МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ДАНИЛЕВИЧ Вадим Юрійович

Математичне та програмне забезпечення для виявлення та оцінки актуальності інформації в сервісно-орієнтованих системах / Mathematical Tools and Software for Identifying And Evaluating the Relevance of Information in Service-Oriented Systems

спеціальність: 121 - Інженерія програмного забезпечення
освітньо-професійна програма - Інженерія програмного забезпечення

Кваліфікаційна робота

Виконав студент групи ІПЗм-21
В. Ю. Данилевич
_____

Науковий керівник:
к.т.н., доцент А. М. Мельник
_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2022

ЗМІСТ

ВСТУП 3

Розділ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ 8

1.1. Поняття актуальності інформації та особливості її організації в сервісно-орієнтованих системах 8

1.2. Аналіз методів виявлення та оцінки актуальності інформації в сервісно-орієнтованих системах 15

1.3 Показники оцінки актуальності інформації 21

1.4. Постановка задачі дослідження 25

Висновки до розділу 1. 26

РОЗДІЛ 2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЯВЛЕННЯ ТА ОЦІНКИ АКТУАЛЬНОСТІ ІНФОРМАЦІЇ В СЕРВІСНО-ОРІЄНТОВАНИХ СИСТЕМАХ 29

2.1. Метрика оцінки актуальності інформації в сервісно-орієнтованих системах 29

2.2. Метод виявлення та оцінки актуальності інформації в сервісно-орієнтованих системах. 34

2.3. Алгоритм реалізації методу оцінки актуальності інформації 38

Висновки до розділу 2 41

РОЗДІЛ 3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЯВЛЕННЯ ТА ОЦІНКИ АКТУАЛЬНОСТІ ІНФОРМАЦІЇ В СЕРВІСНО-ОРІЄНТОВАНИХ СИСТЕМАХ 45

3.1. Особливості програмної реалізації системи для виявлення та оцінки актуальності інформації 45

3.2. Інструкція користувача та тестування системи 48

3.3. Експериментальні дослідження та оцінка ефективності методу виявлення та оцінки актуальності інформації 53

Висновки до розділу 3 59

ВИСНОВКИ 64

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 68

ДОДАТКИ ..... **Помилка! Закладку не визначено.**

Додаток А ..... **Помилка! Закладку не визначено.**

Лістинг програмного коду системи ..... **Помилка! Закладку не визначено.**

ВСТУП

Актуальність теми дослідження. Швидкість росту мережі Інтернет вражає, якщо в 2010 році в мережі було всього близько 250 млн. сайтів, то на даний момент – вже більше 2 млрд[11]. Інформаційна революція захоплює все нові сфери, і в найближчі роки цей тренд , як мінімум, залишиться на тому ж рівні. Однозначно, що ці мільярди інформаційних, банківських, наукових та розважальних сервісів генерують сотні терабайт даних кожну секунду, тому проблема їх обробки та взагалі роботи з ними неймовірно актуальна. Зрозуміло, що з такими масивами даних можуть працювати тільки інші сервіси чи програми. Саме така взаємодія може гарантувати адекватну обробку даних, оптимізувати використання оперативних ресурсів та гарантувати максимальний комфорт для користувача. Для такої взаємодії використовують API (від англ. **A**pplication **P**rogramming **I**nterface)[49] – по суті, функціональний механізм для зручного програмування. Абстрагуючи базову реалізацію певного сервісу API дозволяє проводити швидку та зручну інтеграцію сервісу в те, з чим зараз працює програміст, що прискорює його роботу, зменшує шанс помилки і дозволяє не дублювати вже написаний кимось функціонал.

Саме через різноманітні реалізації API(варіанти яких, особливості архітектур і найбільш популярні протоколи передачі даних будуть проаналізовані далі) і проходить обмін інформації між сервісами та програмними продуктами. Саме при використанні API найбільший шанс отримати неактуальну інформацію з програмного боку. Тепер опишемо поняття «актуальність інформації», та коротко розберемо види актуальності.

Актуальність інформації – це відповідність інформації, яку несе певний текст чи файл, і потреби користувача чи системи. В цифрових системах актуальність складається з двох компонентів – користувацької та програмної. Користувацька

актуальність – це точна відповідність потреб користувача та тієї інформації, яку він отримує. Будь які види фейків в цьому випадку – приклад неактуальної інформації. З програмного боку – будь-яка інформація, яка не відповідає потребам сервісу також є неактуальною. Будь-яка неактуальна інформація наносить серйозну шкоду інформаційним системам. Неактуальна інформація зменшує ефективність роботи на всіх рівнях сервісно-орієнтованої системи, а велика кількість фейкових новин погано впливає на читача, маніпулюючи його емоціями та сіючи недовіру до реальності. Тому, потрібна чітка можливість вимірювати актуальність інформації, та фільтрувати її, для того, щоб збільшити ефективність роботи сервіс-орієнтованих систем, та покращити рівень довіри читача до тієї інформації, яку він читає.

Об’єкт дослідження: поняття «актуальної інформації», «сервісно-орієнтованої системи», оцінка актуальності інформації, її методи, механізми та погляди на неї , проблеми які виникають через неактуальну інформацію в сервісно-орієнтованій системі.

Предмет дослідження: математично-програмний та теоретичний аналіз поняття «актуальності інформації» в сервісно-орієнтованій системі, вплив актуальності на ефективність системи .

Мета роботи : створення методики адекватної оцінки «актуальної інформації».

Завдання:

- Проаналізувати існуючу літературу щодо вивчення проблем актуальності інформації в сервісно-орієнтованій системі
- Систематизувати існуючий досвід по методах оцінки актуальності інформації
- Проаналізувати сучасні підходи до оптимізації актуальності.

- Описати практичний і теоретичний зміст експериментального дослідження проблеми актуальності інформації в сервісно-орієнтованій системі та розробити оптимальний метод оцінки актуальності, описати спосіб фільтрування даних
- Створити гнучку степ-метрику для оцінки актуальності довільної сервісно-орієнтованої системи
- Написати нейромережу, яка зможе відрізняти фейкові тексти від справжніх

Методи дослідження: Вирішення поставлених завдань здійснювалось шляхом використання комплексу загальнонаукових методів та наступних методів дослідження: 1) Теоретичних : аналіз, систематизація наукової літератури, синтез, класифікація, порівняння підходів та їхньої ефективності, тощо 2)загальноматематичних методів. 2) емпіричних : використання скрипів для парсингу, нейромереж, аналізаторів трафіку, тощо

Теоретико-методологічна основа дослідження : роботи вітчизняних та іноземних спеціалістів по темі мікросервісного стилю архітектури, сервісно-орієнтованих додатків, актуальності даних та методів фільтрування даних, нейромереж та машинного навчання.

Наукова новизна та практичне значення дослідження полягає в систематизації та аналізі існуючих поглядів/концепцій на актуальність інформації, їхній синтез з метою покращення та поглиблення теоретичних знань в даній темі, розробка метрики на основі степ-функції. Практичне значення полягає в створенні метода оцінки актуальності інформації та його системній реалізації, та в створенні методу фільтрування інформації з допомогою нейромереж.

Структура кваліфікаційної роботи: робота складається з 3 розділів, сумарно займає 67 сторінок, містить різноманітні списки. Дослідження проведено з

використанням 59 джерел, були залучені іноземні джерела (російськомовні підручники та англомовні підручники та німецькомовні підручники)

Розділ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1. Поняття актуальності інформації та особливості її організації в сервісно-орієнтованих системах

Як вже було сказано – в мережі Інтернет постійно проходить обмін між сервісами десятками ТБ інформації. І значна частина з них – неактуальні. Розбираючи неактуальні дані варто уточнити особливості видів саме неактуальності–

- «Неактуальність» як невідповідність часовим рамкам. В цю категорію потрапляє вся інформація, яка невчасно надходить, і тому не може адекватно виконувати свою функцію. Умовний приклад для цієї категорії – коли ви в бібліотеці попросили газету за вчора, а вам принесли за минулий рік
- «Неактуальність» як недовіра до джерела інформації. Під цю категорію підпадають всі можливі проблеми автентифікації та кібер-ризики підміни даних та інші фактори, коли інформації неможна довіряти. Умовний приклад цієї категорій – це стаття прочитана в «жовтій газеті», якій ви зовсім не довіряйте.
- «Неактуальність» як невідповідність запрошеної інформації до отриманої. Умовний приклад для цієї категорії, це коли в бібліотеці замість книг Гегеля вам приносять книги Гоголя.
- «Неактуальність» як невідповідність структури запрошеної до отриманої. Умовний приклад для цієї категорій, це коли в бібліотеці замість книги вам приносять рекламний буклет[4]

Уточнимо поділ неактуальної інформації на дві сфери та формалізуємо цей поділ –

- Програміст/сервіс або програмна сторона. Це тип неактуальної інформації, яку отримує або сервіс, і не може провести адекватну обробку(зазвичай – ігнорує цю інформацію, або, що куди гірше – опрацьовує далі по принципу «як є») або програміст, який менше вразливий до неактуальної інформації, але не може ігнорувати її, в результаті зменшується ефективність роботи, потрібно писати додаткові тести, оптимізувати код та інше, а якщо неактуальні дані не помічені програмістом – то можна пропустити її далі, що буде зменшувати доступність сервісу, може привести до серйозних помилок і погіршувати стан сервісу.
- Користувач. Зазвичай, користувач захищений від більшості неактуальної інформації, але, тим не менше, він може її отримувати як в форматі статей, так і в форматі відгуків чи відповідей, що створені з допомогою нейромереж спамерами, окремим питанням стоїть «фейкова» інформація, надалі тезово будуть описані способи фільтрування/боротьби з фейковою інформацією.[5]

Тобто, проблема неактуальної інформації комплексна, і потребує рішення відразу в обох сферах, хоч і сервісна проблема є більш важливою, а читацька сторона питання – більш універсальною, тому що робота з широким спектром текстів (медичні, наукові, рекламні, політичні, новинні, комедійні та інше) потребує універсальних рішень, так як існує широкий вибір форматів, а не як в сервісно-програмній стороні проблеми, де більшість інформації так чи інакше структурована.

Говорячи при фейкові новини, потрібно більше розповісти про негативні явища в соціумі, які вони створюють, серед них можна виділити:

- Ріст конспірологічних теорій. Через фейкові новини до читача часто доносять конспірологічні теорії, які не мають нічого спільного з

реальністю, як то історія про «хімтрейли», «чіпи в вакцинах» чи інші псевдонаукові теорії, які негативно впливають на аудиторію.

- Розпалення ненависті. Часто фейкові новини мають чітку емоційну забарвленість та підводять читача до емоційних рішень, які ґрунтовані на ненависті та відчутті власної беззахисності. Зрозуміло, що це дуже погано впливає на здоров'я соціуму.
- Просування ворожого наративу. Цей фактор особливо важливий в контексті гібридних та звичайних війн, завдяки фейковим новинам та ехо-ефекту ворожі пропагандисти можуть дешево і швидко просувати свій наратив, а повторюючи його досить часто – і змушують прийняти його певну частину населення, що може дуже негативно вплинути на існування певної держави.

Підсумуючи це, можна сказати, що боротьба з фейковими новинами необхідна, а для цього повинні бути дієві метрики оцінки інформації та програмні продукти, які можуть швидко знаходити неактуальну інформацію.

Щоб краще зрозуміти, як оцінювати програмну сторону актуальності даних потрібно проаналізувати особливості роботи та побудови сервісно-орієнтованої системи.

Сервісно-орієнтована система – це система яка побудована на базі сервісно-орієнтованої архітектури

Сервісно-орієнтована архітектура(service-oriented architecture, SOA) – стиль архітектури системи, що засновується на стилі мікросервісів, дещо поглиблюючи і розширюючи їх. Коротко виділимо вимоги до реалізації, які точно описують сервісно-орієнтовану архітектуру:

- Інтеграцію користувача. Повинен бути чіткий інтерфейс, який дозволяє взаємодіяти конкретному користувачу з сервіс-орієнтованою системою.

- Поєднування додатків. Як і в мікросервісній архітектурі – кожен процес має бути виділений в окремий чорний ящик, які «спілкуються» між собою з допомогою API.
- Інтеграцію процесів. Це додаткова вимога, яка випливає з мінусів мікросервісного стилю архітектури. Бізнес-процеси мають сприйматись як єдина структура, навіть якщо вони роз'єднані між різними мікросервісами, оптимізувати бізнес-процеси потрібно як єдину структуру, не розбивати їх. Для цього варто виділити окрему команду.
- Інформаційну інтеграцію. Інформаційна інтеграція – це вимога, яка включає в себе єдині, уніфіковані правила та комплексний підхід для роботи з інформацією.

Сервісно-орієнтована архітектура дещо відрізняється від мікросервісної, а саме – якщо мікросервісна краще пристосована до безперервної поставки системного продукту, так як розробники мікросервісної можуть в будь-який момент часу видати новий реліз, також, варто відзначити, що мікросервісна архітектура все ж створює єдиний додаток, коли сервісно-орієнтована архітектура створює множину застосунків, які взаємодіють між собою, що дозволяє створювати цілі простори, які наповнені застосунками, що працюють в режимі оркестру.

В жовтні 2009 року був виданий маніфест сервісно-орієнтованої архітектури, це основний документ і пам'ятка для усіх системних архітекторів, тому варто його розібрати, а саме – 6 його принципів.

1. Ділова цінність важливіша чим технічна стратегія. Принцип, який ґрунтується на правильній цілі – сервіс створюється для певної справи чи діла. Саме виконання цієї справи важливіше, чим технічна стратегія, вона має підлаштовуватись під ділову цінність.

2. Стратегічна ціль важливіша, чим вигода від конкретного проекту. Цей принцип частково виходить з попереднього. Навіть якщо певний конкретний проект чи реалізація стали вигідними – не варто забувати про стратегічну ціль, і тримати її в фокусі.
3. Внутрішня сумісність важливіша, чим інтеграція користувача. Без внутрішньої сумісності не побудувати сервісно-орієнтовану архітектуру, а без цього немає ніякого змісту інтеграція користувача. Тому спочатку всі ресурси мають бути кинуті на створення хорошої і сумісної архітектури, а потім вже повинна бути реалізована інтеграція користувача.
4. Спільні служби важливіші, ніж конкретна реалізація під призначення. Потрібно не забувати про правила ООП, і створювати максимально широкі реалізації, тому що конкретна реалізація може зменшити можливості системи.
5. Гнучкість важливіша чим оптимізація. Складовий сервіс має бути максимально гнучкий, і повністю покривати свою задачу, навіть якщо це призведе до більшого часу виконання сервісу. Оптимізація повинна бути, але не в шкоду гнучкості.
6. Еволюційне вдосконалення набагато важливіше, чим спроба побудувати початкову досконалість. Вимоги до системи часто міняються, і хоч це і важко, з цим потрібно миритись і жити. Тому, спроба побудувати досконалість з самого початку приречена на провал після першого коректування бізнес-цілей, саме тому потрібно не боятись еволюційний змін з боку системи, направляти їх та вдосконалювати, щоб в кожен момент часу система відповідала всім нагальним вимогам і потребам, а не

Саме еволюційність і дозволяє вводити змінити в питанні актуальності інформації на обох рівнях - в першу чергу проводити вимірювання актуальності інформації на першому і другому рівні, а потім використовувати

фільтрування інформації для того, щоб отримати кращий результат і збільшити можливості системи.

Опишемо проблеми з актуальністю, які можуть виникати з першим та другим компонентові згідно отримувача інформації, для цього знову ж таки опишемо проблеми з боку програмного, і з боку користувача.

Більше зупинимось на проблемах, які виникають перед користувачем сервісу[2], так як вони більш універсальні, і основна ціль цієї роботи – створити універсальні інструменти для фільтрації. Його проблеми можна поділити на декілька напрямів:

- Проблеми з відгуками. Неактуальність і невідповідність відгукам – дуже серйозна проблема, так як дуже часто саме відгуки дозволяють користувачеві зробити той чи інший вибір на рахунок покупки певного товару чи послуги. Якщо користувач не довіряє відгукам, або бачить в списку відгуків чітко фейкові відгуки, то він з меншою ймовірністю купить певний товар, що, звичайно, дуже важливо, коли ми говоримо про бізнес-цілі. Канадський спеціаліст по даних [4] Хедер Ахмед виділяє три види проблем з відгуками: По-перше, це неправдиві відгуки, які пишуть для того, щоб надати неправдиву інформацію про продукт та підвищити його репутацію або завдати йому шкоди. По-друге, це огляди, націлені на бренд, але які не виражають враження від цільового продукту. Третя група — не-рецензії та реклама, що містить лише текст, який дуже опосередковано пов'язаний з продуктом. Групи 3 і 2 відносно легко ідентифікувати, тоді як першу трохи проблематично. Ці відгуки пише або один спамер, найнятий власником бізнесу, або група спамерів, які працюють разом в певне часове вікно для маніпулювання репутацією продукту або магазину. Відгуки сильно впливають на бізнес-ефективність будь-якого сервісу, тому боротьба та збільшення актуальності відгуків

зможе значно збільшити прибутки будь-якого інтернет-магазину, так як відгуки справді будуть описувати реальність певного продукту, а не будуть певною компіляцією роботи різного ПЗ для спаму та окремих груп спамерів.

- Проблеми з статтями та матеріалами. Проблема з фейками та «жовтими» публікаціями дуже актуальна на даний момент, так як кількість фейків у всьому світі росте значними темпами. І якщо фейкові відгуки погано впливають на бізнес, то фейкові новини неймовірно шкідливі для усіх сфер людської діяльності – вони створюють і підтримують міфи, погано впливають на політичну ситуацію в країні, використовуються для ведення інформаційного вторгнення в інші країни, та призводить до росту недовіри у відношенні до офіційних ЗМІ. Згідно дослідження Кай Шу [4] фейкову інформацію можна поділити на три групи - неточну інформацію (це інформація з неточностями, міські легенди, міфи та інше), дезінформацію (це фейкові новини і новини, які спеціально роблять для того, щоб ввести аудиторію в оману чи викликати в них певну психологічну реакцію), та хейт-спіч, який пишуть і викладають саме для того, щоб викликати емоції – це емоційні статті, харрасмент[6] та інші психологічні матеріали, які використовують для того, щоб посіяти ворожнечу та маніпулювати суспільними думками.

1.2. Аналіз методів виявлення та оцінки актуальності інформації в сервісно-орієнтованих системах

Для того, щоб краще зрозуміти як неактуальна інформація розходиться по сервісно-орієнтованих системах потрібно описати методи обміну інформації між розподіленими системами. В основному, це два головних методи реалізації API - SOAP і REST(являє собою підхід до HTTP).

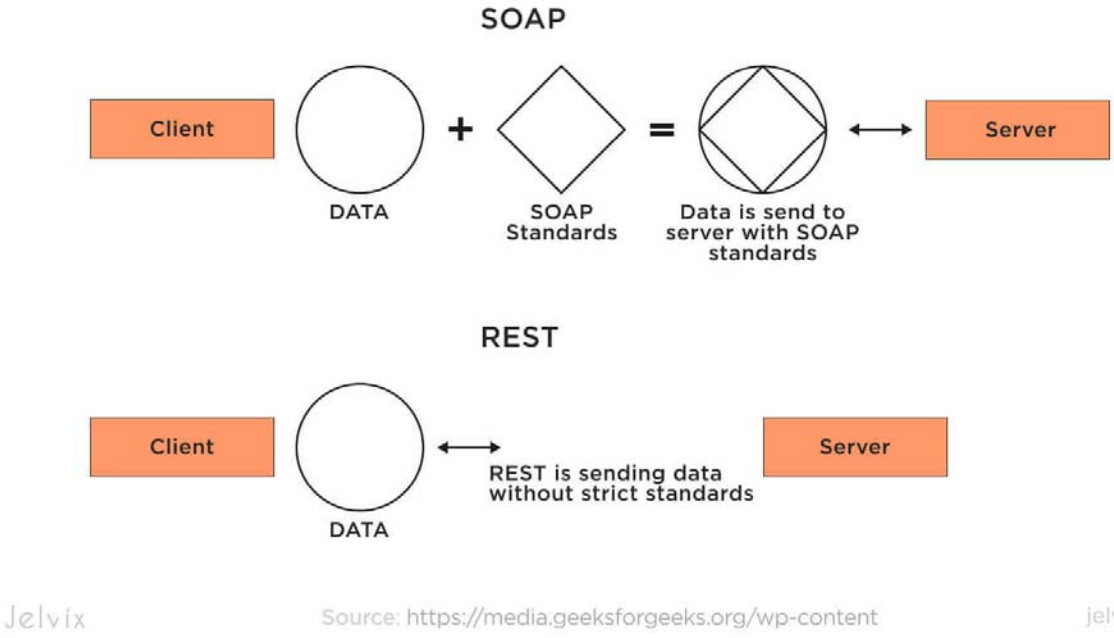


Рис. 1.1. Принцип стуктурування інформації з допомогою SOAP та REST

SOAP (**S**imple **O**bject **A**ccess **P**rotocol) – це структурований протокол обміну повідомленнями в розподіленій веб-системі[48]. Він заснований на форматі XLM. Так як це протокол, його можна використовувати з будь-яким комунікаційним протоколом прикладного рівня. Звичайно, найчастіше його використовують з HTTP. На рисунку 3 можна подивитись на структуру, яку

передбачає даний протокол обміну даними.

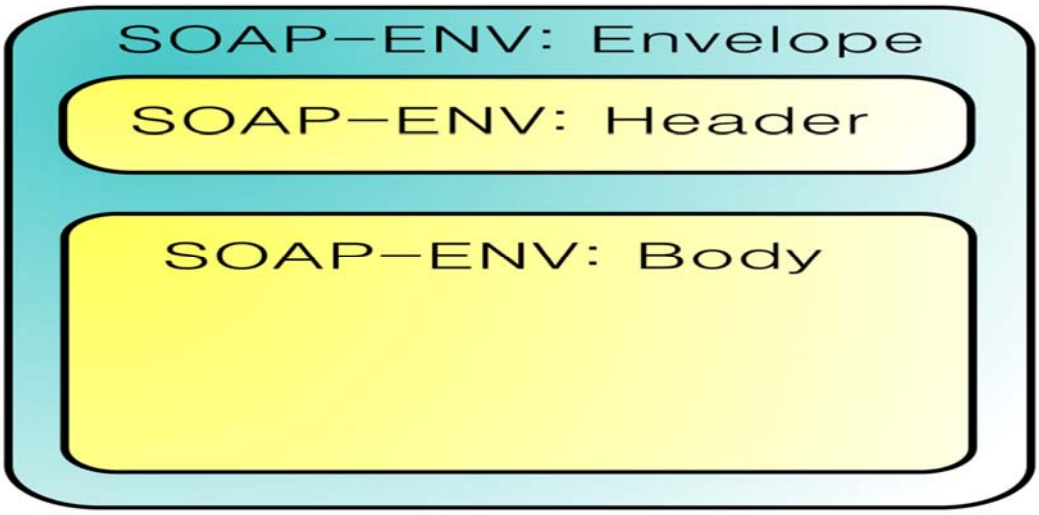


Рис. 1.2. Структура повідомлення SOAP

Серед плюсів SOAP можна виділити значну гнучкість (його можна використовувати з будь-яким протоколом передачі даних, як було сказано вище. Можна навіть використовувати JMS чи SMTP. Також великим плюсом є те, що з допомогою даного протоколу легко встановлювати з'єднання на базі існуючих фаєрволів, без додаткової модифікації протоколу, що максимально зручно та комфортно і для програміста, і дозволяє не плодити лишні сутності, створюючи десятки реалізацій одного й того ж протоколу.

Але є і певні недоліки, і в першу чергу це збільшення розміру повідомлення через використання протоколу. Звичайно, з збільшенням розміру повідомлення значно падає швидкість обробки, що може бути критично важливо в багатьох системах, де швидкість важлива, або де є критичне використання можливостей сервера. Також, не дивлячись на стандартність SOAP, різні системи розробки часто генерують несумісні повідомлення, це швидше всього, пов'язано з проблемами форматування. Про це варто пам'ятати, та уніфіковувати сервера та клієнти.

REST (**Re**presentational **S**tate **T**ransfer) – це архітектурний підхід до розробки мережевих протоколів, що надають доступ до інформації[44]. Як кожен архітектурний стиль він має певні обмеження, які потрібно описати. Однією з основних вимог є так звана «відсутність стану», яка ґрунтується на ідеї, що сервер нічого не зберігає з попереднього запиту, відтак кожен запит має включати в себе все необхідне для його обробки. Ця вимога призводить до незалежності кожного запиту, тому, наприклад, стан сесії зберігається на стороні клієнта і передається заново з кожним наступним запитом до сервера. Ця вимога має декілька значних плюсів – зменшується ціна помилки в запиті, так як помилковий запит – «річ в собі» і він ніяк не впливає інші. Також набагато простіший пошук помилок і тестування – так як потрібно аналізувати тільки один конкретний пакет даних, а не цілий ланцюг запитів, в кожному з яких може бути помилка. Основним мінусом є зниження продуктивності, так як серверу потрібно постійно обробляти додаткову інформацію, яка передається в кожному запиті окремо – якщо в сервісі немає сильної потреби в швидкості, то цю вимогу обов'язково потрібно виконувати. Друга вимога – це запит додаткового коду. Клієнти повинні мати можливість розширити функціональність з допомогою скриптів, але це необов'язкова вимога, і якщо запит додаткового коду може погано вплинути на безпеку, від неї варто буде відмовитись. Також серед вимог є обов'язкове використання шарів абстракції. Воно сильно перекликається з філософією мікросервісів, тому не будемо сильно на ньому зосереджуватись – кожен шар має виконувати чітку функцію та зв'язуватись з шаром більш високого рівня. Ще одна важлива вимога – однорідність інтерфейсу. Однорідність інтерфейсу дозволяє значно зменшити рівень зв'язності мікросервіса та клієнта. Останньою обов'язковою вимогою є кешування. Кешування дозволяє значно збільшити ефективність взаємодії між

сервісом та клієнтом. На рисунку 4 схематично зображена робота REST API

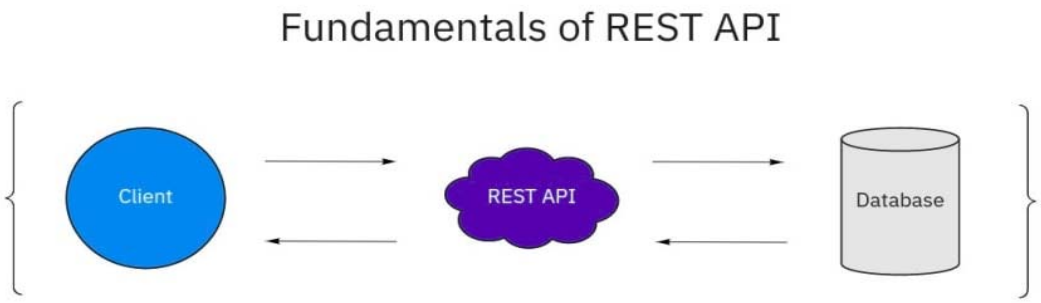


Рис 1.3. Передача інформації з допомогою API

Знаючи та розуміючи структуру даних, яка використовується в API можна набагато простіше відокремлювати неактуальні дані, писати парсерси для аналізу трафіку та готувати датасет для нейронної мережі, щоб фільтрувати інформацію.

Для правильної оцінки потрібно виділити патерни, які характерні для програмної неактуальної інформації, та для неактуальної інформації, яка потрапляє користувачеві.

Випадки неактуальної інформації в програмній сфері, це, в першу чергу неструктурований опис даних, який викликаний тим, що системи не надають опису структур даних, також багато проблем створює різноманіття програмно-технічних специфікацій, що призводить до різної семантики при інтерпретації запитів, і до серйозних проблем з взаємодією, також серйозна проблема – нерелевантне доменне ім’я або відсутність верифікації довіреної DNS зони, це призводить до відсутності способу визначення достовірності джерела, і викликає значну кількість проблем з безпекою. Також проблему викликає

неможливість отримати контекст з сервісу, що призводить до відсутності API для оцінки якості.

Якщо ми говоримо про інформацію, що отримує користувач – тут все набагато важче, тому що відділити однозначно фейкову новину від справжньої дуже важко, і практично неможливо без використання механізмів машинного навчання та нейронних мереж. Звичайно, можна виділити певні метрики, наприклад, досить примітивне співвідношення $\frac{\text{Кемоц}}{\text{Квсього}}$, де Кемоц – кількість емоційних слів в певній новині, а Квсього – кількість слів всього. В результаті алгоритм вимірювання даного співвідношення дуже простий.

1 Кемоц = 0, Квсього = 0
2 Створення словника емоційних стоп-слів, наприклад = [«неймовірно», «катастрофічно», «страшно»]
3 Лемматизація цільового тексту.
4 Перебір кожного слова в тексті.
5 Якщо слово з словника є в тексті, то Кемоц = Кемоц + 1
6 Квсього = Квсього + 1
7 Кінець циклу
8 Вивід співвідношення Кемоц/Квсього

Звичайно, ця примітивна метрика не може видавати серйозні результати, але вона приблизно показує, на чому ґрунтуються всі методи оцінки фейковості новин. В подальшому ця несерйозна метрика допоможе краще зрозуміти патерни, які буде знаходити нейромережа.

Метрик по відношенню до програмної актуальності даних є багато, особливо варто виділити метрику, запропоновану А. М. Мельником, М. П. Диваком, Р. М. Пасічником[1], вона являє собою формулу для оцінки інтерпритованості та

власне метрику для оцінки. Інтерпритованість(Ip) = $\frac{\sum \text{estimation}(\text{type}_i) * v_i}{\sum v_i}$, де type – типи показників інформації, що включають в себе наступні типи :

- P(Патерн) – сервіси та схеми їх використання. В цей тип потрапляє уся інформація про архітектуру системи. В першу чергу в цей тип потрапляє все, що описано в **WSDL** (**W**eb **S**ervices **D**escription **L**anguage) – мові описання веб-серверів, яка грунтується на XML, кожен документ на цій мові складається з чотирьох основних логічних компонентів – по-перше, це визначення типу даних, в якому заключається вид відправлених і отриманих сервісом повідомлень, по-друге, це елементи даних – сюди потрапляють повідомлення, що застосовуються з веб-сервісом, по-третє це абстрактні операції, в якому вказані усі типи операцій, що можуть бути виконані над повідомленням і останній – зв'язування сервісів – з допомогою якого описують спосіб, який даний сервіс доставляє повідомлення між адресатами, також сюди можна віднести усі додаткові дані – моделі даних, що використовує сервіс, умови відповідей та запитів та усі інші нюанси, які несуть інформацію про архітектуру і особливості побудови системи.
- D(Документація) – це опис API та вся додаткова інформація від розробника, яку він надав для кращого розуміння роботи інтерфейсу
- Nh(Нефункціональні властивості) – це різноманітні оцінки якості роботи сервіса, локація сервісу, локалізація та відповідність мови, доступність, час відповіді, рівень надійності, та інші оцінкові характеристики, яким можна довіряти в справі оцінки ресурсу, як то вимірювання актуальності відповіді з допомогою іншої метрики чи просто швидкість відповіді сервіса.
- Fd(Першоджерело) – тут опис версії патерну

- Rs(Достовірність джерела) – сюди вносять дані пов’язані з безпекою, визначають, чи в довіреній доменній зоні знаходиться сервіс, чи проходив програмний продукт реєстрацію в сертифікованому центрі та інші способи визначення рівня безпеки/небезпеки сайту.

Надаючи різні вагові коефіцієнти кожному з компонентів можна точно оцінювати актуальність певного сервісу, а сама вагова модель дозволяє концентрувати увагу на тому, що важливо саме для того розробника чи команди розробників, і над якими проблемами вони зараз працюють.

1.3 Показники оцінки актуальності інформації

Для того, щоб оцінити актуальність програмної інформації в виділеній системі використовують різноманітні метрики. Для того, щоб краще зрозуміти метрики варто ознайомитись з ключовою теоремою, яка описує фундаментальну проблему розподілених систем. Річ йде про теорему CAP. Теорема Брюєра, яка також відома як теорема CAP (**C**onsistency, **A**vailability, **P**artition tolerance) – ключове твердження, яка постулює, що для будь-якої розподіленої системи не можна досягти одночасно узгодженості даних, доступності даних та стійкості даних. Узгодженість – це характеристика, яка заключається в тому, що всі вузли розподіленої системи в будь-який момент часу бачать однакові дані, доступність – характеристика, яка гарантує те, що кожен запит отримає достатню та адекватну відповідь), та остання характеристика – стійкість, це властивість, яка гарантує, що при втраті зв’язку або при ізоляції цілих секторів системи вона зможе в подальшому працювати і зберігати здатність правильно відповідати на запити.

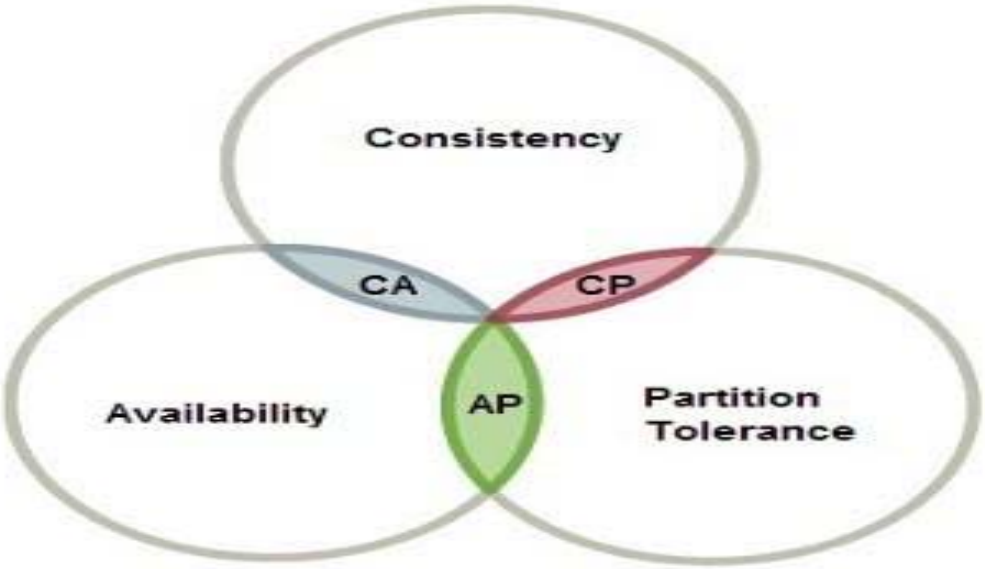


Рис. 1.4. Візуалізація теореми CAP.

Тому, згідно теореми можна досягти тільки двох з даних вище цілей. Тому, можна виділити три типи мереж, які відповідають теоремі CAP – CA,AP,CP.

- CA, або узгодженість-доступність. Це система, в якій не забезпечується стійкість до розділення. Хороший приклад такої системи – реляційні бази даних.
- AP, або доступність-стійкість. Це система, в якій ігнорується цілісність результату, але гарантується стійкість та доступність. Це, в першу чергу. DNS.
- CP, або узгодженість-стійкість. Система, що забезпечує цілісність всіх даних на вузлах, і стійкість, але не гарантує доступність. Хороший приклад – це банківські системи.

Тобто, не у всіх сервісах можна гарантувати узгодженість інформації, а як було описано вище, саме узгодженість є одним з основних факторів, що підтримують актуальність даних в сервісно-орієнтованих системах.

Тепер опишемо метричні методи оцінки. Метрики умовно можна поділити на два типи – степ-метрики та метрики засновані на дійсних числах. Степ-метрики ґрунтуються на певних умовах та легко описуються з допомогою блок-схем. Метрики засновані на дійсних числах реалізуються через звичайні функції, які отримують деякі значення на вхід, та виводять дійсне число, наприклад, в інтервалі від [0,1]. Перевести степ-метрику в дійсне значення дуже просто по формулі $\frac{\text{Значення\_метрики}}{\text{Максимум\_метрики}}$. Тому степ-метрики та дійсні метрики зводимі та можуть легко зрівнюватись в дійсних числах.

Опишемо основні показники, які використовують в роботі з натуральною мовою.

TF — це підхід, який ґрунтується на підрахунку слів, що знаходяться в документах, щоб визначити степінь подібності між документами.

Кожен документ представлено вектором однакової довжини, який містить кількість слів. Далі кожен вектор певним чином нормалізується, щоб його елементи нагадували бінарний класифікатор. Потім кількість кожного слова в тексті перетворюється на ймовірність існування такого слова в документі. Наприклад, якщо слово є в певному документі, воно буде представлено як 1, а якщо його немає в документі, воно буде встановлено на 0. Таким чином, кожен документ представлено групами слів.

Нехай D позначає певний корпус чи набір документів. Нехай d позначає конкретний документ, який ми визначаємо як певний набір слів w. $d \in D$. Позначимо як nw(d) кількість разів, коли слово w з'явиться в документі d, отже розмір документу d можна знайти наступним чином: $|d| = \sum w \in Dnw(d)$

Нормалізований TF для слова w щодо документу d визначається таким чином:

$$TF(w)_d = \frac{nw(d)}{|d|}$$

TF-IDF – це вагова метрика, яка часто використовується в інформаційному пошуку та обробці натуральної мови. Також, це важливий статистичний показник, який часто використовується для вимірювання важливості терміну відносно до тексту. Важливість терміну зростає з збільшення кількості разів, коли слово з'являється в документу, але цьому протидіє частота слова в корпусі (корпус – це набір текстів, які ми обробляємо)

Інверсна частина документу (IDF), для певного терміну w щодо корпусу документів D, що позначена IDF(w)D, є логарифмом загальної кількості документів корпусу, поділеної на кількість документів, у яких зустрічається даний конкретний термін, обчислюється наступним чином:

$$\mathrm{IDF(w)D} = 1 + \log\left(\frac{|\mathrm{D}|}{\left|\mathrm{d}: \mathrm{D}|\mathrm{w} \in \mathrm{d}|\right|}\right)$$

Однією з головних характеристик IDF є те, що він зменшує вагу стоп-слів в TF, одночасно збільшуючи вагу рідкісних слів. Наприклад, такі слова як «або» і «і» часто з'являються в тексті, і якщо ми використовуємо лише TF, такі терміни домінуватимуть у кількості частот.

Однак використання шкал IDF зменшує вплив цих термінів.

TF-IDF для слова w щодо документа d і корпусу D обчислюється наступним чином:

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

Це максимально формальне описання цих метрик. Для кращого розуміння потрібно навести приклад.

Отже, для прикладу візьмемо, що у нас є документ зі 200 слів, і нам потрібно визначити TF-IDF для слова «коти», допустимо, що це слово зустрічається в документі 5 разів, тоді TF = 5/200 = 0.025.

Тепер обчислимо IDF(коти), допустимо, що в нас є корпус з 500 текстів, а слово «коти» вказано в 100 із них, тоді IDF(коти) = 1 + log(500/100) = 1.69. Маючи ці дані ми можемо легко вирахувати TF-IDF(коти) = 0.025 * 1.69 = 0.04056.

Дані метрики дуже зручні для аналізу великих корпусів текстів, тому що обробка корпусу з допомогою машинних методів навчання може займати дуже багато часу (наприклад, видалення стоп-слів, нормалізація, лемматизація – дуже вимогливі задачі по відношенню до машинних потужностей, а що буде, якщо в корпусі багато тисяч текстів?), тому метрики дозволяють швидко оцінити основні показники тексту та їх легко реалізувати апаратно, написавши простенький скріпт, який швидко вирахує ці показники для усього корпусу текстів.

1.4. Постановка задачі дослідження

Провести аналіз існуючих метрик для оцінки актуальності програмних даних. Проаналізувати існуючі досягнення в сфері машинного навчання та нейронних мереж для того, щоб відрізняти фейкові новини від справжніх. Проаналізувати історичне становлення методів машинного навчання для рішення цієї задачі. Виділити основні дослідження, описати їх точність.

На основі існуючих метрик вивести оптимальну метрику для оцінки програмних даних.

Створити декілька степ-метрик для оцінки рівня «фейковості» новин та програмної сторони сервісно-орієнтованої архітектури, обґрунтувати її з точки зору математики та теорії ймовірності.

Описати основні характеристики нейромереж, в чому вони заключаються та як вони впливають на роботу нейронної мережі.

Створити програмний продукт, який заснований на нейронних мережах, що зможе з великою точністю відділяти фейкові новини від справжніх, та може використовуватись для фільтрації фейкових новин, що значно допоможе сервісам працювати з новинами. Змінювати архітектуру нейромережі та активаційну функцію, описати зміну точності та описати їх, знайти оптимальну архітектуру, оптимальне значення дропауту, оптимальну кількість нейронів та шарів нейронів для рішення конкретної задачі.

Спробувати повторити або покращити існуючі досягнення в сфері нейронних мереж для вирішення задачі виділення фейкових новин від справжніх.

З допомогою парсингу новинних сайтів створити датасет, що буде складатись з фейкових нових та справжніх новин, сформувати його в .csv форматі.

Провести лінгвістичний аналіз української мови для пошуку ключових стоп-слів, та сформувати з них файл для подальшої обробки датасету.

Висновки до розділу 1.

Актуальність інформації – інтегральний показник, який формується з багатьох факторів. Можна виділити дві основних складових інтегральної актуальності – програмна актуальність, яка характеризується правильними форматами даних, структурованою інформацією, відповідністю сертифікатів та загальною кібербезпекою, хорошим рівнем реалізації API, та яка важлива для програміста та системи в цілому, і користувацька актуальність, яка зав’язана на боротьбі з фейками та точній і вчасній інформації, яку отримує користувач. Досягнення актуальності з програмного боку – важко, так як гарантувати

узгодженість не завжди можливо, але, через апаратну природу з нею простіше працювати, можна досить об’єктивно вимірювати та оцінювати. Актуальність з боку користувача набагато гірше піддається вимірюванню, так як виділити фейкову новину чи пост від справжнього дуже важко, і можливо досягнути такого розподілу тільки з допомогою машинного навчання, або точкового використання нейромереж. Тільки так можна вивести неявні патерни, які виникають у фейкових новинах та відокремити їх від справжніх новин. Також, варто відмітити, що проблема актуальності даних відносно молода, і в ній буде ще багато нових відкриттів, як у сфері програмного фільтрування так і у сфері фільтрування читацького контенту. Але треба пам’ятати, що таке фільтрування значно збільшить ефективність обміну інформації всередині системи, а у читацькому аспекті дозволить виданням стати набагато більш стійкими до масових фейкових вкидів, що зробить більш стабільним і політичне життя, і економічний розвиток. І не дивлячись на молодість тема неймовірно актуальна, так як вплив неактуальної інформації на ефективність роботи з інформаціє неможливо недооцінити – вона викликає як серйозні проблеми з сервісами, так і не менш серйозні зміни з боку соціуму, якщо ми говоримо про фейкові новини чи хейт-спітч.

Також, згадуючи CAP-теорему потрібно помітити, що не завжди можна побудувати систему, яка зможе гарантувати оптимальну актуальність даних. Тому, ця проблема не може мати чисто архітектурного рішення(просто пишіть хороші системи – і актуальність буде висока), на жаль, це не вина розробників, а особливість функціонування дуже багатьох сервіс-орієнтованих систем. Тому, на програмному рівні потрібно створити ще один шар, який зможе вимірювати актуальність даних та фільтрувати інформацію, яка отримала низьку оцінку актуальності. Особливо важлива така оцінка буде при роботі з API, так як програміст може використовувати неактуальну інформацію одного сервіса для

того, щоб проектувати інший, і таким чином будуть створюватись цілі середовища, що засновані на неактуальних даних. Саме для боротьби з таким видом поширення неактуальної інформації потрібно використовувати нейронні мережі та з їхньою допомогою фільтрувати неактуальну інформацію.

Важливо відмітити, що схожу ланцюгову реакцію можуть викликати фейкові новини, коли один фейк використовує для свого підтвердження інший фейк і сам стає базою для третього фейку, таким чином створюється ціла «павутина» фейків, які перетворюються в систему, що підтримує сама себе.

Тому так необхідна фільтрація і механізми вимірювання рівня неактуальності інформації, а враховуючи фактор кратного росту інтернету, якщо не почати масово боротись з фейковою та неактуальною інформацією є великий шанс тотальної втрати ефективності сервісно-орієнтованих систем та перетворення інформації в мережі в тотальний інструмент, який буде діяти в інтересах певних груп населення, дозволяючи масово маніпулювати соціумом та проводити точкові вкиди хейт-спітчу для того, щоб впливати на ключові вияви народної волі (голосування, мітинги, місцеві голосування по важливих питання та інше)

РОЗДІЛ 2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЯВЛЕННЯ ТА ОЦІНКИ АКТУАЛЬНОСТІ ІНФОРМАЦІЇ В СЕРВІСНО-ОРІЄНТОВАНИХ СИСТЕМАХ

2.1. Метрика оцінки актуальності інформації в сервісно-орієнтованих системах

З математичної точки зору описання тексту для того, щоб з ним далі могла працювати нейронна мережа – не надто просте завдання, яке потребує досить впевнених знань в таких сферах математики, як статистика та теорія ймовірності. Для роботи з текстом в першому приближенні використовують метод «мішка слів».

Метод «мішка слів» - це метод, який дозволяє спростити конструкції натуральної мови. Використовуючи цю модель текст сприймається у вигляді «мішка» (а на справді – мультимасиву) що складається з усіх його слів. Використовуючи невпорядковану модель даних можна оцінити кількість унікальних токенів, що існують в певному тексті. Мішок слів можна подати у вигляді матриці, кожен рядок у якій відповідає окремому документу або тексту, а кожен стовпець — певному слову. Осередок на перетині рядка і стовпця містить кількість входжень слова у відповідний документ. З допомогою «мішка слів» можна легко і швидко знаходити явні патерни, які є в певному тексті, тому він дуже зручний для роботи з метриками.

Іншим методом є так званий метод «мішка N-грам». Цей метод розширює звичайний «мішок слів», бо замість слів використовує N-грами, в першу чергу бі-грами. Бі-грама – це два слова, які в тексті чи корпусі текстів є сусідніми. Щоб краще зрозуміти це поняття наведемо приклад. В тексті «Це було пізньою осінню» є наступні бі-грами – Це було, було пізньою, пізньою осінню. Бі-грами

дозволяють краще зрозуміти логіку тексту, і їх можна ефективно використовувати для оцінки текстів.

Наприклад, можна оцінити словниковий запас тексту, емоційну направленість тексту, кількість синонімів, кількість «води» та інші важливі характеристики. Також дуже важливий семантичний аналіз тексту, який дозволяє знаходити семантичний зв'язок між словами певного тексту. Зазвичай він ґрунтується на логічній побудові так званих «семантичних мереж» та «семантичних моделей» тексту. На наступному рисунку видно, як будується «семантична мережа» тексту, і як вона виглядає. Практично відразу стає зрозуміло, що семантична модель – це граф, так як складається з вершин та ребр виду $G = (V, E)$ і, звичайно, вона піддається всім математичним операціям, які властиві графам, як то обхід по ребрах.

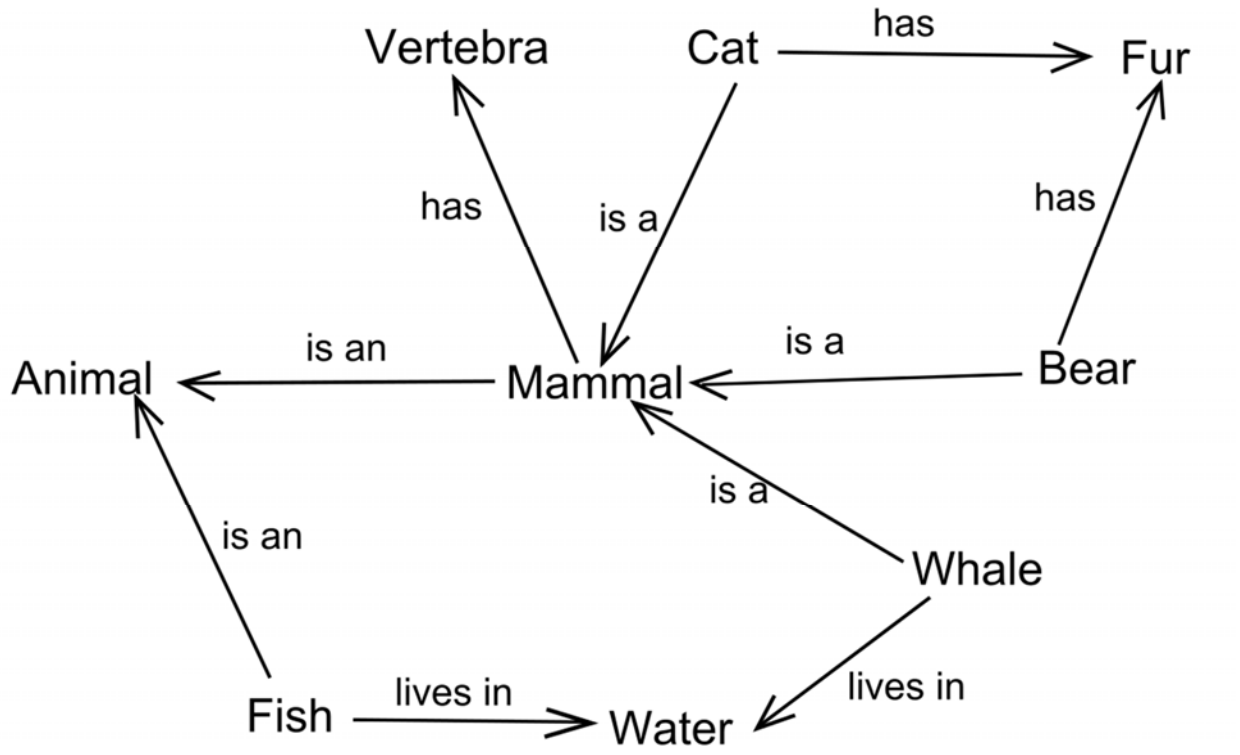


Рис. 2.1. Візуалізація семантичної моделі.

Відтак, можемо розширити «наївну» емоційну метрику для тексту.

Спершу створимо кращу метрику для оцінки емоційності тексту. Виділимо два словника, словник негативних емоційних слів (погано, жахливо, катастрофічно та ін.) та позитивних емоційних слів (чудово, прекрасно, неймовірно та ін.), звідси

$$М_{нег} = \frac{Кнег - Кстоп}{Квсього - Кстоп} \qquad М_{поз} = \frac{Кпоз - Кстоп}{Квсього - Кстоп}$$

Просто пробігаючи циклом по текстові, та знаходячи потрібні емоційні слова, а потім від кількості емоційних слів віднімаємо стоп слова, та від повного розміру тексту також віднімаємо слова ми отримуємо позитивну та негативну емоційну метрику. Далі можна легко оцінити емоційність тексту, якщо $М_{нег} > М_{поз}$ – значить, текст має негативне емоційне забарвлення, якщо $М_{нег} < М_{поз}$, то можна сказати, що текст має позитивне емоційне забарвлення, а якщо метрики рівні між собою – то текст не має емоційного забарвлення. Цікаво, що такі тексти зустрічаються дуже рідко, і якщо при цьому Кпоз і Кнег більше нуля, варто подумати, чи не згенерований цей текст спам-програмою. Тому, сумніви мають викликати як тексти, де емоційність зашкалює, так і тексти, де емоційність рівна нулю. Якщо висока емоційність буде вказувати на маніпулятивний текст, то нульова буде типова або до деяких видів текстів, або буде пов'язана з спам-атаками.

Крім емоційних слів, в фейкових текстах часто зустрічаються слова, які змушують поспішати або акцентують увагу на часові. Це такі слова, як «негайно», «терміново», «незабаром», «швидко» та інші, що пов’язані з часом. Оцінювати цей показник можна точно таким ж методом, як вище вказані.

$$М_{час} = \frac{Кчас - Кстоп}{Квсього - Кстоп}$$

Більше значення цього показника буде вказувати, на високий шанс маніпуляції, а менше значення буде пов'язано з більшим шансом правдивої новини.

Як помітно з формул, стоп-слова ми завжди забирали, але сам факт використання стоп-слів також багато говорить про текст. Тому варто виділити ще рівень стоп-слів

$$М_{стоп} = \frac{Кчас}{Квсього}$$

Високий рівень цього показника буде говорить про те, що текст повний «води» та малоінформативний, а низький рівень може бути показником того, що текст згенерований нейромережею.

Збираючи ці показники разом можна провести класифікацію текстів по наступній метриці, яку легко описати наступним алгоритмом : $М_{нег} > 0.2$ та < 0.4? Якщо так, то +1 бал метрики, якщо ні – ніяких дій.

$М_{час} < 0.2$? Якщо так, то +1 бал метрики, якщо ні – ніяких дій.

$М_{стоп} > 0.1$ та <0.3? Якщо так, то +2 бал метрики, якщо ні – ніяких дій.

Результати можна описати так – 1 бал – мінімальний шанс правдивості, 2 бали – низький шанс правдивості, 3 бали – середній шанс правдивості та 4 бали – високий шанс правдивості.

Звичайно, ця метрика також є досить тривіальною, але вона дозволяє краще зрозуміти те, що відбувається в текстові саме з боку лінгвістики.

Схожу, по суті, метрику можна зробити для програмної сторони оцінки актуальності інформації сервісно-орієнтованої системи, складатись вона може з довільної кількості компонентів, в залежності від того, що саме цікавить зараз розробника. Однозначно має включати наступні характеристики:

- Чи сервіс повертає файл WSDL ?
- Чи правильно передає дані про локацію?
- Чи актуальна і правильна локалізація сервісу?

- Чи дійсний SSL-сертифікат і чи взагалі є він?
- Чи зареєстрований сервіс в сертифікаційному центрі?
- Чи правильно працює API, чи повертає структуровані дані?
- В якій доменній зоні знаходиться сервіс? (тут можна використовувати багато варіантів, просто даючи їм різні числові характеристики, наприклад, ua додасть до метрики 3 бала, а net – тільки два, звичайно, це буде залежати від потреб розробників)

Отримуючи за кожен компонент довільну кількість балів, і переводячи потім бали в float-значення кожна команда програмістів зможе швидко оцінювати актуальність важливої саме для них інформації, а легкість компонування таких метрик дозволить використовувати багато різних варіантів, для оцінки різних сторін сервісу. Наприклад, одна команда може ставити в пріоритет питання кібербезпеки та давати більше балів для метрики в питаннях оцінки саме з боку сертифікатів, безпеки та іншого, і ігнорувати інші питання(наприклад, структуру даних), це дозволить кожній команді займатись справді тим, що їй важливо.

Варто додатково виділити, що таку метрику дуже легко перетворити в блок-схему, записати і обмінюватись метриками саме на рівні блок-схем, в такому випадку їх і легше аналізувати, і набагато легше описувати програмно(по суті, це вже готовий алгоритм)

Також потрібно відмітити, що і в реалізації такі метрики дуже зручні – її цілком можна вмістити в маленький парсер, який приймає тільки лінк на потрібний сервіс, і повертає різноманітні показники актуальності різних компонентів сервісно-орієнтованої системи. Також, простота надавання кожному з компонентів окремої кількості балів повторює дуже зручну вагову структуру, і дозволяє виділяти більш важливі компоненти та менш важливі, а дискретна побудова метрики дозволяє не реалізовувати складні формули, з

якими потім можуть бути проблеми. По суті, дана метрика є реалізацією так званої степ-функції Хевісайда, яка просто має декілька сходинок в залежності від побудови функції. Звичайно, кількість сходинок і їх висоту можна міняти – і це дозволяє компонувати дійсно комфортні та зручні метрики для оцінки інформації.

2.2. Метод виявлення та оцінки актуальності інформації в сервісно-орієнтованих системах.

Використовуючи різноманітні метрики та створюючи на їх основі свої можна приступити до наступного етапу – фільтрування даних. Якщо певні програмні дані, чи текстові дані для читача отримали погані показники по метриці, якій ми довіряємо – інформація має бути відбракована, визнана неактуальною та не має йти нікуди далі. Взагалі, оптимальним алгоритмом для фільтрації будь-якої інформації має стати наступний

1. Створення конкретної метрики, яка дозволяє оцінити ті показники, які зараз важливі
2. З допомогою даної метрики створення датасету вигляду інформація-бінарний класифікатор (актуальна-неактуальна)
3. Завантаження датасету в нейромережу та навчання нейромережі
4. Подальше використання замість метрики навченої нейромережі.

Такий підхід дозволяє повністю автоматизувати роботу з інформацією, а великий датасет переданий нейромережі дозволяє створити дійсно універсальний продукт, що дозволить обробляти широкі масиви інформації.

У зв’язку з тим, що основним та універсальним методом для того, щоб оцінювати актуальність інформації пропонується використання штучних нейронних мереж, потрібно максимально формально описати, що це таке, як вона працює та виглядає.

Штучна нейронна мережа – це обчислювальна функція, яка опрацьовує певний масив інформації та подає на вихід певне число чи масив чисел. Нейромережі були натхненні біологічними нейронними мережами, які знаходяться в нервовій системі людини, тварини та більшості всього живого на планеті. Виникнення нейронних мереж пов’язано з ім’ям Франка Розенблатта, який створив першу примітивну мережу – «перцептрон» в 1955 році[21]. Примітивна нейронна мережа складається з трьох шарів – шару вхідних даних, прихованого шару та шару вихідних даних та вагів, які і піддаються подальшому навчанню.

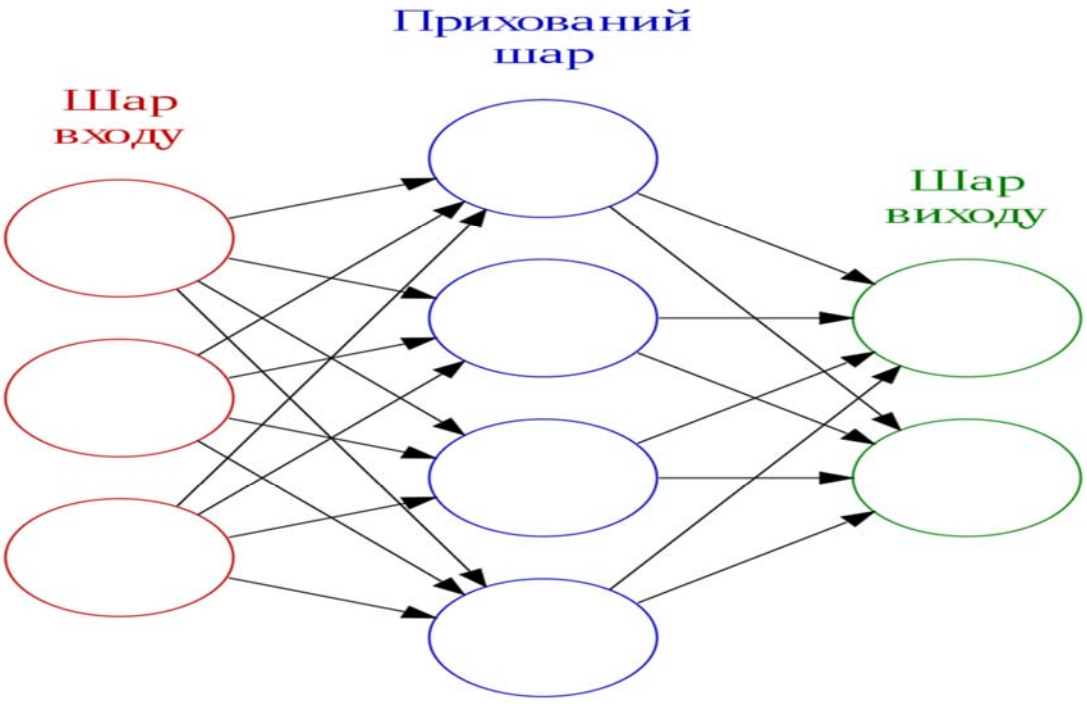


Рис 2.2. Архітектура найпростішої нейронної мережі.

Шар вхідних даних – це шар, в яких передаються значення датасету, на яких і має проходити навчання. Розмірність шару вхідних даних залежить від розміру конкретного екземпляру навчального датасету. Наприклад, якщо датасет

складається з векторів по три числа, наприклад, [1,2,3] – то на шарі вхідних даних має бути саме три входи.

Після цього вхідні дані перемножуються на значення вагів(які формуються випадково в певному інтервалі, наприклад [-2,2] та передаються на нейрон прихованого шару (наприклад, якщо на нейрон 1 прихованого шару ведуть два ваги, що мають значення 1 і 2, а входів два і їх значення 3 і 4, то результат сумації на нейроні буде наступним, 3*1 + 4*2 = 11). Результат сумації на нейроні передається через активаційну функцію, і надалі перемножується вже з вагами, що знаходяться між прихованим шаром та шаром вихідних значень. Після цього проходить отримання значення на нейронах вихідних, активація їхньої функції – і отримання вихідних значень. Після цього від отриманої відповіді віднімається очікувана, і починається процедура навчання, в основному використовуючи технологію зворотного поширення помилки, яке описується наступною формулою, яку можна побачити на малюнку.

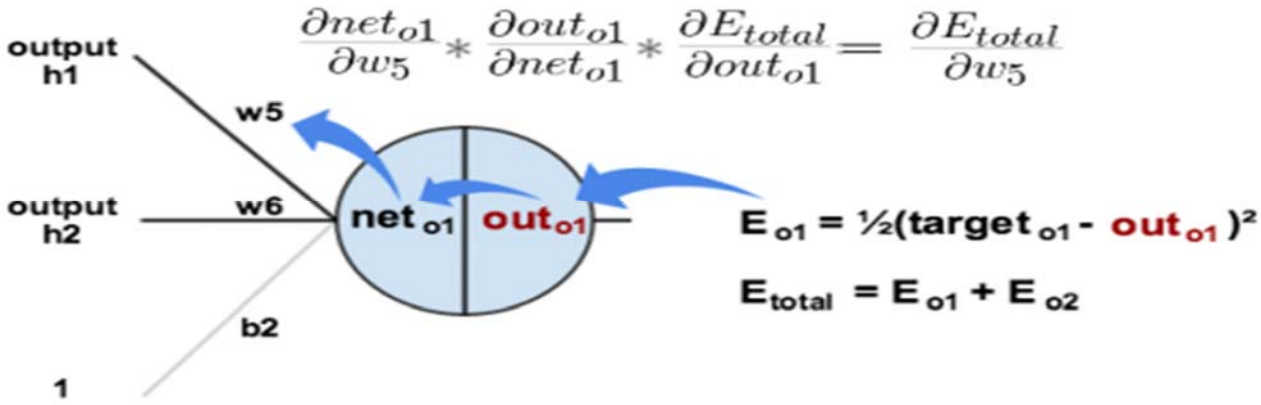


Рис. 2.3. Формула зворотного поширення помилки.

В результаті навчання показники вагів методично міняються для того, щоб ідеально відповідати патернам, які виникають в датасеті. Таке навчання проходить сотні ітерацій, і призводить до оптимального результату.

Також, варто пам’ятати про можливість «перенавчання» - це ситуація, коли нейронна мережа точно повторює патерни, які виникають в навчальному

датасеті, але настільки точно, що вони не відповідають патернам, що взагалі виникають в універсальному датасеті. Така нейромережа дає дуже високу точність, коли працює вже з відомим датасетом, але з схожим, але трішки іншим дає набагато меншу точність. Є декілька способів боротьби з перенавчанням, одним з таких способів є так званий «дропаут», про нього буде написано пізніше.

Звичайно, це не єдина методика навчання нейромережі, є інші, більш екзотичні способи навчання, але на них ми зупинятись не будемо, так як їх варто реалізовувати тільки для дуже специфічних задач.

Кількість прихованих шарів можна міняти, особливо коли йде робота з великими датасетом, тоді є зміст підключення додаткових прихованих шарів, також можна міняти кількість нейронів.

Також треба виділити декілька проблем, які є в нейронних мережах, тому що вони дозволять краще зрозуміти їх можливості та сферу застосування.

- Чорний ящик. Після процесу навчання неможливо зрозуміти як саме нейромережа опрацьовує патерни, які знайшла в масиві даних. Єдиний спосіб оцінки якості нейромережі – це точність з якою вона оцінює певний датасет. Хоча, якщо з теоретичної точки зору це і мінус, то чисто з практичної точки зору – це може бути й плюсом. Реалізація закрита сама в собі (принцип інкапсуляції) та ніяк не впливає на інші процеси. Тому, хоч це і проблема, вона додає зручності в використані нейромережі.
- Високі вимоги до машинних потужностей. Дійсно, якщо говорити про роботу з великим датасетом – потрібно декілька прихованих шарів, кожен з яких буде включати в себе сотні нейронів. Задача сумації та навчання такої мережі займає дуже багато часу, та потребує значних потужностей, на звичайному персональному комп’ютері навчання такої мережі може

займати багато діб. Тому, щоб не витрачати надто багато часу, для дослідження варто використати маленький датасет і не надто переускладнену архітектуру, так як вона гарантує мінімальний час на обробку даних

- Високі вимоги до лемматизації тексту та очистки його від стоп-слів. Без правильної підготовки тексту нейронна мережа не зможе дати адекватний результат.

2.3. Алгоритм реалізації методу оцінки актуальності інформації

Нейромережі не можуть працювати з натуральним текстом, так як це погано впливає на точність та на можливість до навчання. Тому, «сирий» текст потрібно обробити для оптимальної точності та знаходження правильних патернів в тексті. Для правильної обробки натурального тексту з допомогою нейромережі потрібно зробити наступні етапи:

- Очистка. Це найперший етап роботи з текстом. В ньому очищують текст від усіх знаків, які не можуть допомогти зрозуміти внутрішній паттерн, Видаляються усі знаки пунктуації, особливі символи, теги та інше. Хоча, тут треба бути дуже обережним, так як можна очистити ті дані, які важливі саме в цьому тексті. Тому, наприклад треба бути дуже обережним – якщо очистити текст про економіку та валюти від знаків валют – можна втратити патерн. Тому очистка – справжнє мистецтво.
- Препроцессинг. Це великий етап підготовки тексту, який включає в себе декілька кроків:
    1. Приведення символів до одного регістру, щоб всі слова були написані маленькими символами
    2. Токенізація – процес розбиття тексту на токени, тобто на окремі компоненти – слова, речення чи фрази.

3. Тегування частин речення, це процес знаходження частин мови в кожному реченні для використання граматичних правил.
4. Видалення стоп-слів. Стоп-слова – це слова, які не несуть контексту, і ніякого змісту в передачі їх в нейромережу немає. Зазвичай, до стоп-слів варто відносити ті слова, що використовуються для зв'язки речень і не несуть змісту, як окремі слова. Їх потрібно видалити. Ключова проблема – створення словника з стоп-словами. Під час цієї роботи був створений набір з 1783 українських стоп-слів, тому що не було знайдено достойних словників, для того, щоб видаляти стоп-слова.
5. Спелл-чекінг – процес знаходження та виправлення помилок, опечаток та іншого. Так як помилкове слова погано впливає на точність нейромережі і подальшу лемматизацію.
6. Лемматизація або стеммінг. Це механізм для приведення слів до єдиної форми. Лемматизація практично завжди показує значно кращі результати, чим стеммінг, тому що стеммінг це груба і загальна операція, а лемматизація – більш інтелектуальна операція, яка показує кращі результати при подальшій передачі інформації в нейромережу. Стеммінг знаходить основу слова, і тому може часто відрізати частинку від кореня, що однозначно погано впливає на точність.

- Векторизація. Після закінчення препроцессингу слова перетворюються в числові дані, нейромережі працюють не з текстом, а з числами. Тому використовуються такі методи як «мішок слів» і «мішок N-грам», вони були описані вище.

Для того, щоб правильно працювати з текстом та домогтись оптимальної оцінки новин варто тезово описати прогрес, який на даний момент є в сфері нейронних мереж та обробки текстів. Вперше поставили перед собою питання на рахунок розподілу фейкових і реальних новин в 2008 році вчені Джіндаль та

інші [7]. Вони використали звичайні механізми машинного навчання – наївний баєсовський класифікатор та логістичну регресію, і отримали досить хороші результати у відношенні боротьби з фейковими новинами – 95% точності, та, точність на якісних фейкових статтях була значно нижчою, і вони так і не змогли запропонувати новий спосіб боротьби з неактуальною інформацією. Максимальна точність, яку вони могли отримати при роботі з великими фейковими статтями – 65%, що цікаво, подальші дослідження [7] довели, що приблизно з такою точністю можуть розмічати фейкові та реальні новини звичайні люди. Тобто, результат Джіндаля та його команди 2008-2009 році був наближений до людського результату.

Наступний етап відкрила команда під керівництвом Отта, вони почали використовувати метод опорних векторів для того, щоб з більшою точністю виділяти патерни в текстах. Метод опорних векторів - це механізм, що розширює типове тренування з учителем. Цей метод генерує оптимальну гіперплощину для того, щоб розділити два класи та провести оптимальну класифікацію. На рисунку 7 показано, як цей метод знаходить оптимальну гіперплощину та правильно розділяє два різних класи. Вони отримали точність близько 85%

Потім прийшла епоха різноманітних унікальних архітектур, які використовували неповні нейронні мережі, чи нейрони з унікальними зв'язками між шарами, чи досить специфічні активаційні функції. Вони змогли показати ще більшу точність, але, при цьому, мали багато серйозних недоліків, і в першу чергу – низьку універсальність, що не підходить, якщо ми говоримо про

нейромережу, що зможе оцінювати широкий спектр різноманітних текстів.

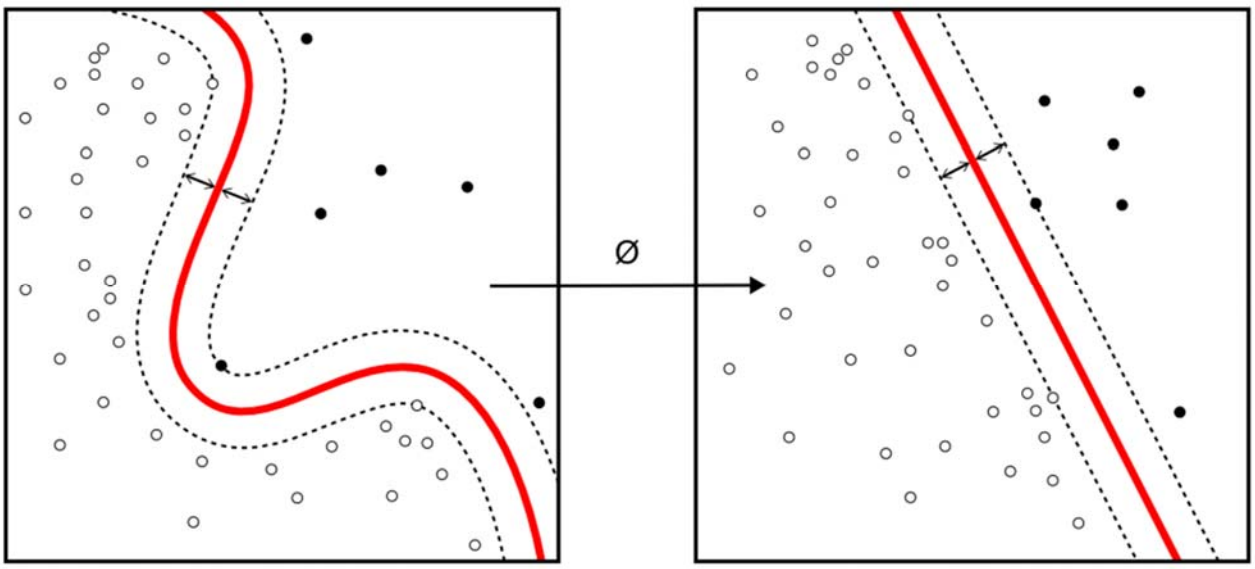


Рис. 2.4. Візуалізація методу опорних величин.

Висновки до розділу 2

В цьому розділі були проаналізовані основи побудови нейромереж, було описано, як вони працюють, як будуються та які функції можуть виконувати, був описаний механізм створення досить універсальних метрик для оцінки актуальності даних з програмного та текстового боку. Підсумовуючи, варто сказати, що використовувався не надто багатий математичний апарат саме через універсальність задач, що вирішують нейромережі. Не варто забувати, що нейромережа може ефективно знаходити патерни в будь-яких даних, і використовувати її можна як для текстових даних, так і для апаратних, як то дані з API, REST чи Soap-пакети та інші формати даних. Достатньо правильно розмітити датасет та використати його для того, щоб провести навчання нейромережі.

Сучасна інформатика дуже зацікавлена в тому, щоб вирішити таку проблему, як фільтрація неактуальних даних, не дивлячись, що це питання дуже

молоде, та ще не встигло отримати широке поле для аналізу. Тим не менше, його важливість важко переоцінити. Ще в 2008 році перші групи вчених вже почали цікавитись тим, як боротись з неактуальною інформацією, як її фільтрувати та звільнити сервіси від неї. Якщо ознайомитись з даними статистики, стає зрозуміло, наскільки деструктивний вплив несуть в собі фейкові новини.

Насправді, серйозну роль в ефективності фейків відіграють соціальні та психологічні чинники, наприклад, статистично доведено, що користувачі відчувають себе непевними та дезорієнтованими, коли перед ними стоїть завдання розрізнити між собою правдиві та фейкові новини.

А згідно наукових досліджень здатність людини розрізнити між собою фейкову інформацію від правдивої трішки більша, чим випадковість, показники точності знаходяться в діапазоні 53-38% з середньою точністю 54% для 1000 учасників у понад 100 експериментах. З фейковими новинами ситуація ще гірша, а коли в справу вступає «павутина фейків» показники стають взагалі катастрофічними – менше 20% людей може знайти брехню.

Сумні дані і з програмної точки зору – згідно результатів, що отримали А. М. Мельник, М. П. Дивак, та Р. М. Пасічник – більше половини API сервісів, що надають доступ до оцінки стану ґрунтів отримали оцінку актуальності меншу або рівну 0.5, що показує низький рівень актуальності інформації багатьох сервісів. Схожі дані надають і європейські та англійські колеги по питанню вивчення актуальності інформації.

Тому, зручні та модульні метрики дозволять проводити швидкі та дешеві дослідження для оцінки програмної актуальності сервіса (наприклад, на мові програмування Python з допомогою бібліотеки requests можна швидко та зручно написати простий парсер, що зможе перевіряти багато сайтів на актуальність

інформації за мінімальний час), саме простота на першому етапі допоможе збирати компактні, але показні датасети, на яких можна масово вчити нейромережі і в подальшому використовувати їх для точної та зручної фільтрації інформації. Саме унікальна можливість нейромереж, яка заключається в можливості легко знаходити приховані патерни даних може привести до революції в питані актуалізації даних та фільтрації неактуальної інформації.

Звичайно, має зміст і розробка більш точних і універсальних метрик для оцінки, але одна універсальна метрика буде мати декілька проблем:

- Проблеми з простотою написання та використання. Універсальна метрика буде витрачати значно більше часу для використання та аналізу тексту, що буде затягувати процес оцінки, якщо річ йде про великі простори чи глобальні продукти
- Неможливість концентрації на більш важливих для конкретної оцінки факторах, що призведе до того, що маленькі групи розробників, які можуть оцінювати окремі сторони актуальності сервісу будуть змушені розробляти власні метрики для оцінки саме тих сторін сервісу, що їм важливі, в результаті це призведе до нішевості даної метрики.
- Проблеми з фільтрацією. Для широкої оцінки та фільтрації все одно треба буде використовувати нейромережі. Через правило «чорного ящику» ми все одно не зможемо оцінити ті патерни, які виділила нейромережа. Тому, розробка широкої та універсальної метрики не зможе суттєво змінити ситуацію з використанням нейронної мережі.

Тому, використання найбільш простих степ-метрик, створення яких просте і універсальне дозволить оптимізувати оцінку актуальності інформації та проводити найбільш зручну та ефективну фільтрацію інформації.

Метрики мають бути використані для того, щоб створювати набори датасетів, які потім будуть використані для навчання нейронної мережі, а через універсальність інформації, яку можна передати в нейромережу датасети можуть використовуватись як для оцінки актуальності програмної, так і для оцінки актуальності інформації текстової, в вигляді статей, новин та іншого.

Все одно різна інформація буде проходити однакові кроки – очищення від спеціальних знаків, викидання стоп-слів, лемматизацію, нормалізацію та векторизацію.

Окремим питанням стоїть створення зручних та універсальних наборів українських стоп-слів, баз для лемматизації та нормалізації, словників для векторизації. На жаль, на даному етапі з ними є певні проблеми, але з розвитком вітчизняних нейронаук ці бази мають бути напрацьовані.

РОЗДІЛ 3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЯВЛЕННЯ ТА ОЦІНКИ АКТУАЛЬНОСТІ ІНФОРМАЦІЇ В СЕРВІСНО-ОРІЄНТОВАНИХ СИСТЕМАХ

3.1. Особливості програмної реалізації системи для виявлення та оцінки актуальності інформації

Опишемо життєвий цикл програми, що заснована на нейромережі. Основними етапами розробки нейронної мережі є:

1. Постановка задачі і вибір архітектури нейромережі.
2. Аналіз вхідних та вихідних даних
3. Формування датасету
4. Обробка та нормалізація датасету
5. Навчання нейромережі
6. Тестування нейромережі
7. Практичне використання нейромережі. Останній етап розробки нейромережі, коли точність достатня, і нейромережу можна використовувати для практичної задачі. В такому випадку ваги нейромережі можна зберегти та використовувати їх в подальшому для практичного використання.

Розберемо те, що було розроблено на кожному етапі роботи з нейронною мережею. Постановка задачі – нейромережа, яка зможе знаходити фейкові новини і відрізняти їх від чесних нових. На вивід має виходити результат роботи сигмоїдальної функції – число з плаваючою комою в кордоні 0,1, де 1 – точно правдива новина, 0 – точно фейкова новина. Розробка архітектури нейромережі – однозначно має бути один прихований шар, в якому може знаходитись від 40-50 нейронів, кількість буде підбиратись експериментально методом зрівняння точності навченої нейромережі. На виході буде стояти нейрон з сигмоїдою в вигляді активаціонної функції, або ReLu (також відомий

як випрямлений лінійний вузол), для того, щоб мінімізувати помилку. Це потрібно також знайти експериментально, при чому нейромережа буде видавати кращі результати.

З використанням парсінгу на мові програмування Python був створений датасет – 50 текстів, кожен яких позначений бінарно – 1 якщо новина правдива, і 0 – якщо новина явний фейк. Датасет збирався з стрічки новин декількох українських новинних сайтів на початку листопада 2022 року.

Після цього починається обробка тексту для того, щоб в подальшому передати їх нейромережі – в результаті проходить очистка від стоп-слів, нормалізація, лемматизація та векторизація тексту. Після векторизації кожне слово замінене на частотність цього слова. Також, якщо відрізняється розмір новин в слова проходить приведення до одного розміру – всі вектора приводяться до однієї довжини, до довжини максимального тексту новин, всі вектори, що менше максимального – додаються нулям до однієї розмірності.

Саме для того, щоб боротись з розрідженими векторами, на які , зазвичай, нейромережі реагують дуже погано був використаний ембеддінг. Ембеддінг – це механізм зменшення розмірності вхідного вектора для того, щоб підняти

точність нейромережі.

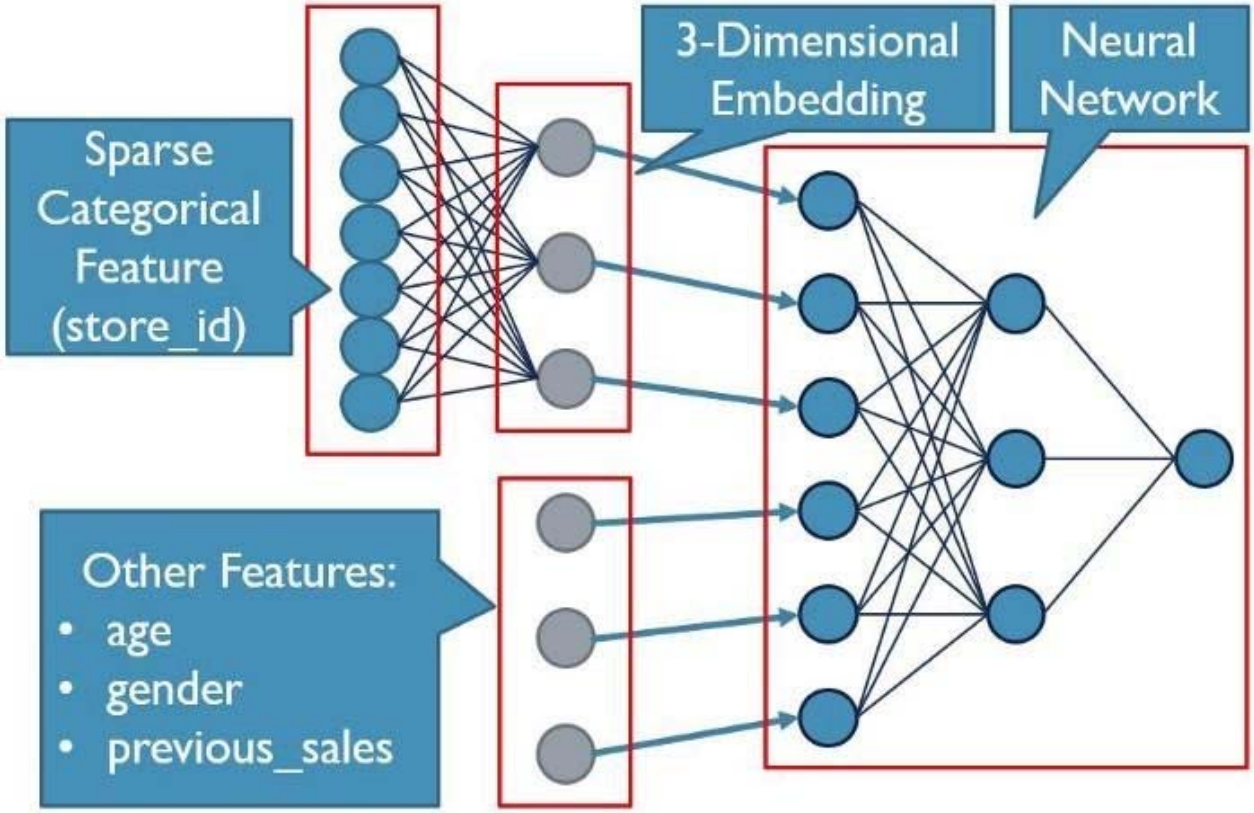


Рис. 3.1. Візуалізація ембеддінгу

На рисунку 3.1 показано, як саме працює ембеддінг, міняючи набір вхідних даних та зменшуючи його. В нашому випадку, коли ми працюємо з текстом, шар ембеддінгу отримує на вхід номера слів, а на вихід повертає їх векторні представлення. Це змальовано на рисунку 3.2.

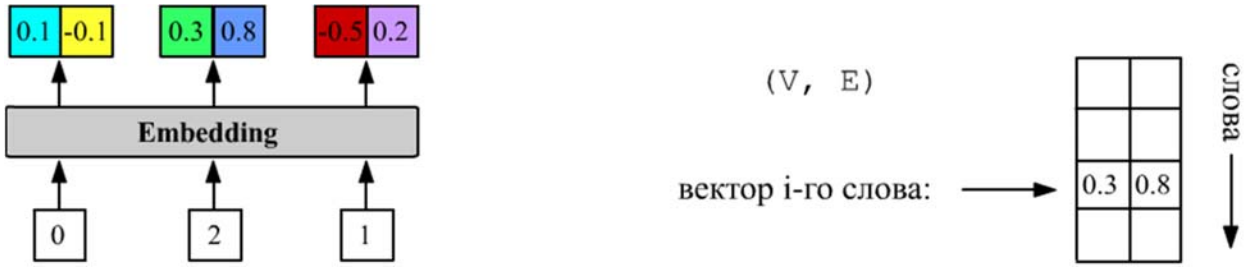
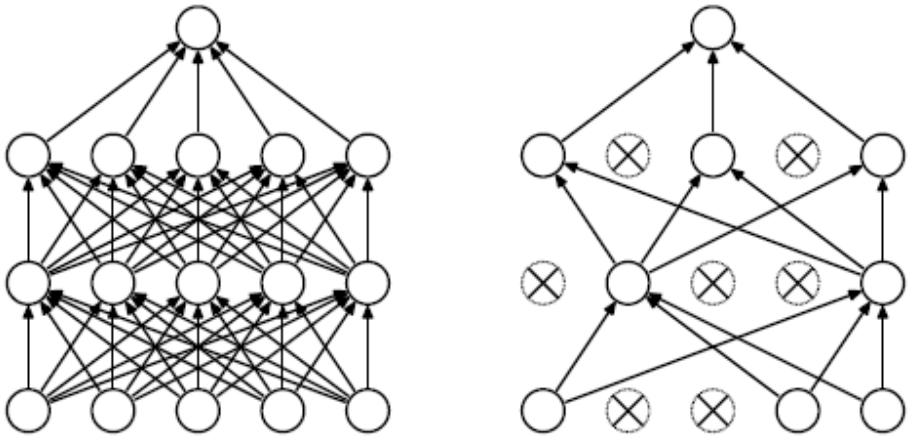


Рис. 3.2. Робота ембеддінгу

Також потрібно використати так званий дропаут (dropout, виключення) – специфічний механізм, який дозволяє оптимізувати роботу нейронної мережі, кожну ітерацію відключаючи певну кількість нейронів(в процентах) для того, щоб навчити нейромережу більш рівномірно, і не приводити до перенавчання окремого нейрона, що погано впливає на точність та ефективність роботи нейромережі. На рисунку зображено порівняння нейромережі без використання дропауту, і з використанням на прикладі конкретної ітерації. На наступній ітерації будуть відключенні інші випадково вибрані нейрони, що дозволить мінімізувати перенавчання нейромережі.



(a) Standard Neural Net (b) After applying dropout.

Рис. 3.3. Принцип роботи дропауту

3.2. Інструкція користувача та тестування системи

Користувач може використати дану нейромережу в двох режимах – повному(в якому йому потрібно буде самостійно навчити нейромережу на власному датасеті) або в навченому варіанті(згідно якого він вже використовує готові навчені ваги нейромережі, та просто передає на нейромережу свій текст, який хоче перевірити на фейковість.

Почнемо з навченого варіанту – це вже готова нейромережа, яка зберігає архітектуру та значення вагів в файлі .h5 формату, під час запуску в терміналі користувач вводить свій текст, та отримує на вихід число в float форматі, яка оцінює шанс того, що новина фейкова. Приклад роботи навченої нейронної мережі можна бачити на наступному малюнку. У відповідь на фейкову новину нейромережа видає правильну відповідь – ця новина схожа на правду тільки на 4%. Точність на цьому конкретному прикладі дуже близька до оптимальної, хоча при більших прикладах вона буде не така вражаюча.

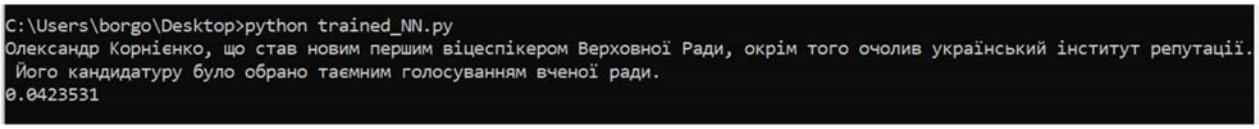


Рис 3.4. Приклад роботи навченої нейромережі

Повна робота з нейромережею буде набагато складнішою. Для цього потрібно спочатку сформувати датасет форматі .csv(**C**omma-**S**eparated **V**alues, значення розділені комою – це текстовий формат для того, щоб записувати табличні дані, також потрібно очистити текст перед завантаженням в цей формат від ”), в який внести дані в форматі “текст”, бінарне значення фейковості(якщо 1 – новина правдива, якщо 0 – фейкова). Після цього потрібно почекати, поки функція підготовки тексту проведе перевірку датасету, очистку тексту, видалення стоп-слів, лемматизацію та нормалізацію тексту, і нарешті - векторизацію тексту, створить достатню архітектуру нейромережі(де кількість нейронів буде відповідати кількості текстів датасету) та проведе навчання нейромережі та збереження вагів та архітектури в тому ж форматі .h5. Звичайно, що з допомогою повного навчання користувач зможе оптимізувати точність нейромережі до свого конкретного датасету, що дозволить сильно збільшити точність нейромережі та отримати якісні результати, або виділити патерн саме певного набору фейкових новин (наприклад, одного автора чи однієї спам команди)

Тестування нейромережі проводилось на кожному етапі обробки тексту в unit-форматі, щоб ловити баги до кожної конкретної функції, а не шукати конкретний баг у всьому коді. Як має виглядати правильно unit-тестування можна подивитись на наступному рисунку. Саме тестування кожної окремої функції дозволяє ефективно ловити конкретні помилки та виправляти їх з максимальною ефективністю.

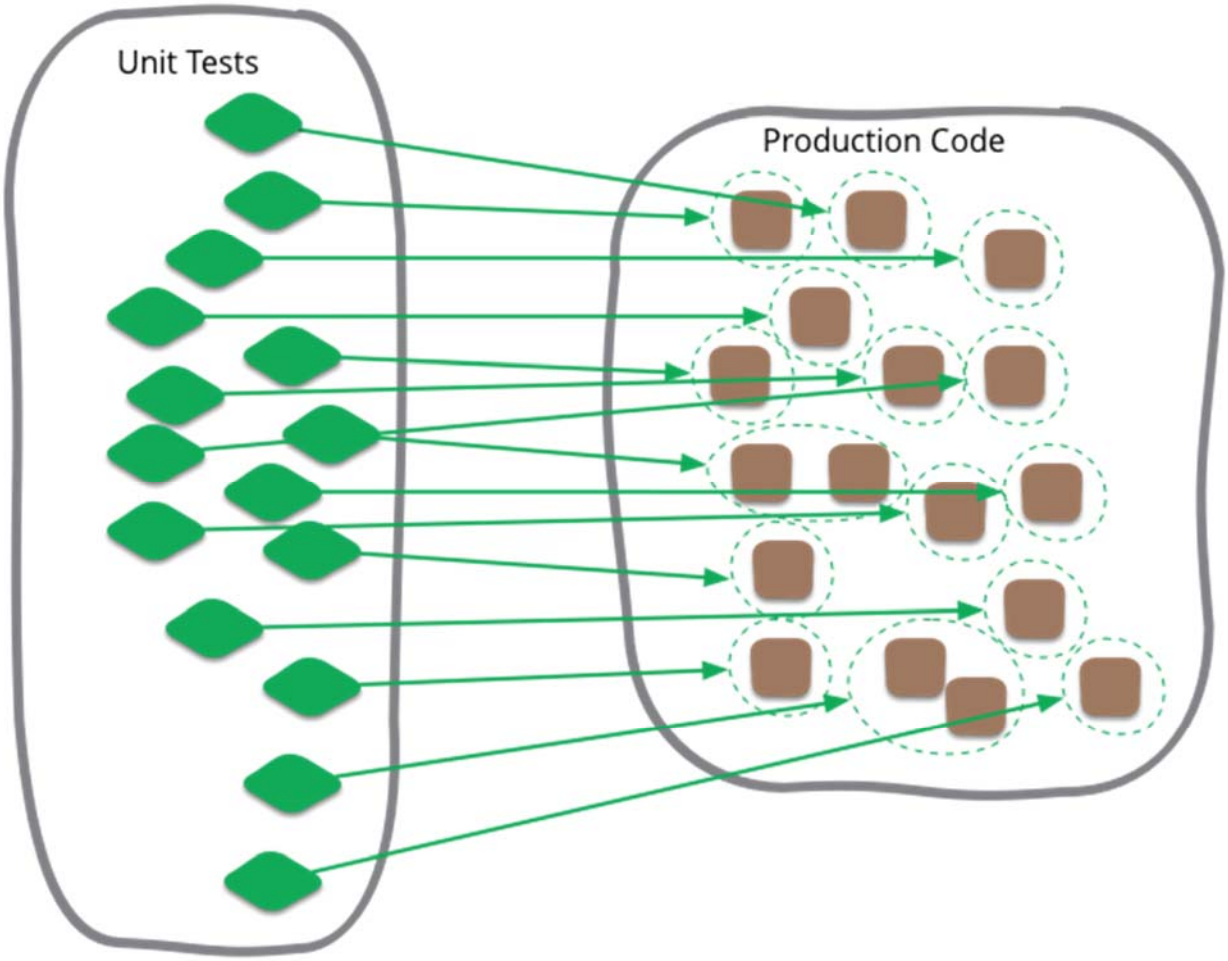


Рис. 3.5. Візуалізація основного принципу unit-тестування

Тому спочатку проводиться тестування та перевірка на валідність датасету. Чи в правильному кодуванні файл, чи не побитий він, чи достатньо записів (мінімум 10), чи немає неповних даних (тільки бінарний класифікатор без тексту, чи навпаки – тільки текст без бінарного класифікатора).

Приклад того, як виглядає датасет можна подивитись на наступному рисунку. Це приклад валідного датасету, який складається з 50 записів, які правильно розмічені, немає ні пропусків, ні помилок заповнення. Кожному текстові правильно відповідає бінарний класифікатор.

	text	label
0	Міністр цифрової трансформації України Михайло...	1
1	У Пентагоні повідомили, що наступного тижня ві...	1
2	Україна отримала від Польщі 1570 додаткових те...	1
3	В Україні розробили новий розвідувальний безпі...	1
4	Україна отримає американські самохідні зенітно...	1
5	Бронемашини Kirpi турецького виробництва допом...	1
6	Європейський парламент схвалив угоди ЄС з Укра...	1
7	НАЕК Енергоатом запропонував уряду приєднати д...	1
8	Два великі кораблі з вимкненими трекерами з'яв...	1
9	Уряд прогнозує майже вдвічі зростання державно...	1
10	Сергій Надал вже не хоче бути близьким до наро...	0
11	Російська кінокомпанія Централ Партнершип на з...	0
12	Петро Порошенко став співвласником Дизель Шоу ...	0
13	Голова Офісу президента Андрій Богдан змінив і...	0
14	Співробітники Укрексімбанку, які не змогли пов...	0
15	Польські Військово-морські сили розпочали проц...	1
16	Народний депутат Олександр Трухін купив собі л...	0
17	Благодійній ініціативі «ОКО ЗА ОКО» за тиждень...	1

Рис. 3.6. Приклад датасету для нейронної мережі.

Далі відбувається процес очищення від усіх спеціальних символів, як то кома, крапка, знак оклику та інші, та приведення до нижнього регістру. Так текст «Нарешті я вільний, – сказав Трухін. – Я давно мріяв пожити без метушні:» перетворюється в «нарешті я вільний сказав трухін я давно мріяв пожити без метушні», після цього відбувається відбраковка стоп-слів, і вище описана фраза перетворюється в «я вільний трухін я давно мріяв пожити без

метушні», в подальшому фраза проходить через модель, навчену на українській мові і заснованій на частотності слів, після цього відбувається векторизація тексту, як вона виглядає можна подивитись на наступному рисунку: Це набір очищених слів, що перетворені в частотність кожного слова в навченній українській моделі. Саме такий вектор передається на вхід нейромережі, після цього там відбувається ембеддінг(зменшення розмірності вектора), після цього відбувається власне навчання, яке складається з навчальної та тестової вибірки даних.

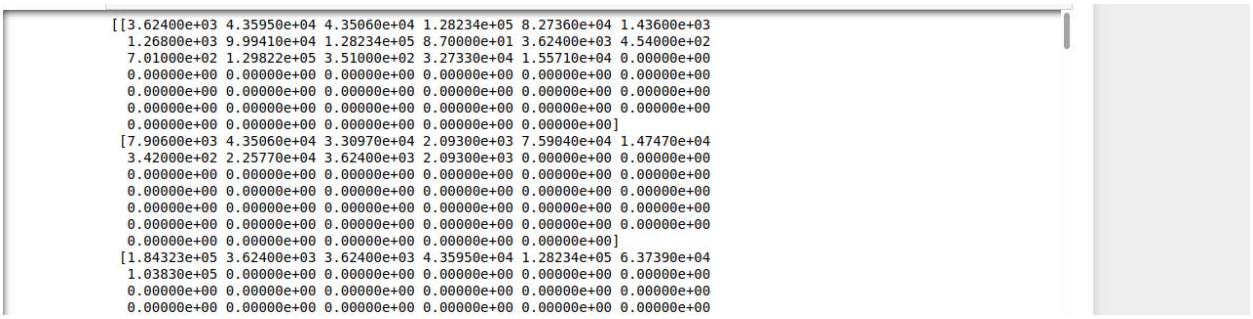


Рис. 3.7. Результат векторизації тексту.

Навчання складається з епох, після кожної з них відбувається оцінка основних параметрів точності нейромережі, тому можна дивитись, як саме відбувається навчання та чи не наступив параліч нейромережі. Як видно, при перших ітераціях навчання нейромережу сильно осцилює, але потім наступає впевнений тренд що заключається в постійному падінні середньоквадратичної оцінки.

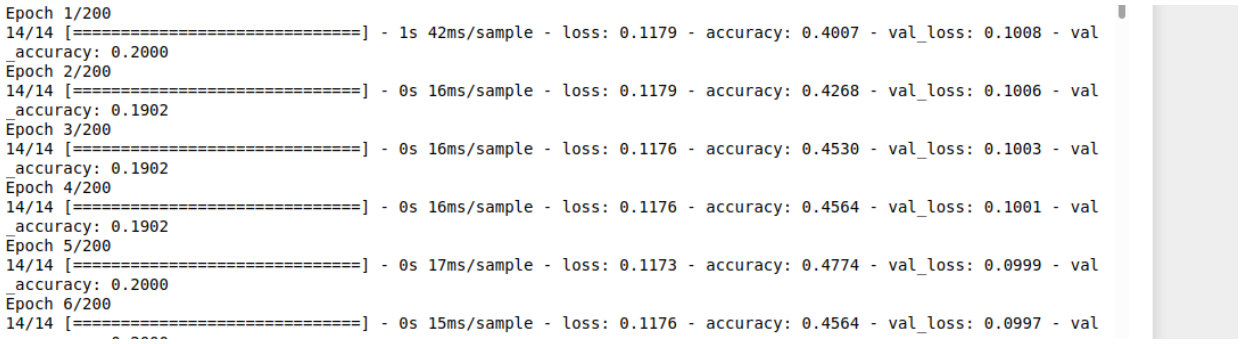


Рис. 3.8. Приклад того, як відбувається навчання.

Завдяки unit-тестуванню під час написання нейромережі всі баги були швидко знайдені та виправлені. Єдині проблеми були з використанням бібліотеки tensorflow, яка швидко поставилась та запустилась під керуванням операційної системи Linux, у деяких користувачів можуть бути проблеми пов’язані з установкою потрібних бібліотек, але це можна вирішити з допомогою спеціального ПЗ, яке може якісно перетворювати .py – код в комфортні для більшості користувачів Windows .exe – файли. В результаті дану нейромережу можна з мінімальними втратами передавати іншим користувачам, а через функціональну природу легко розширювати та доповнювати стороннім розробникам, які мають мінімальні знання в мові програмування Python.

3.3. Експериментальні дослідження та оцінка ефективності методу виявлення та оцінки актуальності інформації

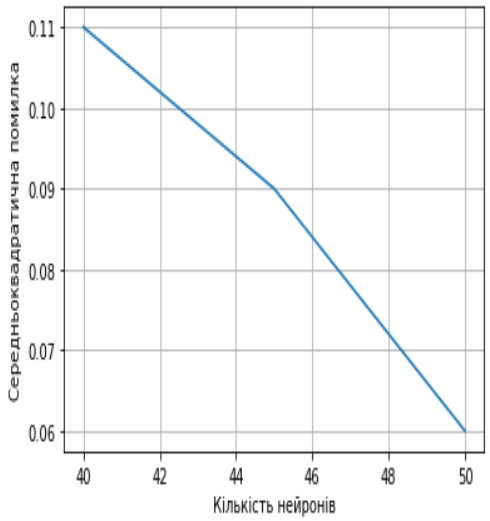
Першим кроком до того, щоб знайти оптимальний розмір та архітектуру нейромережі потрібно підібрати оптимальну кількість нейронів. Золотим стандартом для малих датасетів є кількість нейронів, що відповідає розміру датасету, через те, що розмір датасету – 50 текстів, було обрано саме 50 нейронів. Також було проведено перевірку з 40, 45, 55 нейронами. Для оцінки точності нейромережі використовується середньоквадратична помилка. Її формулу можна побачити на рисунку. Формально її можна описати як суму залишків регресії (залишок регресії – результат віднімання очікуваної відповіді від отриманої), що підняті в квадрат, що ділиться на загальну кількість помилок. Середньоквадратична помилка – золотий стандарт, що використовується в

нейронних мережах та технологіях машинного навчання.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Рис. 3.9. Формула середньоквадратичної помилки.

Справді оптимальна помилка була при 50 нейронах. Як мінялась помилка можна подивитись на рисунку – при збільшенні кількості нейронів до 50 помилка починала падати з максимуму в 0.11 до абсолютного мінімуму в 0.06. На наступному малюнку можна побачити, що з ростом кількості нейронів більше 50 помилка знову починала рости, тому можна сказати, що оптимальний рівень нейронів при даному датасеті – 50.



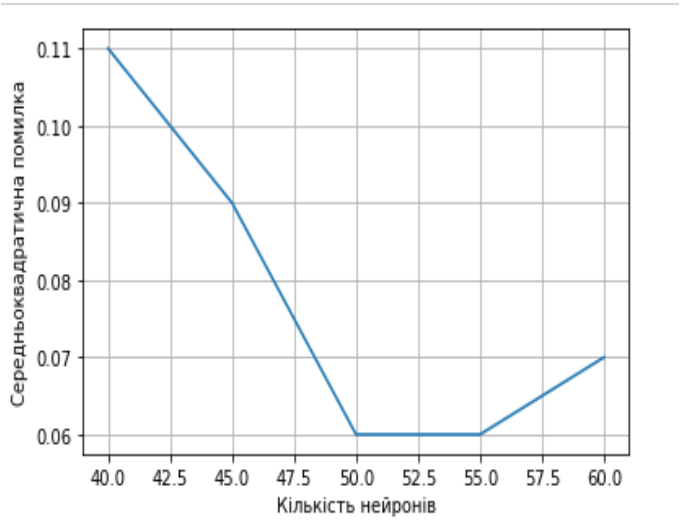


Рис. 3.10 та рис. 3.11. Точність в залежності від кількості нейронів

Наступним важливим питанням оптимальності нейромережі був розмір ембеддінгу. До якого розміру потрібно зменшувати вектор, щоб отримати

оптимальний варіант по помилці. Розмір вектора ембеддінгу тестувався в інтервалі від 4 до 20. Середньоквадратична помилка при розмірі вектора в 4 склала 0.14, різко підскочила до 0.18 при розмірі в 5(що можна пояснити як і невдалою генерацію вагів всередині прихованого шару нейромережі, так і особливостями датасету), і плавно спускалась до абсолютного мінімуму при розмірі ембедінгу 12 чисел. З ростом понад 12 помилка знову починала рости, тому, можна сказати, що оптимальна розмір ембеддінгу – 12.

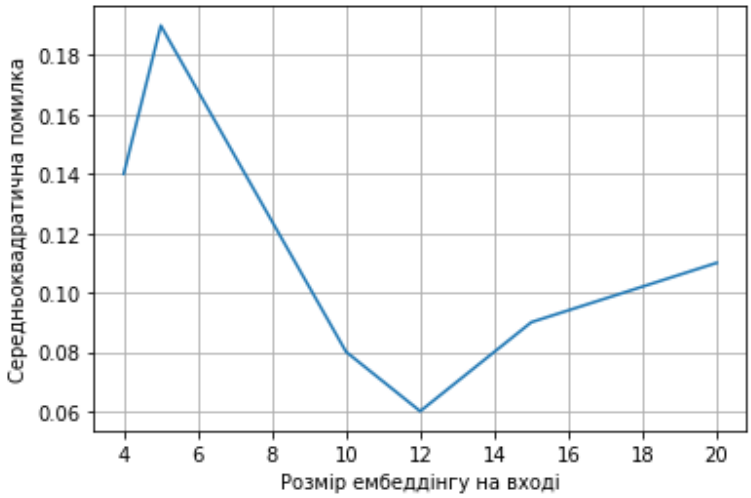


Рис. 3.12. Точність в залежності від ембеддінгу.

Також важливим питанням була кількість прихованих шарів з оптимальною кількістю нейронів. Тому що навішування нових шарів в нейромережу може як збільшувати точність, так і зменшувати її, особливо якщо річ йде про малі датасети. Нульовою гіпотезою є те, що одного шару нейронів достатньо, щоб оптимально вирішувати задачу, і якщо збільшувати кількість нейронів, буде відбуватись перенавчання, і точність буде падати. Щоб перевірити це, були проведені тести на 1,2,3 шарах, що складатись з 50 нейронів. Результат видно на наступному малюнку.

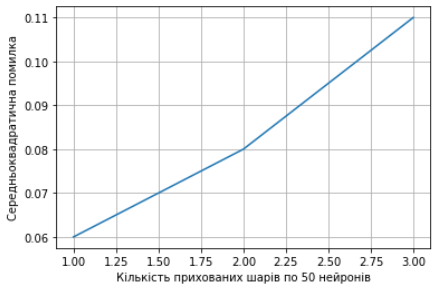


Рис. 3.13 Точність в залежності від кількості прихованих шарів.

З ростом кількості шарів та нейронів в них помилка тільки росла, і досягала локального мінімуму в точці 0.11, тобто, нульова гіпотеза повністю підтвердилась, справді, один шар і 50 нейронів – оптимальна кількість для даного датасету.

В останню чергу потрібно підібрати оптимальний дропаут, щоб мінімізувати перенавчання та оптимізувати роботу нейромережі. Дропаут перебирався з інтервалу від 0.4 до 0.8, і було помітно як падіння помилки від 0.09 до 0.06 при рості проценту дропауту від 0.4 до 0.6, так і різкий ріст помилки починаючи від 0.61 і закінчуючи 0.80, коли помилка стала близько 0.16. Відтак, можна стверджувати, що оптимальне значення дропауту для даної архітектури – 0.6, або 60% нейронів відключались кожну ітерацію.

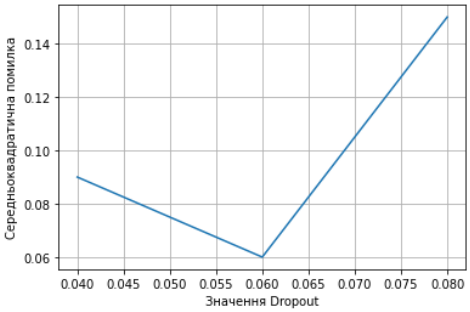


Рис. 3.14. Точність в залежності від дропауту.

Сумуючи проведені експериментальні дослідження нейромережі можна описати оптимальну архітектуру нейромережі для рішення конкретного датасету.

- Кількість нейронів – 50
- Кількість прихованих шарів – 1
- Розмір вектора ембеддінгу – 12
- Дропаут – 0.6%

З такою архітектурою отримується мінімальне значення середньоквадратичної помилки – 0.06, тому, можна зробити висновок, що для даного датасету абсолютним мінімумом є саме це значення – 0.06. Також можна виділити локальний мінімум функції – 0.11, саме при цьому значенні помилки часто наступав параліч нейромережі, коли вона зупинялась і осцилювала між двома значеннями, наприклад, 0.114 і 0.115. Щоб освіжити в пам’яті терміни «локальний мінімум» та «абсолютний мінімум» можна подивитись на наступний рисунок. Локальний мінімум – це мінімальна точка в певному околі біля значення функції, а глобальний мінімум – це найнижче значення функції взагалі. Якщо сприймати нейромережу як функцію, що перетворює вхідні дані в вихідні, то, при знаходженні локального мінімуму може наступити параліч мережі, досягнення якого не дозволить знайти глобальний мінімум функції. Саме таким глобальним мінімумом функції є 0.06 у випадку нейромережі, яка була реалізована, а 0.11 є хорошим прикладом локального мінімуму.

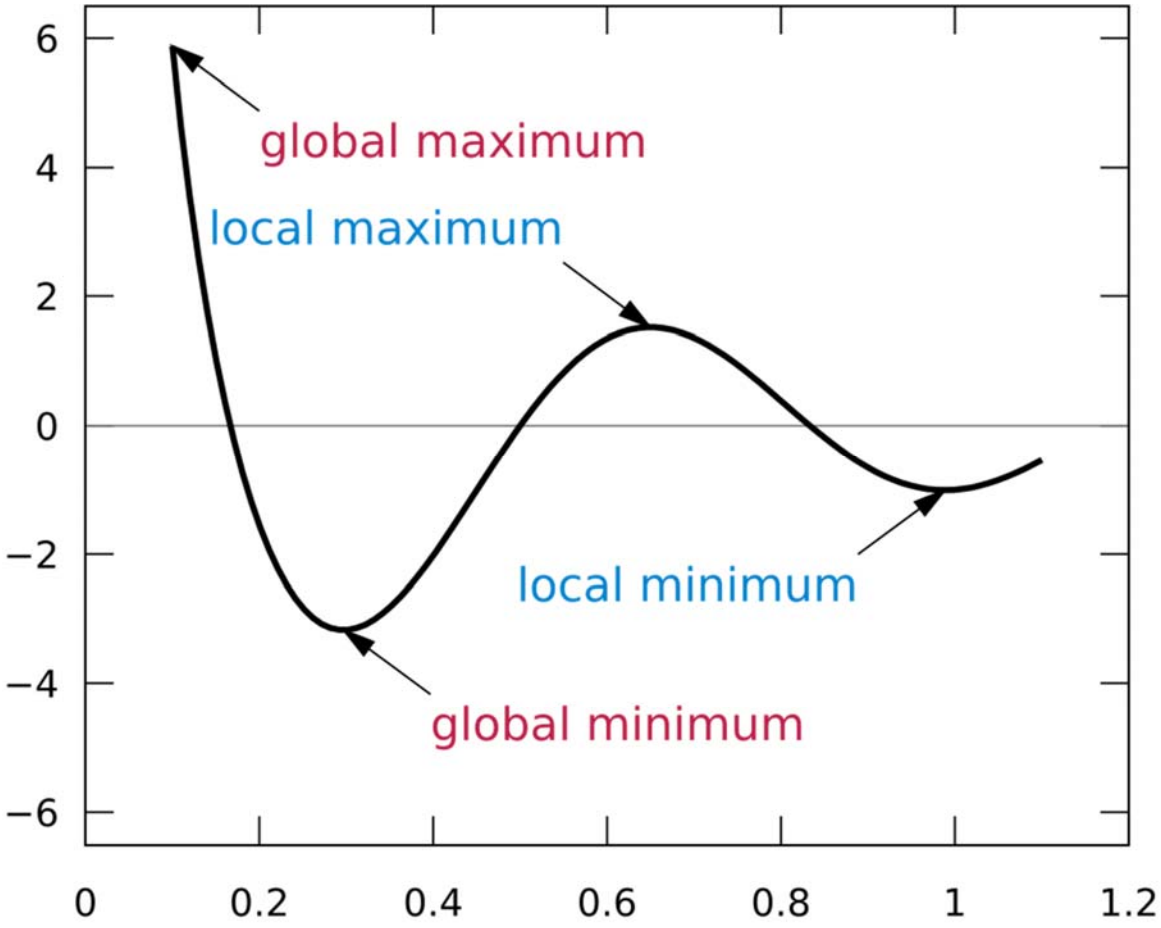


Рис. 3.15 Глобальні та локальні екстремуми функції.

Також варто зрівняти точність, яка була отримана з допомогою програмного продукту з тими значеннями точності, що отримали різні команди вчених в минулому. З наступного рисунку видно, що дана нейромережа має набагато меншу середньоквадратичну помилку, чим нейромережа, яка була запропонована командою Джіндаля при будь-якому розмірі датасету, і тільки трішки уступає нейромережі, яку запропонувала группа Отто, хоча і показує себе трішки гірше при великому розмірі датасету.

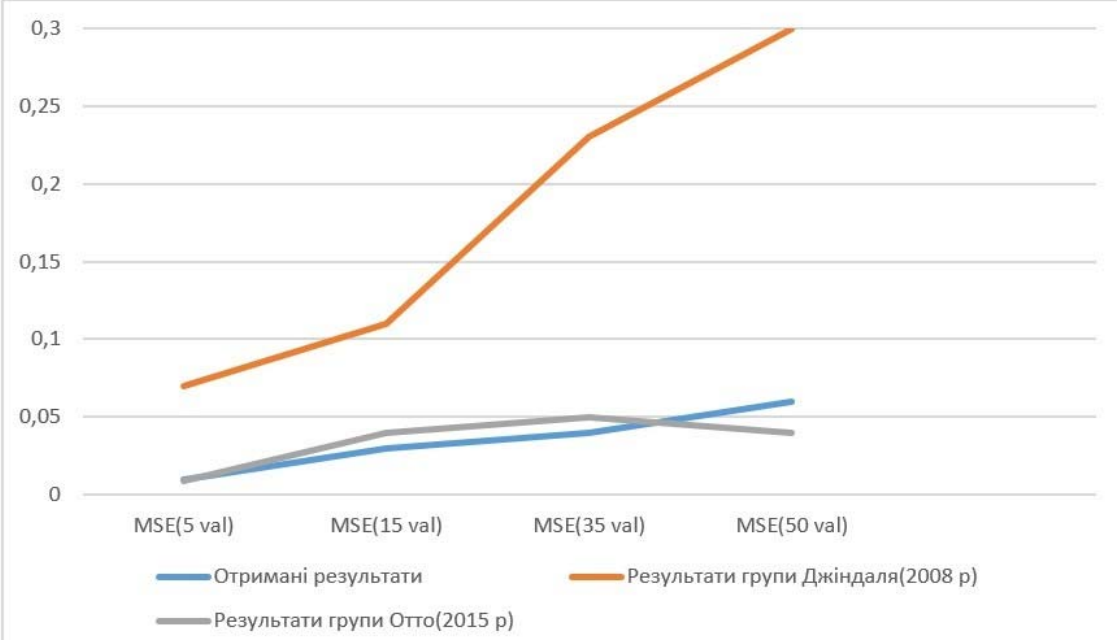


Рис. 3.16. Зрівняння точностей з існуючими результатами.

Висновки до розділу 3

Була написана нейромережа, яка може швидко і якісно відрізняти фейкові новини від справжніх та підтримувати актуальність інформації певного сервіс-орієнтованого сайту, на прикладі агрегатора новин, форума, новинного сайту чи інше. Нейромережа написана в функціональному стилі, тому її можна легко адаптовувати та масштабувати для вирішення конкретних задач. Через постійне використання unit-тестування були виловлені всі можливі баги, що і дозволило безпечно та оптимально використовувати програмний продукт. Отримана хороша точність, так як середньоквадратична помилка всього 0.06, що значно краще, чим ранні результати групи Джінжаля, середньоквадратична помилка в яких сягала 0.35-0.40. Звичайно, це пов'язано з покращенням архітектури нейромереж, використанню ембеддінгу та дропауту. Ці результати показують, що нейромережі – дуже перспективні для роботи з неактуальною інформацією,

і можуть прекрасно використовуватись для того, щоб фільтрувати інформацію та збільшувати рівень актуальності системи.

Також, варто виділити фактори, які можна покращити, щоб добитись серйозніших результатів.

- В першу чергу, це розмір датасету. 50 текстів – надто мало, щоб навчити універсальну мережу. Але рішення про роботу з маленьким датасетом було усвідомлено – потрібно проводити багато тестів, а великий датасет потребує значних комп'ютерних потужностей, які недоступні, тому, для нормального аналізу і було прийнято рішення задовільнитись маленьким датасетом. Для отримання справді універсальної нейромережі, яка зможе працювати з усіма типами текстів потрібно значно наростити датасет, і включити в нього велику кількість текстів – рекламних, медичних, загально-наукових та інших, в такий спосіб можна створити справді унікальний продукт для роботи з будь-яким текстом української мови.
- Словник стоп-слів. На етапі початку роботи виявилось, що в мережі практично немає достойних словників українських стоп-слів. Тому потрібно було створити його самостійно. Аналітично та шукаючи аналоги був створений словник українських стоп-слів, що складається з 1983 слів. Зрозуміло, що без додаткової освіти в сфері лінгвістики скласти достойний словник важно – тому цілком допускаємо, що кращий і актуальніший словник збільшить точність нейромережі та зменшить час обробки окремого датасету.
- Словник лемматизації. Лемматизація дуже важлива для нормальної роботи нейромережі, тому словник для неї – дуже важливий компонент будь-якої обробки натуральної мови. Нажаль, його точність залишає

бажати кращого, тому покращення словника лемматизації точно збільшить ефективність нейромережі та покращить результати. Словник лемматизації – це важка задача, яка знаходиться між лінгвістикою та інформатикою, і створення максимально глибокого словника дозволить кратно краще працювати з неактуальною текстовою інформацією.

- Покращення моделі UDPipe. В нейромережі була використана звичайна модель для аналізу синтаксичних структур тексту (лінк на неї, так як вона завантажується можна подивитись на рисунку), але це модель виявилась застарілою та не дуже ефективною. Це ще одна задача для лінгвістів – покращення такої моделі дозволить знову ж таки кратно збільшити ефективність нейромережі.

```
n [5]: import ufal.udpipe as udp
       import corpy.udpipe as crp
       import wget
       import os

       # Завантаження моделі UDPipe, що навчена на українській мові
       udp_model_url = r'https://lindat.mff.cuni.cz/repository/xmlui/bitstream/handle/11234/1-3131/ukrainian-iu-ud-2.5
       udp_model_filename = udp_model_url.rsplit('/', 1)[-1]
       if not os.path.isfile(udp_model_filename):
         wget.download(udp_model_url)

       # Загрузка моделі в оболонку
       corpy_model = crp.Model(udp_model_filename)
```

Рис. 3.17. Приклад завантаження моделі в оболонку.

- Використання екзотичних архітектур нейронної мережі. Справді, існує множина досить екзотичних архітектур нейромереж, які можуть показувати хороші результати. Наприклад, це архітектура по типу SpineNet, яка включає в себе перестановку прихованих шарів кожні X епох, вона дає дуже хороші результати, наприклад, при роботі з текстом, або архітектура DenseNet, яка заміняє прихований шар структурованим масивом нейронів, що і називається Dense, вона показує чудові результати в питання класифікації, і може

використовуватись для нашої задачі. Зрозуміло, що такі специфічні архітектури використовуються для конкретних задач та виходять за теми цієї роботи, але їх використання може значно збільшити точність результатів нейромережі.

Також, варто зауважити, що нейромережу можна вчити не тільки на звичайних текстах. Згадуючи першу главу даної роботи варто виділити, що вона може працювати з неактуальними форматами даних, такі як REST і SOAP, і розробник з базовими знаннями мови програмування Python може просто змінити список стоп-слів на англійський, базу UDPipe на англійську та легко працювати з даними, що вертають різноманітні сервіси. Так, можна використовуючи вищевказані метрики розмітити певних датасет, навчити на ньому нейромережу та в майбутньому швидко та зручно використовувати навчену нейромережу для оцінки актуальності програмних даних. Відповідно, такий підхід дозволить значно покращити тестування та роботу з застарілими сервісами, і, як наслідок, значно пришвидшить роботу розробника і дозволить ефективно боротись з неактуальними програмними даними. Тому що метрики дозволяють набагато краще зрозуміти те явище, що ми вивчаємо, але виділити глибокі патерни та використовувати їх можна тільки з допомогою нейромережі, відтак, можна зробити висновок – що найкращий спосіб знаходження та фільтрації неактуальної інформації – широке використання нейронних мереж, що дозволить оптимізувати роботу з неактуальною інформацію і значно покращити ефективність роботи в сервісно-орієнтованих мережах.

Окремо варто виділити особливості навчання нейромережі – так основна частина навчання нейромережі проходила за коротку кількість епох –

близько 50, після цього наступала черга досить призупинення навчання та маленьких осциляцій, які практично не впливали на точність. Питання ідеального learning rate та кількості ітерацій для точного навчання до сих пір залишаються досить спірними питаннями в сучасних нейронауках, тому вони не розглядались в даній роботі, так як, з моєї точки зору не так сильно впливають на точність нейромережі, як особливості архітектури, дропаут та кількість прихованих шарів, які буквально детермінують усі особливості навчання та поведінки штучної нейронної мережі.

ВИСНОВКИ

1. Питання актуальності інформації в даний час неймовірно важливо, воно буквально знаходиться в списку найбільш важливих питань та викликів, що стоять перед сучасною інформатикою. Це пов’язано, в першу чергу, з значним ростом мережі Інтернет, яка продовжує рости неймовірними темпами, з швидким збільшенням кількості наявних сайтів в мережі Інтернет, і з тим, що все більше людей включаються в цифровий світ. Тому, він все більше впливає на нас, як на користувачів, так і на розробників і системних спеціалістів (архітекторів, аналітиків). Цей вплив може бути дуже негативним, і єдиним способом покращити його – є фільтрування інформації та очистка мережі від застарілих сервісів та текстів.

2. Сервіс-орієнтована система – це веб-система, яка побудована на основі сервісно-орієнтованої архітектури, що ґрунтується на таких правилах, як модульний підхід до розробки програмного забезпечення, створенні слабко-пов’язаних компонентів, які використовують для зв’язку стандартні протоколи обміну даними та стандартизовані інтерфейси.

3. Термін «актуальна інформація» - інтеграційний, і складається з двох компонентів, програмного та користувацького. Програмний належить до сервісів, які передають неактуальну інформацію між собою та обробляють її, та до розробників, що можуть приймати неправильні рішення користуючись неактуальною інформацію. Користувацький – це вся неактуальна інформація, яку отримує користувач певного сервісу, в тому числі сервіс-орієнтованої системи.

4. Користувацька неактуальна інформація, включає в себе:

- Фейкові новини
- Фейкові відгуки

- Застарілу інформацію
- Хейт-спітч
- Маніпулятивну інформацію
- Помилкову інформацію
- Неякісну інформацію(неправильний формат файлу, поганий переклад та інше)

5. Програмна неактуальна інформація включає в себе:

- Неструктуровану інформацію
- Інформацію, якій не можна довіряти (через відсутність сертифікату, шифрування даних чи інше)
- Дані, що пов’язані з помилковим API.
- Можливість неоднакової семантики при роботі з сервісом
- Застаріла інформація, що отримана з API

6. Була проведена значна аналітична робота щодо опису різноманітних метрик для оцінки програмної сторони актуальності інформації та метрик, що використовуються для оцінки різноманітних патернів, що виникають в натуральних текстах, та описано як їх визначати, і що для цього можна використовувати скрипти, як готові так і самописні. Так, були описані такі показники як TF для оцінки тексту, та описана метрика, що була запропонована А. М. Мельником, М. П. Диваком, Р. М. Пасічником[1] для оцінки програмної сторони актуальності тексту

7. Запропонована проста та гнучка метрика для оцінки програмної та користувацької актуальності інформації – степ-метрика, яку кожна команда розробників може зібрати для свого використання, так як вона являє собою, по суті, блок-схему, та легко піддається масштабуванню, покращенню,

специфікації. Не дивлячись на свою простоту – степ-метрика – прекрасний інструмент для оцінки неактуальної інформації.

8. Було доведено, що для фільтрації неактуальної інформації та більш ефективної оцінки актуальності інформації потрібно використовувати штучні нейронні мережі, так як тільки вони можуть гарантувати потрібну гнучкість для цього завдання. Було описано як працює нейронна мережа, та які основні процеси в ній відбуваються, які є сучасні погляди на оптимізацію нейронної мережі, та як з нею працювати.

9. Проведений опис правил роботи з обробкою натуральної мови, дані пояснення таким термінам, як «стоп-слово», «лемматизація», «векторизація», та інші, був описаний алгоритм перетворення звичайного тексту в вектор, що дозволяє передати його на вхід нейронній мережі. Також, був створений датасет для нейронної мережі, словник стоп-слів для роботи з натуральним текстом.

10. Був написаний програмний продукт, який заснований на нейронній мережі. Цей ПЗ отримує на вхід файл формату .csv з набором текстів та бінарним класифікатором для кожного тексту, після цього проводить його обробку та векторизацію, після цього відбувається навчання нейронної мережі та збереження вагів нейромережі в форматі .h5. Після цього нейромережу можна використати для того, щоб оцінити рівень фейковості певного тексту, який передати на неї. Звичайно, датасет можна змінювати і таким чином вчити нейромережу знаходити інші патерни даних, так її можна навчити опрацьовувати дані, що отримані з API

11. Були проведені тести нейромережі для того, щоб підібрати оптимальні показники архітектури – кількість нейронів, кількість прихованих шарів, розмір дропауту та розмір ембеддінгу. В результаті була підібрана оптимальна

архітектура нейронної мережі для рішення конкретного датасету, що був зібраний з допомогою парсінгу

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. А. М. Мельник, М. П. Дивак, Р. М. Пасічник, МЕТОД ВИЯВЛЕННЯ НЕАКТУАЛЬНОЇ ІНФОРМАЦІЇ В СЕРВІСНО-ОРІЄНТОВАНИХ КОРПОРАТИВНИХ СИСТЕМАХ НА ПРИКЛАДІ СИСТЕМ ОЦІНЮВАННЯ ЯКОСТІ ГРУНТІВ, ISSN 1999-9941, “ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА КОМП’ЮТЕРНА ІНЖЕНЕРІЯ”, 2021, № 1
2. S. A. García, G. G. García, M. S. Prieto, A. J. M. Guerrero, and C. R. Jiménez, «The impact of term fake news on the scientific community scientific performance and mapping in web of science», Social Sciences, vol. 9, no. 5, 2020.
3. S. Akhtar, F. Hussain, F. R. Raja et al., «Improving mispronunciation detection of arabic words for non-native learners using deep convolutional neural network features», Electronics, vol. 9, no. 6, 2020.
4. Ahmed, S., Hinkelmann, K., Corradini, F., «Combining machine learning with knowledge engineering to detect fake news in social networks − a survey», In: Proceedings of the AAAI 2019 Spring Symposium, vol. 12 (2019).
5. B. Marr, Coronavirus fake news: how Facebook, Twitter, and Instagram are tackling the problem. Forbes (2020). [Online]. Available: https://www.forbes.com/sites/bernardmarr/2020/03/27/finding-the-truth-about-covid-19-how-facebook-twitter-and-instagram-are-tackling-fake-news.
6. H. Sparks, H. Frishberg, Facebook gives step-by-step instructions on how to spot fake news (2020). [Online]. Available: https://nypost.com/2020/03/26/facebook-gives-step-by-step-instructions-on-how-to-spot-fake-news/
7. Zhou, X., Zafarani, R.: Fake news: a survey of research, detection methods, and opportunities (2018). arXiv preprint arXiv:1812.00315.