_axis('Recommended
                                    Items').reset_index()
    recommendations.columns = ['Recommended Items', 'Predicted Rating']
    return recommendations.head(top_n)
```

Прва функција, predict, врши предвиђање рејтинга корисника за производе користећи информације о постојећим оценама и сличности између корисника. Ова функција помаже у оцени колико је вероватно да ће корисници оценити одређени производ. Друга функција, predict_probability, допуњује прву тако да предвиђа вероватноћу додељивања оцене. Ова вероватност се добија применом сигмоидне функције на предвиђени рејтинг. Ово је корисно када желимо да генеришемо

вероватноће конекција. Коначно, трећа функција, recommend_items, враћа најбољих N препорука за одређеног корисника на основу предвиђених рејтинга. Она сортира предвиђене рејтинге (или вероватноће) у опадајућем редоследу и враћа топ N производа које би корисници највероватније волели.

userId	Recommendation 1	Predicted Rating 1	Recommendation 2	Predicted Rating 2	Recommendation 3	Predicted Rating 3	Recommendation 4	Predicted Rating 4	Recommendation 5	Predicted Rating 5
11851.000	180.000	0.927	114.000	0.811	75.000	0.754	471.000	0.752	192.000	0.752
12440.000	129.000	0.989	104.000	0.679	436.000	0.564	151.000	0.555	241.000	0.547
9164.000	490.000	0.984	67.000	0.677	404.000	0.579	614.000	0.555	555.000	0.547
9259.000	597.000	0.887	449.000	0.842	69.000	0.701	555.000	0.611	511.000	0.611
11213.000	97.000	0.950	190.000	0.799	449.000	0.591	265.000	0.568	555.000	0.563
...	...	...	...	...	...	...	...	...	...	...
17494.000	511.000	0.930	265.000	0.791	614.000	0.640	513.000	0.609	508.000	0.578
8455.000	70.000	0.971	55.000	0.930	511.000	0.628	394.000	0.591	265.000	0.561
14884.000	514.000	0.944	215.000	0.732	496.000	0.724	511.000	0.616	508.000	0.594
10384.000	90.000	0.986	102.000	0.968	138.000	0.739	555.000	0.530	526.000	0.528
15224.000	286.000	0.981	219.000	0.677	508.000	0.614	511.000	0.605	761.000	0.578

Слика 65. Резултати препорука КФ система за препоруке

Ове функције су корисне компоненте у развоју система за препоруке, омогућавајући нам да генеришемо персонализоване препоруке за кориснике на основу њихових интеракција и сличности са другим корисницима, међутим имају много својих недостатака:

- Са малим бројем инстанци, тј. веза међу корисницима, је немогуће добити икакве резултате, зато је потребна велика количина података за функционисање оваквог система.
- Огроман број корисника уопште не добија шансу пре самог старта имплементације система(проблем хладног почетка), јер се велики број инстанци одбацује због малог броја веза, узмимо реалан пример, корисници који нису поручивали производе до сада, или корисници који су имали мало поруџбина нису циљна група оваквог система, што је огроман недостатак.
- Матрица сличности је реткопоседнута, што представља беспотребно коришћење огромне количине меморијског простора(реткост и *curse of dimensionality*).
- То што је корисник имао одређене конекције, не гарантује да је препорука исправна, добијени резултати показују да у већини случајева, стварне конекције које је корисник остварио, нису ни у топ 5 препорука од стране оваквог система.
- Систем захтева велики број инстанци, што аутоматски повећава димензије матрице повезаности, док то утиче на могућност евалуације, тј. што је већи број инстанци, то је шанса да хардверски ресурси неће моћи да обраде те податке.

```
MemoryError: Unable to allocate 17.9 GiB for an array with shape (49070, 49070) and data type float64
```

- Додавање нових елемената у евалуацију, поред оцена, би повећавало димензионалност задатка за једну димензију, што је математички веома скупа ситуација за рачунање и аутоматски захтева примену метода декомпозиције.
- Када се евалуирају вероватноће међу корисницима и производима, морају се избацити дупликати у везама између корисника и производа, што негативно утиче на систем, јер постоје корисници који купују исти производ више пута.
- Систем је подложан нападима инјекције веза и оцењивања у циљу популаризовања одређених производа.

Карактеристике-Проблеми	Колаборативно филтрирање	Графовске неуронске мреже
Потребан велики број конекција по кориснику	✓	✗
Проблем хладног почетка (Cold start problem)	✓	✗
Матрица сличности заузима велику количину меморије	✓	✗
Коришћење више од једног атрибута за предикцију	✗	✓
Могућност рада са разноврсним типовима података	✗	✓
Подешавање хиперпараметара	✗	✓
Проблем недостатка података	✓	✓
Скалабилност (Scalability)	✓	✗
Синонимија (Synonymy)	✓	✗
Сиве овце (Gray Sheep)	✓	✗
Црне овце (Black Sheep)	✓	✗
Shilling напад (Shilling Attack	✓	✗
Проклетство димензионалности (Curse of Dimensionality)	✓	✗
Сложеност модела	✗	✓
Временски захтеван проблем	✗	✓
Комплексност имплементације	✗	✓
Интерпретабилност(објашњивост резултата)	✓	✗

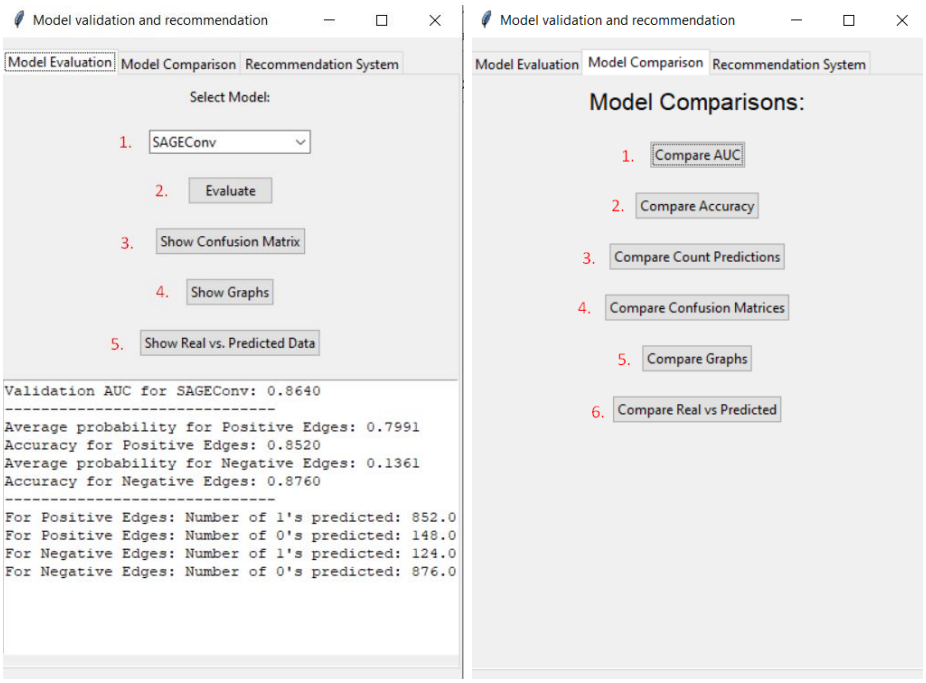
Табела 10. Карактеристике КФ у односу на ГНН

Демо систем персонализованих препорука

Након проналажења најбоље комбинације модела и параметара који иду уз њих, идеја је да се на визуелни начин прикажу добијени резултати и на крају представи систем персонализованих преЗпорука за кориснике који су у оптицају за дати скуп података. Уз помоћ Tkinter библиотеке представљена је демо апликација, где су спојене све методе за евалуацију, валидацију, упоредну анализу и персоналне препоруке. Функција коју је битно поменути за персонализоване препоруке јесте управо:

```
def get_user_recommendation_probabilities(model, user_id, sampled_data, device):
    model.eval()
    model.to(device)
    user_recommendations = []

    with torch.no_grad():
        item_ids = sampled_data['product'].node_id.to(device)

        # Generate user-item pairs for the specified user
        user_item_pairs = [(user_id, item_id.item()) for item_id in item_ids]

        for _, item_id in user_item_pairs:
            edge_label_index = torch.tensor([[user_id], [item_id]], device=device)

            # Create pseudo sampled_data with edge_label_index
            pseudo_sampled_data = sampled_data.clone()
            pseudo_sampled_data['reviewer', 'reviews', 'product'].edge_label_index =
                        edge_label_index

            pred = model(pseudo_sampled_data)
            prob = torch.sigmoid(pred).item()
            user_recommendations.append((item_id, prob))

    user_recommendations.sort(key=lambda x: x[1], reverse=True)
    return user_recommendations
```

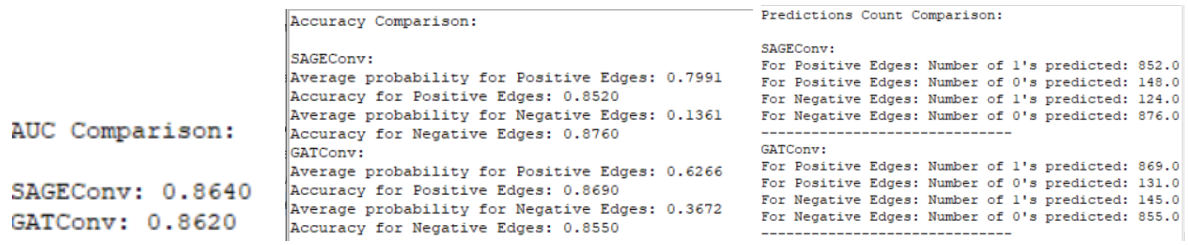
Идеја код ове функције јесте да се креира мини граф, са вим могућим везама од стране изабраног корисника ка производима, на тај начин ће бити добијене предикције за кориснике понаособ након примене већ истренираног модела, у односу на производе. Обзиром да модел ради са тензором парова корисник-производ прво је потребно креирати такав тензор, који ће представљати тензор ивица, тј. веза међу корисницима и производима, као и у претходном делу практичног рада, из тако добијених предикција од стране модела, се сортирају предикције по вероватноћи и прослеђују даље на употребу за персонализоване препоруке.



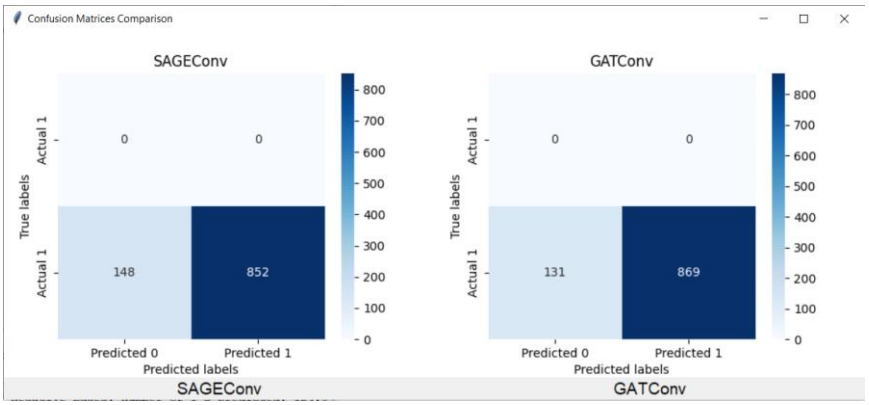
Слика 66. Прозор за евалуацију(лево), прозор за упоредну анализу модела(десно)

Покретањем апликације приказује се прозор са приказаним опцијама за евалуацију модела, поред опција за евалуацију, могу бити одабране још две опције из менија и то поређење модела и систем за персонализоване препоруке.

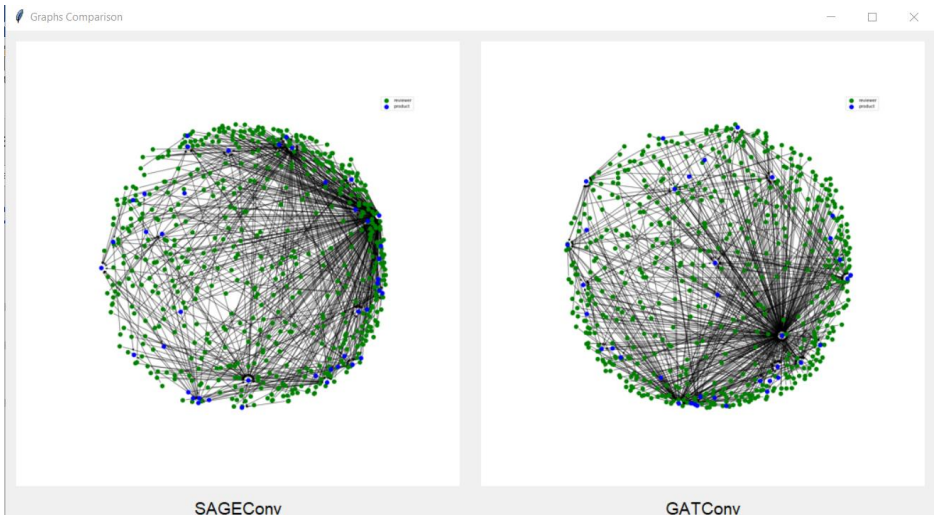
Први прозор (слика 66.(лево)) има опцију избора модела за који корисник жели да евалуира карактеристике (слика 66.(лево) - 1.), избор се своди на приказ резултата тачности предикције (слика 66. (лево) - 2.), приказ матрица конфузије (слика 66.(лево) - 3.), приказ стварног и предиктованог графа (слика 66.(лево) - 4.) и упоредне листе стварног и предвиђеног броја конекција по кориснику(слика 66.(лево) - 5.), за изабрани модел. Други прозор (слика 66.(десно)) садржи опције за упоредни приказ карактеристика евалуираних модела, поређење AUC скора, тачности предикција, броја предикција, матрице конфузије, добијених предиктованих графова и однос стварног и предиктованог броја конекција за јединствене кориснике, респективно. На следећим сликама биће приказан изглед бирања понуђених опција.



Слика 67. Евалуација тачности(поређење)



Слика 68. Матрице конфузије(поређење)



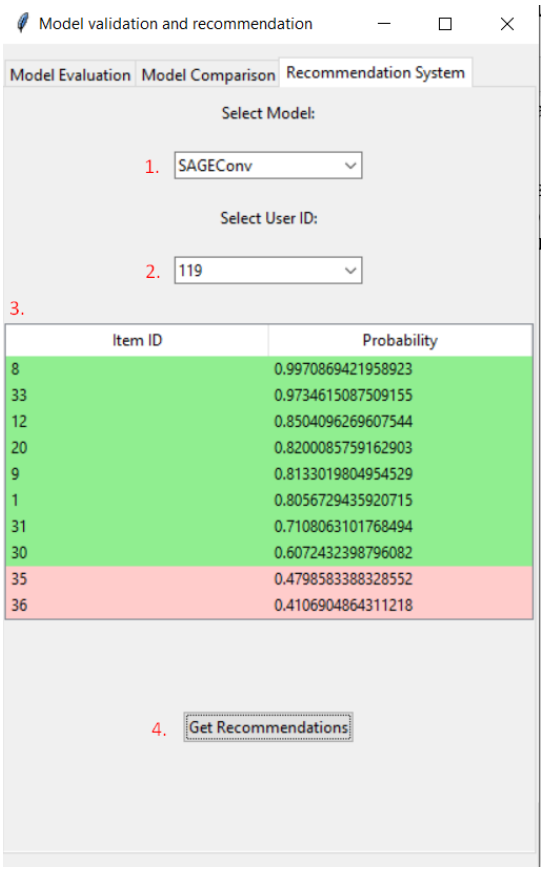
Слика 69. Валидациони графови(поређење)

Real vs. Predicted Data - All Models

User	Real Appearances	SAGEConv	GATConv
1103	2	1	2
654	2	0	1
1391	2	1	1
1439	2	2	2
791	2	2	2
1413	2	2	1
334	2	0	0
91	2	1	1
1763	2	2	2
634	2	2	2

Слика 70. Стваран и предвиђени број предикција по јединственом кориснику(поређење)

Након поређења, следи персонализована препорука производа изабраном кориснику, бира се један од корисника из валидационог скупа корисника, на основу његовог јединственог идентификатора, затим се израчунава вероватноће са којом ће производи из понуде бити понуђени одабраном кориснику.



Слика 71. Персонализована препорука

На одабраном прозору се може бирати модел који корисник жели да користи за предвиђање препорука (слика 71. - 1), такође корисник може изабрати кориснички идентификатор из понуде за кога жели да предвиди препоруке, или да једноставно укуца идентификатор (слика 71. - 2). У табели су приказане вредности вероватноћа које су предвиђене за одређене производе у опадајућем редоследу, црвена боја означава да производ не прелази дефинисани праг препоруке, а зелена производе који задовољавају праг (слика 71. - 3). Притиском на дугме се врши ново израчунавање (слика 71. - 4).

ЗАКЉУЧАК

Системи за препоруке имају дугу историју и овим проблемом се многе компаније баве већ деценијама, превасходно због тога што минимално побољшање овог система може довести до огромног повећања зараде. Из године у годину се представљају нова решења и нови приступи проблему, објављују се такмичења за креирање најбољег система над њиховим подацима, како би се дошло до скоро перфектних резултата, тренутно популаран приступ јесу системи за препоруке засновани на графовским неуронским мрежама и истраживања у овом правцу обећавају. Велике компаније још увек нису имплементирале ове системе, али већина то има у плану и увелико ради на интеграцији, што је још један показатељ растуће популарности и перспективе ових система.

Када је реч о системима за препоруке, незаобилазни појам јесте колаборативно филтрирање, представља основу рада и идеју у најпростијем облику оваквих система. Користи алгоритме за анализу корисничких рецензија у циљу креирања персонализованих препорука за кориснике са сличним интересима. Колаборативно филтрирање се дели на модел заснован на меморији, заснован на моделу и хибридни приступ, при чему се модел заснован на меморији даље дели на подмоделе засноване на производу и кориснику. Кључна разлика између меморијског и моделског приступа је у коришћењу параметара и алгоритама за оптимизацију, док меморијски приступ користи косинусну сличност и Пирсонове коефицијенте корелације за израчунавање сличности без учења параметара, моделски приступ укључује сложеније алгоритме факторизације матрице (PCA, SVD) и аналитичке процесе. Процес препоруке у системима заснованим на садржају обавља се кроз три корака: анализатор садржаја представља ставке у структурираном облику за даљу обраду, модул за учење профила гради кориснички профил заснован на корисничким преференцијама, а филтрирајућа компонента предвиђа интересовања корисника упоређујући карактеристике ставки са корисничким профилом, генеришући тако препоруке. Системи засновани на знању у евалуације укључују експертска знања и не одступају од чињеница које су наметнуте, деле се на системе засноване на ограничењима и случајевима. Хибридни модели комбинују карактеристике претходно наведених модела како би изградили што успешнији систем препоруке, ове системе користи већина компанија и циљ је да се искористи што више информација о кориснику и ставкама, један од ових система јесте свакако систем заснован на графовским неуронским мрежама.

Системи за препоруке засновани на графовским неуронским мрежама комбинују карактеристике различитих приступа препорукама, оно што овај приступ издваја од других јесте то што је графовска структура идеална када је у питању систем за препоруке, нема расипања меморије на непостојећим везама, док с друге стране сваки чвор у графу представља јединственог корисника или јединствен производ са свим њиховим карактеристикама, преференцијама и специфичностима, а слични корисници су удаљени свега два корака (гране) у графу у односу на корисника од интереса. У основи се налази филтрирање графа, тј. *message passing* механизам који омогућава комуникацију међу сличним чворовима и ажурирања параметара чворова, затим активациони слојеви и пулинг слојеви на излазима мрежа. *GraphSAGE* је просторно заснован филтер који генерише нове карактеристике за чворове путем агрегације информација из насумично одабраних суседних чворова, комбинујући ове информације са претходним карактеристикама чвора за формирање усавршених атрибута. GAT филтер представља револутивни филтер у графовским неуронским мрежама и користи популарни механизам

пажње за ажурирање карактеристика чворова у графу. Ова два филтера представљају *state-of-the-art* решења када су у питању системи за препоруке, тако да је у практичној имплементаци акценат управо на имплементацији модела са овим филтерима.

У практичном делу рада је имплементиран систем за препоруке заснован на графовским неуронским мрежама, са комплетним процесом реализације, од прикупљања и предобраде података, преко креирања самог графа од интереса, до тренинга модела графовске неуронске мреже и проналажења најефикснијег модела, добијени резултат је да *GraphSAGE* филтер у комбинацији са *BPR* (енг. Bayesian Personalized Ranking) функцијом губитака и *LSTM* (енг. Long Short-Term Memory) агрегатором информација даје најбоље резултате за предвиђање веза у графу креираном на основу Амазоновог скупа података предвиђеног за системе препорука. Алати без којих практични део пројекта не би могао бити реализован јесу програмски језик Python, PyTorch за рад са тензорским подацима, PyTorch Geometric за рад са графовима и пратеће библиотеке машинског и дубоког учења.

ЛИТЕРАТУРА

[1] D. Z. M. F. A. &. F. Jannach, Recommender systems: an introduction, Cambridge University Press, 2010.

[2] S. Cook, *Netflix statistics & facts that define the company's dominance in 2023,* 2023.

[3] eMarketer, *Average time spent on YouTube,* 2023.

[4] Paul, „Recommender Systems for Dummies," p. 8, 2023.

[5] P. Grover, *Various Implementations of Collaborative Filtering,* p. 20, 2017.

[6] M. Z. A. F. G. F. Dietmar Jannach, Recommender Systems, An Introduction, Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore,: Cambridge University Press, 2010.

[7] S. L. James Bennett, „The Netflix Prize," p. 4, 2007.

[8] L. R. B. S. Francesco Ricci, Recommender Systems Handbook, Springer, 2022.

[9] ,. M. B. Z. E. P. H. R. Q. T. Arjan Jeckmans, *Privacy in Recommender Systems*, p. 18, 2022.

[10] S. V. Vaishnavi M, *The Three-Tier Architecture of Federated* , p. 5, 2023.

[11] U. M. J.A. Bondy, Graph Theory, San Francisco: Springer, 2008.

[12] distill.pub, *A Gentle Introduction to Graph Neural Networks*, 2023.

[13] J. T. Yao Ma, Deep Learning on Graphs, Cambridge: Cambridge University Press, 2023.

[14] J. Zhou, *Graph neural networks: A review of methods and applications,* p. 26, 2020.

[15] P. Velickovic, *GRAPH ATTENTION NETWORKS,* p. 12, 2017.

[16] P. C. J. P. L. Z. Lingfei Wu, Graph Neural Networks: Foundations, Frontiers and Applications, Mountain View, CA, USA: Springer, 2022.

[17] P. L. J. A. G. Lemei Zhang, *Recommending on graphs: a comprehensive review from a,* p. 86, 2021.

[18] M. D. ,. X. Z. J. T. Zhen Yang, *Region or Global? A Principle for Negative,* t. 35, br. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2023.

[19] S. Rajamanickam, *Graph Neural Network (GNN) Architectures for Recommendation Systems,* 2021.

[20] Y. K. Yuliana Lavryk, *Product Recommendation System Using Graph Neural,* 2023.

[21] D. Li, *Recommender Systems with GNNs in PyG*, p. 24, 2022.

[22] V. Kumar, Concepts and Techniques of Graph Neural Networks, Dharmendra Singh Rajput, India: Koneru Lakshmaiah Education Foundation (Deemed), 2023.