НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ТЕЛЕКОМУНІКАЦІЙ

Кафедра Мобільних та відеоінформаційних технологій

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«Розробка системи моніторингу Інтернет-трафіку та її впровадження в мережеві технології»

Виконав: студент 6 курсу, групи РТДМ-61

спеціальності 172 Телекомунікації та радіотехніка

(шифр і назва спеціальності)

Тарганчук О.В.

(прізвище та ініціали)

Керівник Кравченко В.І.

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Нормоконтроль _____

(прізвище та ініціали)

Київ - 2019

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ТЕЛЕКОМУНІКАЦІЙ

Кафедра Мобільних та відеоінформаційних технологій
Ступінь вищої освіти Магістр
Спеціальність 172 Телекомунікації та радіотехніка
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Мобільних та відеоінформаційних технологій
_____ В.І. Кравченко

«___» _____ 2019 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Тарганчуку Олексію Володимировичу

1.Тема роботи:«Розробка системи моніторингу Інтернет-трафіку та її впровадження в мережеві технології», керівник роботи Кравченко Владислав Ігорович, к.т.н., завідувач кафедри МВТ затверджені наказом вищого навчального закладу від 14.11.2019 року року № 518.

2. Строк подання студентом роботи 20.12.2019 р.

3. Вихідні дані до роботи:

1. Доцільність прогнозування трафіку мережі.
2. Розпізнавання мережевими технологіями.
3. Захист даних.
4. Програмна реалізація.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

1. Системи моніторингу Інтернет-трафіку.
2. Дослідження аналізу застосовності до прогнозування.
3. Впровадження систем безпеки в технологію системи моніторингу.

5. Перелік графічного матеріалу:

1.Мета роботи;

2.Загальна структура системи моніторингу інтернет-трафіку;

3.Графік моніторингу вхідного трафіку світового ресурсу;

4.Моніторинг каналів зв'язку;

5.Система моніторингу поділяється на такі види: бездротові та дротові;

6.Приклад графіків «Інтернет-трафіку»в штатному режимі;

7.Приклад графіку «Інтернет-трафіку»в надзвичайній ситуації;

8.Висновки.

6. Дата видачі завдання 05.09.2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Підбір науково-технічної літератури	09.09.19	
2.	Системи безпровідного зв’язку	11.09.19	
3.	Дослідження впливу модуляцій та кодування сигналу на ефективність безпровідного систем	23.09.19	
4.	Вплив застосування канального кодування на підвищення ефективності безпровідного систем	07.09.19	
5.	Висновки, вступ, реферат	21.10.19	
6.	Розробка презентації	04.11.19	

Студент _____ Тарганчук О.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Кравченко В.І.
(підпис) (прізвище та ініціали)

Доповідь

Пояснювальна записка до дипломного проекту «Система моніторингу Інтернет-трафіку» містить 84 с., 41 рис., 13 літературних джерел, 1 додаток.

НЕЙРОННА МЕРЕЖА, ТЕСТУВАННЯ, КОМП'ЮТЕРНА МЕРЕЖА, ТРАФІК, PYTHON.

Дипломний проект присвячений розробленню нейронної мережі прогнозування станів трафіку комп'ютерної мережі на основі інтернет технологій та системних плат, дана розробка є актуальним завданням.

Об'єкт дослідження – нейронна мережа прогнозування трафіку комп'ютерної мережі.

Предмет дослідження – моделювання нейронної мережі для аналізу трафіку та його прогнозування.

Мета дипломного проекту – побудувати систему, що зможе коректно прогнозувати стан трафіку в мережі.

Метод дослідження – аналіз існуючих варіантів створення та прототипів, розробка алгоритмів роботи підсистеми, збору даних, системи зв'язку.

Установлено, що розроблений сервіс забезпечує:

1) спрощення процесу передбачення помилок та сбоїв;
2) зниження часу на аналіз трафіку;
3) дистанційну комунікацію користувача і підсистеми.

Актуальність – розвиток нових технологій та популярність нейромережевих технологій в наш час

Матеріали дипломного проекту рекомендуються використовувати в компаніях при обслуговуванні комп'ютерної мережі.

Прогнозні припущення щодо розвитку об'єкта дослідження – створення інтерфейсу та підвищення точності прогнозів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

NN	Neural Network (нейронна мережа)
SQL	Structured Query Language (мова структурованих засобів)
IDE	Integrated Development Environment (вбудоване середовище розробки)
ORM	Object-Relational Mapping (об'єктно-реляційне відображення)
URL	Uniform Resource Locator (уніфікований покажчик ресурсу)
WEB	World Wide Web (всесвітня мережа)
IP	Internet Protocol (інтернет протокол)
GPIO	General-Purpose Input/Output(інтерфейс зв’язку компонентів)
НМ	Нейронна мережа
БД	База даних
ПК	Персональний комп’ютер
АРМ	Автоматичне робоче місце
ТЗ	Технічне завдання

ВСТУП

Розроблена система є актуальною, адже вона надає кожному українському провайдеру в будинках, приватних установах не відставати від сучасних технологій, які можуть використовувати свої функції за допомогою нейронної мережі:

1) прогнозування;
2) аналіз;
3) кластеризація.

Інтернет мережа реалізовується за допомогою програмного комплексу модулів та системних структур. В інтересах розробленого проекту було проведено дослідження стандартів розробки архітектури. Змодельовано алгоритми: роботи мережі, збору даних, системи зв'язку за прямої необхідності покращення існуючих розробок. Хоча розвиток цих алгоритмів в системах NN проводиться більше 10 років, він є актуальним і зараз, оскільки з розвитком Internet-технологій, їх характеристики потребують змін.

Проектовану систему запропоновано розробити за допомогою Python. Цей продукт дозволяє поєднати різноманітні програмні технології: C++, PHP , SQL, HTML.

Ця тематика дипломного проекту відповідає цілям та завданням кафедри, щодо розробки фахівців в галузі систем та мереж і включає знання з пройдених дисциплін.

Мета дипломного проекту – побудувати систему, що зможе коректно прогнозувати стан трафіку в мережі.

- спростити процес доступу до інформації;
- налагодити коректну передачу даних з датчиків;
- реалізувати запропоновану нейронну мережу.

Об'єкт дослідження – нейронна мережа прогнозування трафіку комп'ютерної мережі.

Предмет дослідження – моделювання мережі для аналізу трафіку та його прогнозування.

Метод дослідження – аналіз існуючих варіантів створення та прототипів, розробка алгоритмів роботи підсистеми, збору даних, системи зв'язку.

Матеріали дипломного проекту рекомендуються використовувати в компанія при обслуговуванні комп'ютерної мережі.

Змодельовано Інтернет-мережу, розроблено алгоритм обробки даних, написано програмний код, створено діаграму класів, прецедентів та компонентів

Практичне значення результатів дає змогу користувачам розробленої підсистеми, регулювати трафік залежно від прогнозів які зробила нейронна мережа

Потенціал систем для українського ринку важко переоцінити. Використання концепції дозволяє досягти відчутного полегшення в керуванні трафіком, енергозбереженні та розумінні ризиків.

При виборі теми дипломного проекту, я зупинився саме на цій. Оскільки розроблена система може не тільки показати всі навички, набуті за період навчання, а й бути корисною в майбутньому.

РОЗДІЛ 1. ДОЦІЛЬНІСТЬ ПРОГНОЗУВАННЯ ТРАФІКУ МЕРЕЖІ

1.1 Базові поняття комп’ютерної мережі

Комп'ютерні мережі. Інформаційно-комунікаційні технології, що з'явилися у другій половині XX ст., суттєво змінили життя людства. Саме вони створили передумови формування інформаційного суспільства, в якому визначальну роль відіграють інформація та нові знання. Перші ЕОМ були призначені лише для швидкої обробки числових даних. Згодом обчислювальна техніка стала широко використовуватися в наукових дослідженнях, виробництві, освіті, побуті тощо. У користувачів віддалених один від одного комп'ютерів з'явилася потреба у швидкому обміні даними. Для цього було запропоновано об'єднати комп'ютери в єдину систему і таким чином передавати дані від одного комп'ютера до іншого. Так були створені комп'ютерні мережі. Комп'ютерна мережа – це сукупність комп'ютерів та інших пристроїв, зв'язаних каналами передавання даних.

Комп'ютерні мережі забезпечують спільний доступ до даних. У мережі виділяють комп'ютери, на яких розміщують великі масиви даних, а користувачі інших комп'ютерів мережі одержують доступ до них. Це дає можливість, наприклад, людям, котрі працюють над одним проектом, використовувати дані, створені іншими, тобто працювати над проектом одночасно.

За допомогою комп'ютерної мережі стає можливим спільне користування периферійними пристроями: принтерами, сканерами, модемами тощо. Невигідно мати їх біля кожного персонального комп'ютера, наприклад, у комп'ютерному класі або в банку.

Комп'ютерні мережі також дозволяють у короткі терміни розв'язувати складні інженерні задачі (прогнозування стихійних лих, проектування аерокосмічих апаратів, обробка знімків Землі, отриманих зі супутників, моделювання й аналіз перимент в у фізиці тощо).

Існуючі мережі прийнято в даний час ділити в першу чергу за територіальною ознакою.

1) Локальні мережі LAN. До локальних мереж зазвичай відносять мережі, комп'ютери яких зосереджені на відносно невеликих територіях (менше 2000 м). Прикладом локальної мережі є мережа малого підприємства, розташованого в одному або декількох будівлях. Невеликий розмір локальних мереж дозволяє використовувати для їх побудови досить дорогі і високоякісні технології, що забезпечує високу швидкість обміну інформацією між комп'ютерами.

2) Глобальні мережі WAN – побудована на основі комутованих або виділених каналів існуючих мереж або GAN побудована на основі використання супутникових та наземних ліній зв'язку. До глобальних відносять мережі, призначені для об'єднання окремих комп'ютерів і локальних мереж, розташованих па значній відстані (сотні і тисячі кілометрів) один від одного. Оскільки організація спеціалізованих високоякісних каналів зв'язку великої протяжності є досить дорогою, то в глобальних мережах нерідко використовуються вже існуючі і спочатку не призначені для побудови комп'ютерних мереж лінії (наприклад, телефонні або телеграфні). Невигідно мати їх біля кожного персонального комп'ютера, наприклад, у комп'ютерному класі або в банку. У зв'язку з цим швидкість передачі даних в таких мережах істотно нижче, ніж в локальних.

3) Регіональні мережі MAN. Подібні мережі існують в межах певного регіону (міста, району). Кожна така мережа є частиною деякої глобальної мережі і особливою специфікою, але відношенню до глобальної мережі не відрізняється. Однак для побудови таких мереж використовуються досить якісні цифрові лінії зв'язку, що дозволяють здійснювати взаємодію на відносно високих у порівнянні з глобальними мережами швидкостях. Види комп’юторних мереж, які бувають локальні і глобальні представлено на рис. 1.1.

Рис. 1.1 – Види комп’ютерних мереж

Архітектура має на увазі представлення мережі у вигляді системи елементів, кожен з яких виконує окрему функцію, при цьому всі елементи разом погоджено вирішують загальну задачу взаємодії комп'ютерів. Іншими словами, архітектура мережі відображає декомпозицію загального завдання взаємодії комп'ютерів на окремі підзадачі, які повинні вирішуватися окремими елементами мережі. Одним з важливих елементів архітектури мережі є комунікаційний протокол – формалізований набір правил взаємодії вузлів мережі.

Проривом в стандартизації архітектури комп'ютерної мережі стала розробка моделі взаємодії відкритих систем (Ореп System Interconnection, OSI), яка на початку 80-х років узагальнила накопичений на той час досвід. Модель OSI є міжнародним стандартом і визначає спосіб декомпозиції завдання взаємодії «по вертикалі», доручаючи це завдання комунікаційним протоколам семи рівнів. Рівні утворюють ієрархію, відому як стек протоколів, де кожен вищий рівень використовує нижчий рівень як зручний інструмент для вирішення своїх завдань.

Стеки протоколів, що існують сьогодні (або що існували ще недавно) в цілому відображають архітектуру моделі OSI. Проте в кожному стеку протоколів є свої особливості і відмінності від архітектури OSI. Так, найбільш популярний стек TCP/IP складається з чотирьох рівнів.

Стандартна архітектура комп'ютерної мережі визначає також розподіл протоколів між елементами мережі - кінцевими вузлами (комп'ютерами) і проміжними вузлами (комутаторами і маршрутизаторами). Проміжні вузли виконують лише транспортні функції стека протоколів, передаючи трафік між кінцевими вузлами. Кінцеві вузли підтримують весь стек протоколів, надаючи інформаційні послуги, наприклад веб – сервіс. Такий розподіл функцій означає зсув «інтелекту» мережі на її периферію.

Існує три основні типи топологій мережі:

1) мережева топологія шина (bus), при якій всі комп'ютери паралельно підключаються до однієї лінії зв'язку і інформація від кожного комп'ютера одночасно передається всім іншим комп'ютерам (рис. 1.2);

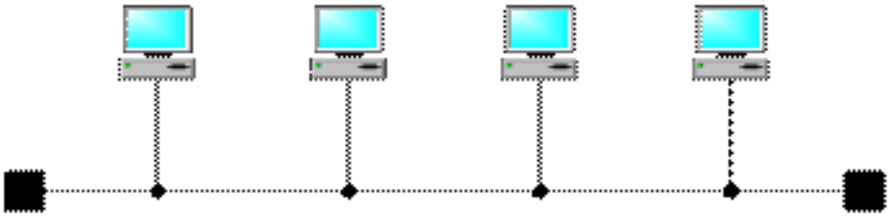

Рис. 1.2 – Мережева топологія «шина»

2) мережева топологія зірка (star), при якій до одного центрального комп'ютера приєднуються інші периферійні комп'ютери, причому кожен з них використовує свою окрему лінію зв'язку (рис. 1.3);

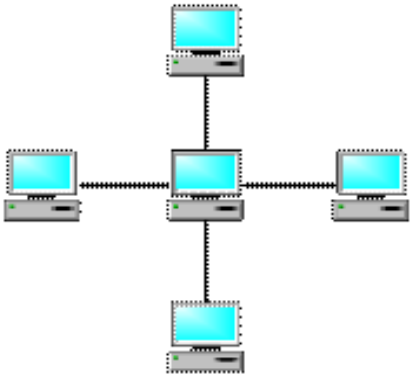

Рис. 1.3 – Мережева топологія «зірка»

3) мережева топологія кільце (ring), при якій кожен комп'ютер передає інформацію завжди тільки одного комп'ютера, наступного ланцюжку, а отримує інформацію тільки від попереднього комп'ютера в ланцюжку, і цей ланцюжок замкнута в «кільце» (рис. 1.3). Що є основною робочею мережевою топологією у компаніях.

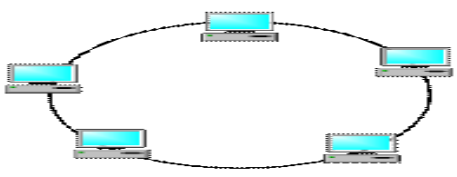

Рис. 1.4 – Мережева топологія «кільце»

Головне функціональне призначення мереж – використання різноманітних інформаційних ресурсів користувачами з різних організацій, міст, країн.

Локальні мережі призначені для використання в межах одного приміщення чи однієї організації. Глобальні ж мережі створюються для з'єднання комп'ютерів, що розташовані на значних відстанях один від одного. Локальні мережі поділяються на однорангові та багаторангові. В однорангових мережах всі користувачі мають однакові права. Користувачі такої мережі можуть здійснювати обмін даних між собою, використовувати спільні ресурси (принтери, диски і т.д.) Прикладом такої мережі може служити мережа на базі операційної системи Windows95.

Глобальні мережі поділяються на регіональні та міжнародні. Регіональні мережі призначені для використання користувачами певного регіону. В Україні існує декілька мереж регіонального призначення – УкрПак, мережа податкової адміністрації, залізниці, УМВС та інш. Глобальні мережі мають користувачів у всьому світі. Існує декілька загальновідомих всесвітніх мереж. Це такі мережі як: FidoNet, InterNet, EuroNet, система міжбанківських розрахунків SWFIT. Широке розповсюдження отримала в країнах колишнього Радянського Союзу мережа RelCom. Однак найвідомішою з них є всесвітня мережа InterNet – найбільша глобальна комп'ютерна мережа, що зв'язує десятки мільйонів абонентів у більш як 170 країнах світу.

Швидкість передавання даних мережею – це кількість бітів даних, що можуть бути передані за одну секунду. У перших мережах швидкість становила кілька кілобітів за секунду. Серед комп'ютерних мереж розрізняють мережі з середньою швидкістю (від 1 до 10 Мбіт/с), високошвидкісні (до 100 Гбіт/с), надшвидкісні (не менше 1 Гбіт/с). Сучасні розробки наближають цей показник до 100 Гбіт за секунду.

Кабельними мережами передаються електричні або оптичні (світлові) сигнали, бездротовими – інфрачервоні або радіосигнали. Яким би не був сигнал, він слабішає в мережі і може бути загубленим, якщо його не підсилити. Для мережі визначають максимальну відстань між пристроями, на яку сигнал передається без спотворення. Для різних середовищ передавання даних

максимальна відстань передавання даних без підсилення сигналу становить від 10 м (інфрачервоний зв’язок) до 100 км у мережах на оптоволоконному кабелі або декількох тисяч кілометрів при використанні супутникових каналів зв’язку.

Вита пара – це тип мідного кабелю з такою будовою. Дроти кабелю розбито на пари. Пара дротів утворює ланцюг, яким передають дані. Дроти пари переплетено для захисту від перешкод-шумів, породжених сусідніми парами дротів кабелю. Пари дротів укладено в кольорову пластикову ізоляцію і сплетено воєдино. Пучки кручених пар захищає зовнішнє обплетення. Такий кабель використовують у телефонному зв'язку і у більшості мереж Ethernet. Швидкість передачі даних від 10 Мбіт/с до 1 Гбіт/с

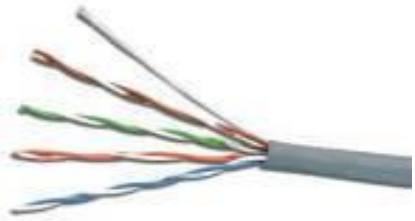

Рис. 1.5 – Кабель «вита пара»

Коаксіальний кабель – це кабель з ізольованою мідною жилою, оточеною металевою оболонкою-екраном. Коаксіальний кабель використовують для підключення комп'ютерів у мережі та поширення сигналів телебачення. Швидкість передачі даних до 10 Мбіт/с.

Рис. 1.6 – Коаксіальній кабель

Оптоволоконний кабель – сучасний і найшвидший мережний кабель – складається із напівпрозорих скляних чи пластикових волокон, кожне з яких тонше за людську волосину. Швидкість передачі даних від 10 Мбіт/с до 2 Гбіт/с.

Рис. 1.7 – Оптоволоконний кабель

Мережева операційна система є "мозком" мережі і забезпечує коректну взаємодію її програмного та апаратного забезпечення. Мережеві операційні системи (ОС) поділяються на дві категорії: однорангові і клієнтісерверні. Однорангові операційні системи передбачають можливість використання будь-якого комп'ютера як робочої станції і сервера одночасно. В однорангових мережах мережеві ОС (LANtastic, LanSmart, Windows for Workgroups тощо) інсталюються на кожному комп'ютері, у цьому разі назва мережі – це похідна від операційної системи, що утворює однорангову мережу. Таким чином, кожний із комп'ютерів отримує можливість надавати свої ресурси всім іншим комп'ютерам у мережі. Продуктивність однорангових мереж значно знижується за збільшення розмірів мережі і збільшення кількості взаємодій мережевих комп'ютерів. Експлуатація і підтримка таких мереж, як правило, ускладнена. Не маючи можливості централізованого управління, адміністратори змушені керувати множиною сервісів на кожній машині окремо. Така робота ускладнюється ще й тим, що користувачі, працюючи на кожному з комп'ютерів, мають можливість самостійно змінювати настройки ОС, що часто призводить до непрацездатності всього програмного забезпечення робочої станції.

У мережах клієнт/сервер мережна ОС (Windows 95/98, Windows 2000, Windows NTt Windows XP, Windows Millennium, Novell NetWare, UNIX тощо) установлюється на сервері. Цей комп'ютер керує мережею і надає свої ресурси клієнтським робочим станціям. Мережева ОС, працюючи на сервері (серверна ОС), відповідає за координацію всіх дій, пов'язаних із використанням ресурсів і сервісів цього сервера. Клієнтом у такій мережі є будь-який мережевий пристрій, що формує запит до сервера для використання його ресурсів і сервісів. Для забезпечення взаємодій клієнта і сервера на комп'ютер і-клієнті встановлюється і

функціонує клієнтське програмне забезпечення, яке підтримує загальний протокол взаємодії клієнта і сервера. У клієнт/серверній мережі користувачі "реєструються" зі своєї робочої станції. Для реєстрації користувач повідомляє серверові своє ім'я і пароль, якщо ім'я і пароль коректні, сервер аутентифікує користувача і надає йому доступ до всіх тих ресурсів і сервісів (використання файлів і принтерів, забезпечення безпеки даних і надання можливостей мережевої взаємодії), на які користувачу були надані права. Серверна ОС гарантує надійність і безпеку будь-яких даних, що зберігаються і опрацьовуються на сервері.

Мережева операційна система дає змогу користувачам спільно використовувати: дорогі апаратні ресурси мережі – принтери, сканери, дискові накопичувані тощо; програмне забезпечення, інстальоване тільки на сервері; інформаційні ресурси – базу даних сервера; організувати сумісну роботу великого колективу користувачів з оперативним обміном інформації між ними. До складу сучасних операційних систем (Windows XP, Windows 2000, Windows NT Server, Net Ware) входять компоненти: керування локальними ресурсами комп'ютера; серверна частина для надання власних ресурсів і послуг у загальне користування; клієнтська частина операційної системи для розпізнавання і переспрямовування в мережу запитів до віддалених ресурсів; комунікаційні засоби для обміну повідомленнями в мережі.

Програма переспрямування резидентно міститься в пам'яті комп'ютера. Коли користувач або його програма звертається із запитом до операційної системи комп'ютера, ця програма перехоплює запит, аналізує, хто може його виконати, і спрямовує або до ОС того ж комп'ютера, або до сервера, якому адресовано запит, показано їх на табл. 1.1.

Таблиця 1.1

Список деяких мережевих операційних систем із вказівкою їх виробників

Операційна система	Виробник
Apple Talk	Apple
LANtastic	Artisoft
NetWare	Novell
NetWare Lite	Novell
Personal NetWare	Novell
NFS	Microsystems
OS / 2 LAN Manager	Microsoft

Додатки абонентів можуть генерувати трафік, що перевищує пропускну здатність каналів зв'язку. Все це призводить до виникнення перевантажень на ділянках мережі, а внаслідок цього до порушення цілісності, виникнення загроз втрати даних, помилок, відмов в обслуговуванні і уповільненою роботі в мережі.

1.2.Базові поняття трафіку комп’ютерної мережі

Мережевий трафік або інтернет-трафік – обсяг інформації, що передається через комп'ютерну мережу за певний період часу (рис. 1.8). Кількість трафіку вимірюється як в пакетах, так і в бітах, байтах і їх похідних: кілобайт (КБ), мегабайт (МБ).

Трафік поділяється на:

– вихідний (інформація, яка надходить в зовнішню мережу);
– вхідний (інформація, яка надходить із зовнішнього мережі);
– внутрішній (в межах певної мережі, найчастіше локальної);
– зовнішній (за межами певної мережі, найчастіше - інтернет-трафік).

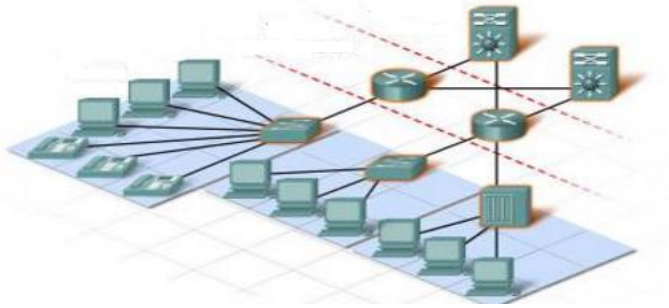

Рис. 1.8 – Передача трафіку в мережі

Програми, які здійснюють підрахунок мережевого трафіку: TMeter, BWMeter, NetWorx, DU Meter, NetTraffic, NetBalancer.

За даними доповіді аналітиків з британської компанії СімілярУеб (квітень 2015), найбільший обсяг інтернет-трафіку в світі дають прямі переходи на сайти з браузера (43,4%). За ними слідують пошукові системи з часткою 27,8%, посилання з інших ресурсів дають додаткові 21,13%. Соціальні мережі мають відносно невелику частку в 5,8%, а на частку переходів з електронної пошти і різної реклами припадає трохи більше 1%.

Частка населення різних країн в генерації інтернет-трафіку істотно різниться. У рейтингу, опублікованому експертами СімілярУеб в квітні 2015 року, США (25,05%), Великобританія (5,51%) і Російська Федерація (5,05%) є трьома найбільшими генераторами інтернет-трафіку в світі. В першу десятку також потрапили Бразилія (4,4%), Франція (3,9%), Німеччина (3,5%), Індія (3,4%), Канада (3,2%), Японія (2,6 %) і Туреччина (2,5%). Слід зазначити, що інтернет-трафік з КНР не враховувався в силу діючих обмежень на користування цілим рядом світових загальнодоступних інтернет-ресурсів.

У перші роки існування інтернету його оплата стягувалася відповідно до кількості прийнятої та переданої інформації. Для того, щоб підключити до

інтернету одного користувача, інтернет-провайдеру доводилося йти на досить серйозні витрати, які окупалися тільки з абонентської плати клієнтів.

Тому інтернет оплачувався за тим же принципом, що і вода або електроенергія – за лічильником. Потім з'явилися лімітовані тарифи, які встановлювали «оптові» ціни на трафік, причому перевищення ліміту оплачувалося по істотно більш високою ціною. Такі тарифи і зараз іноді зберігаються у деяких операторів бездротового інтернету (рис. 1.9).

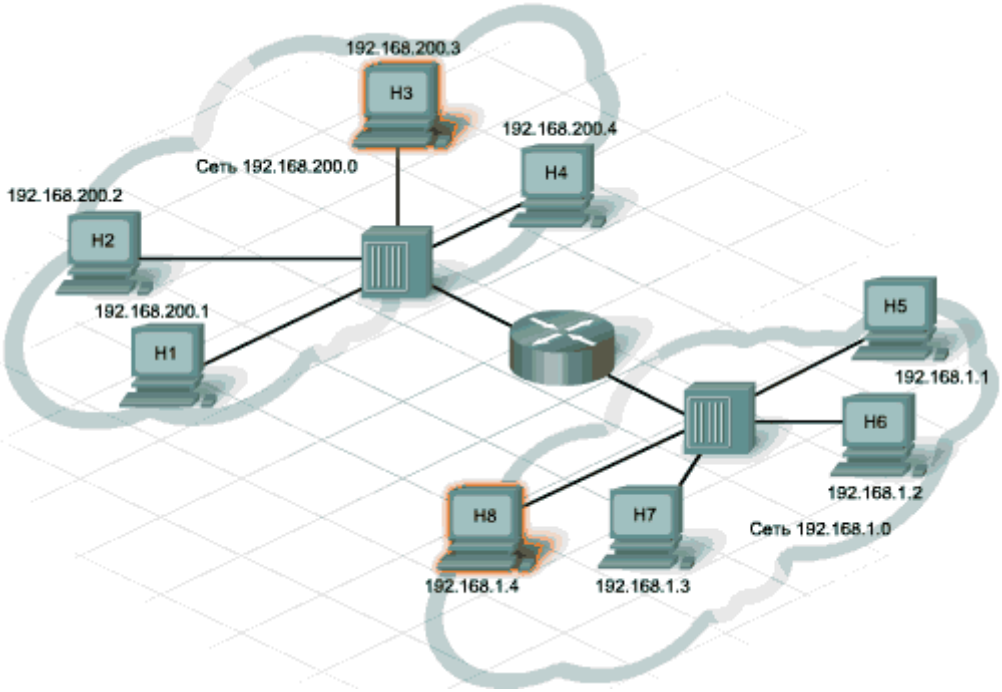

Рис. 1.9 – Приклад роботи лімітованого трафіку

Але оплата трафіку – це сторона, яка стосується користувачів. З точки зору провайдерів і програмістів трафік – це ресурс, користування яким потребує грамотного обслуговуванні. Пропускні можливості інтернет-мереж і серверного устаткування не безмежні.

Для того, щоб у провайдера не виникало необхідності постійно замінювати прокладені кабелі, він повинен враховувати перспективи зростання трафіку, причому як за рахунок збільшення користування інтернетом з боку вже підключених клієнтів, так і за рахунок появи нових підключень.

Таким же чином організовується робота інтернет-сайтів і порталів, які розраховані на певний рівень відвідуваності, тобто трафік. Якщо він перевищується, сайт «падає».

До сих пір деякі інтернет-провайдери надають послуги лімітованого трафіку. Найчастіше це стосується мобільного інтернету, що надається через стільникові мережі.

Іноді трапляється, що місячний обсяг трафіку закінчується, а до кінця місяця ще залишилося кілька днів. Залишатися хоча б один день без інтернету сьогодні не хочеться нікому – це незручно, а для багатьох ще й звужує їх професійні можливості.

Якщо у вас закінчився трафік, потрібно одразу ж зв'язатись з вашим оператором (провайдером) і обговорити умови надання трафіку понад включеного ліміту. Як правило, ця послуга виходить істотно дорожчий, але іноді просто немає іншого виходу.

Різні провайдери можуть надавати пакети трафіку на добу, тиждень або просто порахувати кількість мега- або гігабайт, які ви використовуєте до кінця місяця, і внести їх оплату в наступний рахунок за послуги інтернету.

Мобільним називають інтернет-трафік, який організовується за допомогою бездротових мереж мобільного (стільникового) зв'язку. Оскільки він залежить від пропускної здатності дорогого стільникового обладнання, можливості мобільного інтернету, як правило, обмежені в порівнянні з кабельним.

Мобільний трафік використовують смартфони, планшети, ноутбуки, комунікатори та інші комп'ютерні пристрої, які зручно носити з собою.

Оператори, що надають безлімітний мобільний інтернет, насправді теж обмежують його користування, однак трафік зменшується в цьому випадку за допомогою зниження швидкості обміну пакетами даних. Якщо ви завантажили за один день кілька фільмів або ігор, то можете помітити, що швидкість мобільного інтернету у вас падає. На рис. 1.10 представлений приклад передачі даних при скачуванні.

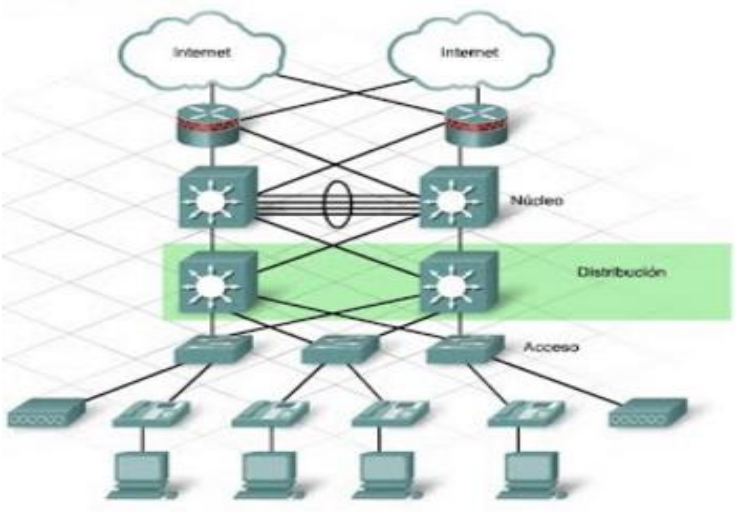

Рис. 1.10 – Приклад передачі даних

Як правило, мобільні компанії відстежують добовий трафік користувачів і регулюють його таким ось непомітним способом.

Компанія Google відстежує трафік кожного користувача, і в тому випадку, якщо його комп'ютер проявляє нехарактерну активність, видає повідомлення про «підозрілу трафіку».

Це означає, що з вашого комп'ютера йде або розсилка запитів, або пряме перекачування великих об'ємів інформації. І те, і інше може бути результатом роботи комп'ютерних шахраїв або діяльністю вірусу. Нерідко «підозрілим трафіком» є ваша пошукова діяльність, коли ви відправляєте в одиницю часу багато пошукових запитів. Тоді ви просто відповідаєте на запропонований перевірки питання Google і працюєте далі.

Якщо ж ви не займаєтеся пошуком, але побачили повідомлення про підозрілий трафіку – бажано відключити пристрій від інтернету і перевірити його спеціальної антивірусною програмою.

1.3.Прогнозування трафіку комп’ютерної мережі

Основна вимога до комп'ютерних мереж з погляду якості обслуговування – це виконання мережею її основної функції – забезпечення користувачам потенційної можливості доступу до всіх ресурсів комп'ютерів у мережі. Всі інші

вимоги – продуктивність, надійність, сумісність, керованість, захищеність, розширюваність і масштабованість – тісно пов'язані з якістю виконання цієї основної задачі. Вузлове обладнання пакетних мереж є високовартісним. З огляду еконо- мічної доцільності, необхідно забезпечувати ефективне використання ресурсів мережного обладнання. Збільшення коефіцієнта завантаження обладнання, який визначається як відношення швидкості передавання пакетів обладнанням до пропускної здатності обладнання, могло б істотно підвищити ефективність використання мережевого обладнання (рис. 1.11).

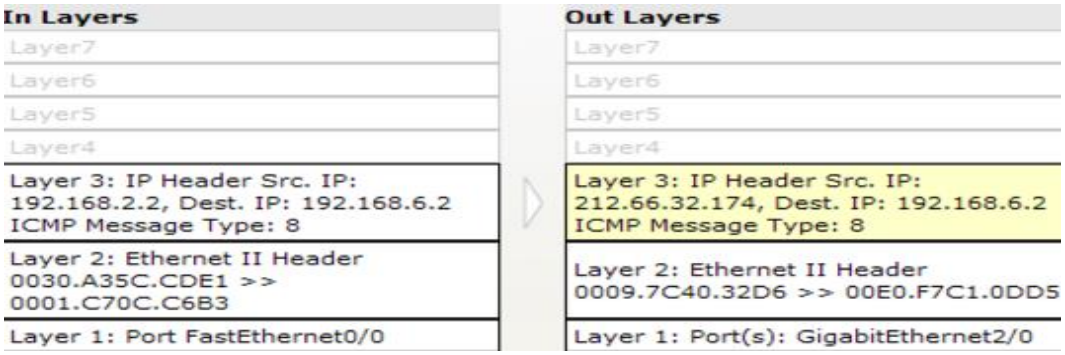

Рис. 1.11 – Приклад використання мережевого блоднання

Проблема полягає в тому, що трафік комп'ютерної мережі має скачкоподібний характер, що заважає забезпеченню високої завантаженості обладнання. Робота пакетної мережі може вважатися ефективною, якщо кожен її ресурс є істотно завантаженим, але не перенавантаженим. Застосування сучасних методів інженерії трафіку забезпечує коефіцієнт завантаження вузлового обладнання на рівні 0,5-0,55. Збільшення значення коефіцієнта завантаження на 0,05-0,1 дозволило б зекономити значні кошти. Зазвичай пропонують кілька способів збільшення навантаження на мережеве обладнання за рахунок адаптивного перерозподілу пропускної спромож- ності вузлового обладнання у реальному часі.

Приклда прогнозування трафіку показаний на рис. 1.12.

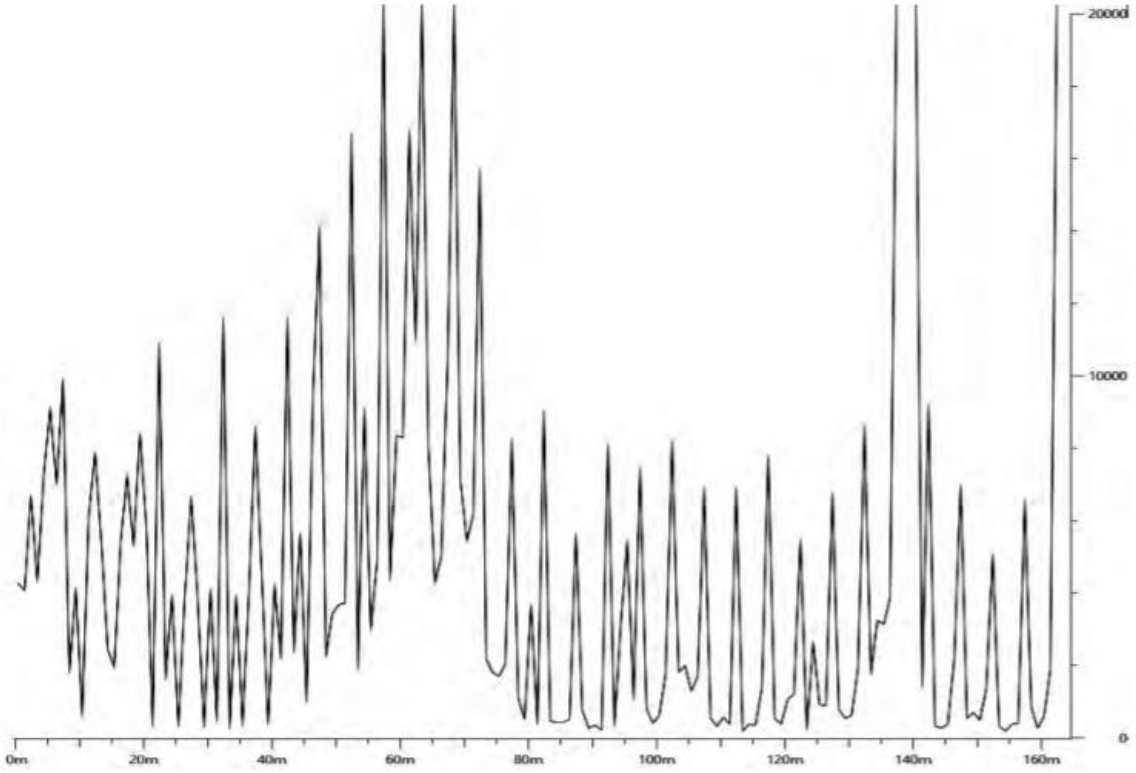

Рис. 1.12 – Приклад прогнозування трафіку мережі

Захоплення трафіку можна здійснювати за допомогою таких методів:

– за допомогою аналізу побічних електромагнітних випромінювань та відновленням у такий спосіб того трафіку, який власне «прослуховується»;

– через відгалуження (апаратне або програмне) трафіку, а також із відсиланням копії його на сніфер;

– «прослуховуванням» мережевого інтерфейсу (цей метод є ефективним за умови використання в сегменті концентраторів («hubs») замість комутаторів («switches»), в іншомувипадку метод є дуже малоефективним, оскільки на аналізатор мережевих протоколів потрапляють всього лише окремі фрейми);

– під’єднанням аналізатора мережевих протоколів у розрив каналу;

– через здійснення атаки на мережевому (IP- spoofing), чи канальному (MAC- spoofing) рівні, що призводить до перескерування даних трафіку «жертви» або загалом всього трафіку.

Результати роботи такого принципу показано на рис. 1.13.

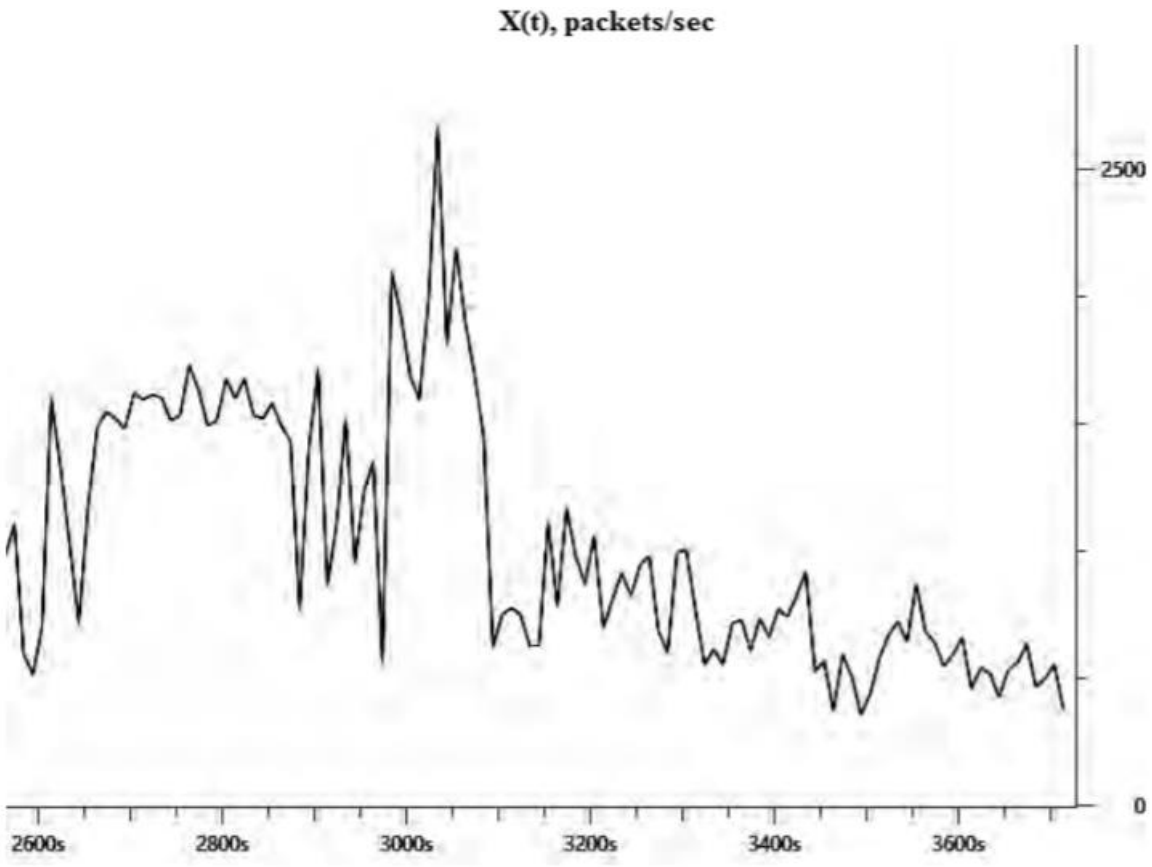

Рис. 1.13 – Частина проаналізованого трафіку

Один із способів балансування на вантаження – це введення апаратного розподілювача навантаження (Hardware Load Balancer) у мережу між мережним обладнанням та серверами. Розподілювач навантаження має всю інформацію про активність у мережі. Зокрема, обсяг трафіку до сервера або від нього, швидкість, з якою відповідає сервер на TCP/IP запити, кількість з'єднань, яке підтримує в даний момент часу кожен сервер, історія відповідей на попередні запити. У розподілювач навантаження закладено кілька стандартних алгоритмів, якими може скористатись системний адміністратор для покращання роботи мережі. Інший підхід запропоновано – це створення адаптивної системи управління перерозподілом пропускної здатності вузлового обладнання між його портами, що враховує інформацію про "поведінку" трафіку у реальному часі. У цій роботі пропонуємо на основі моніторингу мережі в реальному часі та розробленої математичної моделі прогнозування тренду трафіку мережі приймати ефективне рішення про розподіл навантаження обладнання. Розглянуто параметри, що впливають на якість обслуговування мережею: продуктивність, надійність, сумісність, керованість, захищеність, розширюваність і масштабованість та їх

зв'язок з такими основними параметрами трафіку у мережі: пропускна здатність; затримка передачі та варіація затримки передачі.

1.4.Висновки до розділу

На підставі проведеного аналізу базових понять комп’ютерних мереж, основних прийомів і методів, встановлено:

1) види комп’ютерних мереж;

2) архітектура комп’ютерних мереж;

3) призначення мереж локального та глобального типу;

4) швидкість комп’ютерних мереж;

5) мережеві ОС;

6) проблеми комп’ютерних мереж:

– проблема надійності;

– проблема захисту;

7) прогнозування трафіку комп’ютерних мереж.

Проаналізований досвід розробки аналогічних методів та засобів було використано при розробці власного продукту.

РОЗДІЛ 2. РОЗПІЗНАВАННЯ МЕРЕЖЕВИМИ ТЕХНОЛОГІЯМИ

2.1. Мережеві технології

Мережевими технологіями називають комплекс інформаційних технологій, заснованих на застосуванні штучних нейронних мереж.

Штучні мережі – це програмно або апаратно реалізовані системи, побудовані за принципом організації та функціонування їх біологічного аналога.

Як образів можуть виступати різні за своєю природою об'єкти: символи тексту, зображення, зразки звуків і т. д. При навчанні мережі пропонуються різні зразки образів із зазначенням того, до якого класу вони відносяться. Зразок, як правило, представляється як вектор значень ознак. При цьому сукупність усіх ознак повинна однозначно визначати клас, до якого належить зразок. У разі, якщо ознак недостатньо, мережа може співвіднести один і той же зразок з декількома класами, що невірно

Топологія такої мережі характеризується тим, що кількість нейронів у вихідному шарі, як правило, дорівнює кількості визначених класів. При цьому встановлюється відповідність між виходом нейронної мережі і класом, який він представляє. Коли мережі пред'являється якийсь образ, на одному з її виходів повинен з'явитися ознака того, що образ належить цьому класу. У той же час на інших виходах повинен бути ознака того, що образ даного класу не належить.

Під кластеризацією розуміється розбиття множини вхідних сигналів на класи, при тому, що ні кількість, ні ознаки класів заздалегідь не відомі. Після навчання така мережа здатна визначати, до якого класу належить вхідний сигнал. Мережа також може сигналізувати про те, що вхідний сигнал не відноситься ні до одного з виділених класів – це є ознакою нових, відсутніх в навчальній вибірці, даних. Таким чином, подібна мережа може виявляти нові, невідомі раніше класи сигналів.

Відповідність між класами, виділеними мережею, і класами, існуючими в предметної області, встановлюється людиною. Кластеризації здійснюють,

наприклад, нейронні мережі Кохонена. Приклад таких мереж зображено на рис. 2.1.

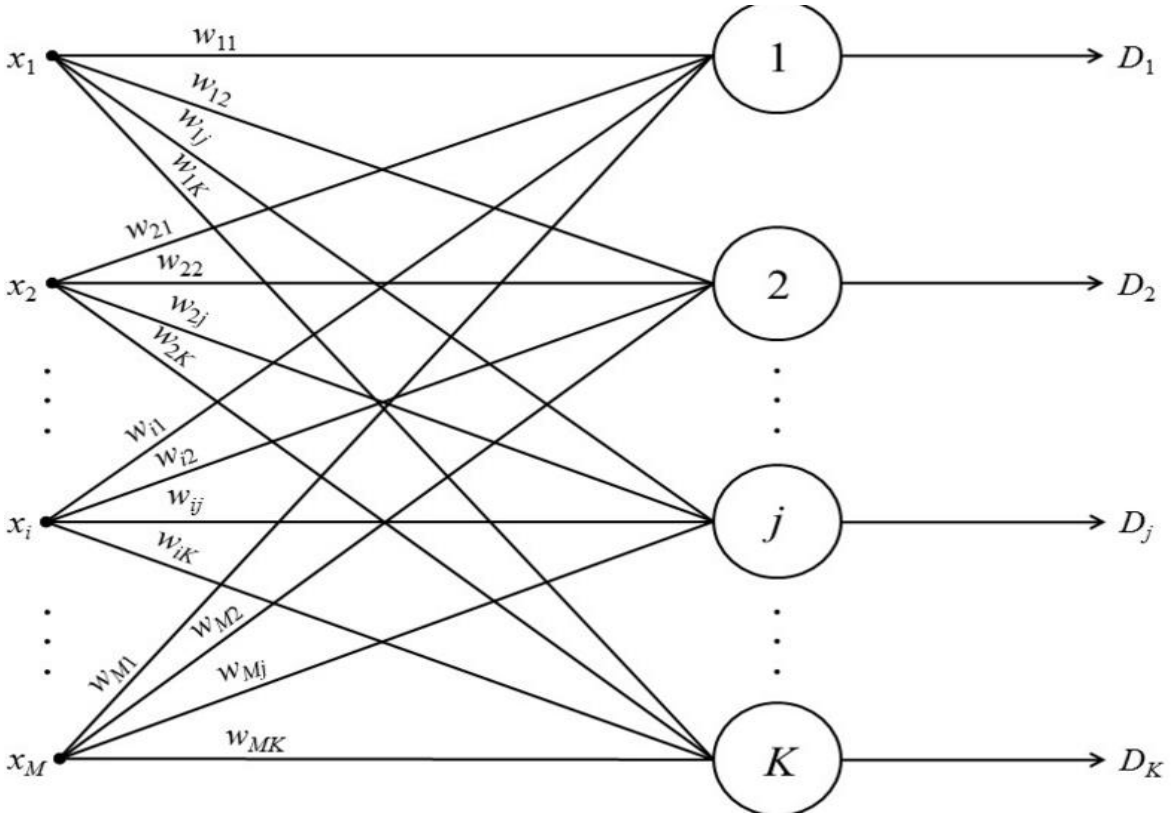

Рис. 2.1 – Приклад мережі для кластеризації

Етапи вирішення завдань за допомогою мережі.

– Збір даних для навчання.
– Підготовка і нормалізація даних.
– Вибір топології мережі.
– Експериментальний підбір пропускної здатності мережі.
– Експериментальний підбір параметрів навчання.
– Власне навчання.
– Перевірка адекватності навчання.
– Коригування параметрів, остаточне навчання.
– Вербалізація мережі з метою подальшого використання.

Слід розглянути докладніше деякі з цих етапів.

1) Збір даних для навчання.

Вибір даних для навчання мережі та їх обробка є найскладнішим етапом вирішення задачі. Набір даних для навчання повинен задовольняти декільком критеріям.

Репрезентативність – дані повинні ілюструвати справжній стан речей в предметної області.

Несуперечливість – суперечливі дані в навчальній вибірці приведуть до поганої якості навчання мережі.

Вихідні дані перетворюються до вигляду, в якому їх можна подати на входи мережі. Кожен запис у файлі даних називається навчальною парою або навчальним вектором. Навчальний вектор містить по одному значенню на кожен вхід мережі і, в залежності від типу навчання (з учителем або без), по одному значенню для кожного виходу мережі. Навчання мережі на «сирому» наборі, як правило, не дає якісних результатів. Існує ряд способів поліпшити «сприйняття» мережі (рис. 2.2).

Рис. 2.2 – Приклади даних для нейромережі

Нормировка виконується, коли на різні входи подаються дані різної розмірності. Наприклад, на перший вхід мережі подаються величини зі значеннями від нуля до одиниці, а на другий – від ста до тисячі. При відсутності нормування значення на другому вході будуть завжди надавати істотно більший

вплив на вихід мережі, ніж значення на першому вході. При нормуванні розмірності всіх вхідних і вихідних даних зводяться воєдино.

Квантування виконується над безперервними величинами, для яких виділяється кінцевий набір дискретних значень. Наприклад, квантування використовують для завдання частот звукових сигналів при розпізнаванні мови.

Фільтрація виконується для «зашумлених» даних.

Крім того, велику роль відіграє саме уявлення як вхідних, так і вихідних даних. Припустимо, мережа навчається розпізнаванню букв на зображеннях і має один числовий вихід - номер букви в алфавіті. У цьому випадку мережа отримає неправильне уявлення про те, що букви з номерами 1 і 2 більш схожі, ніж букви з номерами 1 і 3, що, загалом, не так. Для того, щоб уникнути такої ситуації, використовують топологію мережі з великим числом виходів, коли кожен вихід має свій сенс. Чим більше виходів в мережі, тим більшу відстань між класами і тим складніше їх сплутати.

2) Вибір топології мережі.

Вибирати тип мережі слід, виходячи з постановки задачі і наявних даних для навчання. Для навчання з учителем потрібна наявність для кожного елемента вибірки «експертної» оцінки. Іноді отримання такої оцінки для великого масиву даних просто неможливо. У цих випадках природним вибором є мережа, яка навчається без учителя (наприклад, самоорганізована карта Кохонена або нейронна мережа Хопфілда). При вирішенні інших завдань (таких, як прогнозування часових рядів) експертна оцінка вже міститься у вихідних даних і може бути виділена при їх обробці. В цьому випадку можна використовувати багатошаровий перцептрон або мережу Ворда.

3) Експериментальний підбір характеристик мережі.

Після вибору загальної структури потрібно експериментально підібрати параметри мережі. Для мереж, подібних перцептроном, це буде число шарів, число блоків в прихованих шарах (для мереж Ворда), наявність або відсутність обхідних з'єднань, передавальні функції нейронів рис. 2.3. При виборі кількості шарів і нейронів у них слід виходити з того, що здатності мережі до узагальнення

тим вище, чим більше сумарне число зв'язків між нейронами. З іншого боку, число зв'язків обмежене зверху кількістю записів в навчальних даних.

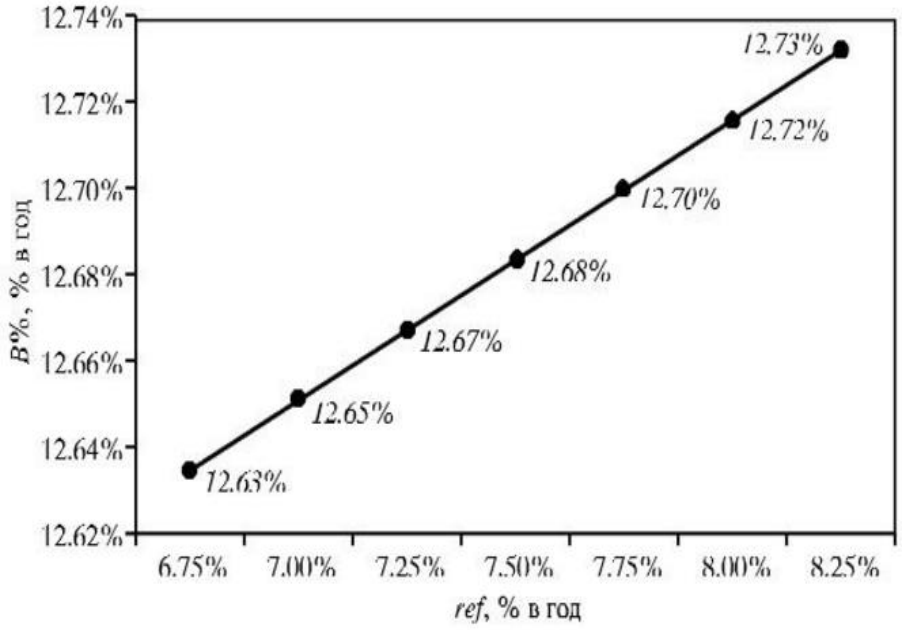

Рис. 2.3 – Експерементальний підбір характеристик мережі

4) Експериментальний підбір параметрів навчання.

Після вибору конкретної топології необхідно вибрати параметри навчання нейронної мережі. Цей етап особливо важливий для мереж, що навчаються з учителем. Від правильного вибору параметрів залежить не тільки те, наскільки швидко відповіді мережі будуть сходитися до правильних відповідей. Наприклад, вибір низькій швидкості навчання збільшить час сходження, проте іноді дозволяє уникнути паралічу мережі. Збільшення моменту навчання може привести як до збільшення, так і до зменшення часу збіжності, в залежності від форми поверхні помилки. Виходячи з такого суперечливого впливу параметрів, можна зробити висновок, що їх значення потрібно вибирати експериментально, керуючись при цьому критерієм завершення навчання (наприклад, мінімізація помилки або обмеження за часом навчання), представлено рис. 2.4.

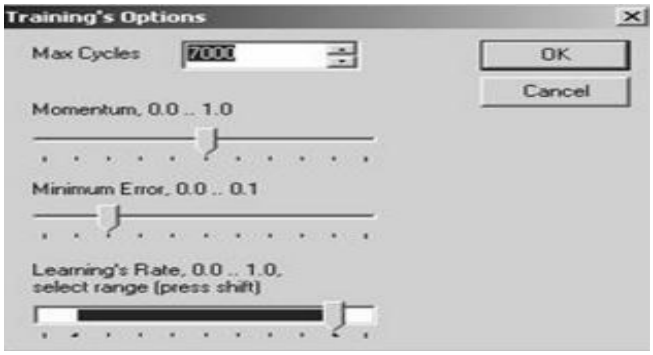

Рис. 2.4 – Підбір параметрів мережі

5) Адаптація трафіку.

В процесі адаптації в певному порядку переглядає навчальну вибірку. Порядок перегляду може бути послідовним, випадковим і т. д, представлено на рис. 2.5.

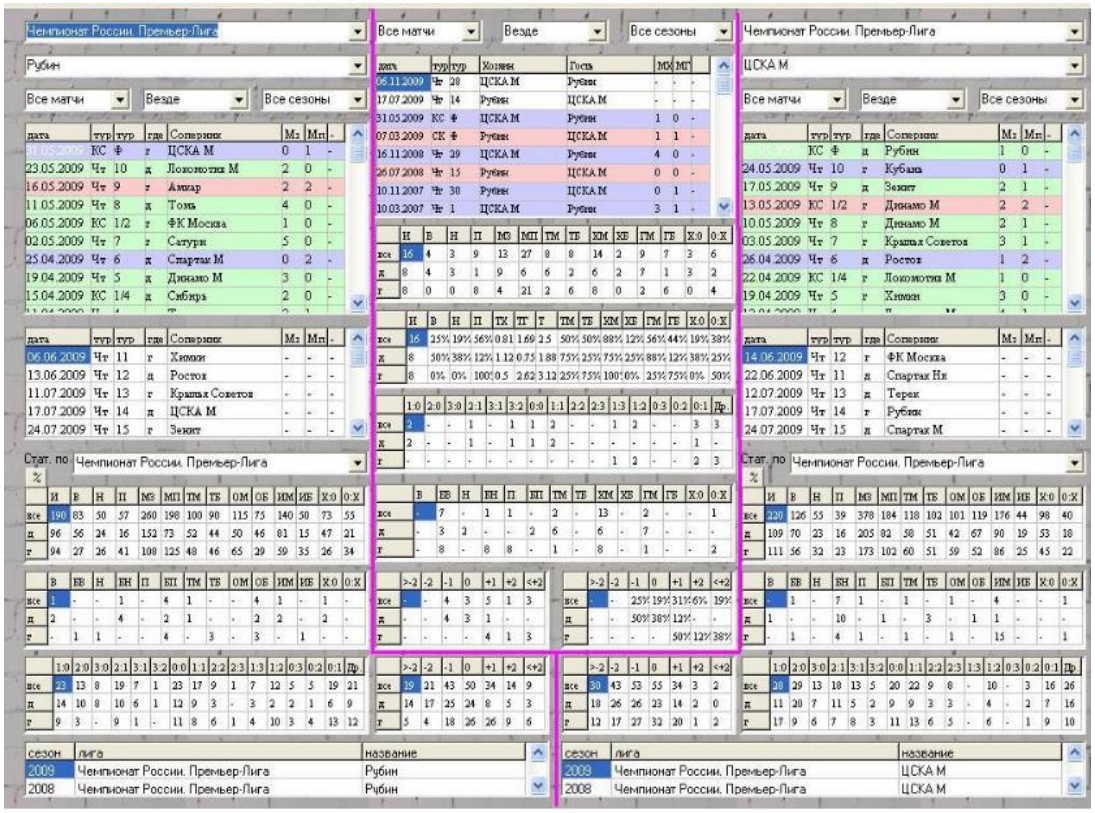

Рис. 2.5 – Приклад адаптації мережі

Деякі мережі, які навчаються без учителя (наприклад, мережі Хопфілда), переглядають вибірку тільки один раз. Інші (наприклад, мережі Кохонена), а також мережі, які навчаються з учителем, переглядають вибірку безліч разів, при цьому один повний прохід по вибірці називається епохою навчання. При навчанні з учителем набір вихідних даних ділять на дві частини - власне навчальну вибірку і тестові дані; принцип поділу може бути довільним.

Навчальні дані подаються мережі для навчання, а перевірочні використовуються для розрахунку помилки мережі.

6) Перевірка правельності навчання.

Навіть в разі успішного, на перший погляд, навчання мережа не завжди навчається саме тому, чого від неї хотів творець. Відомий випадок, коли мережа навчалася розпізнаванню зображень танків по фотографіях, проте пізніше з'ясувалося, що всі танки були сфотографовані на одному і тому ж фоні. У результаті мережа «навчилася» розпізнавати цей тип ландшафту, замість того, щоб «навчитися» розпізнавати танки. Таким чином, мережа «розуміє» не те, що від неї вимагалося, а то, що найпростіше узагальнити. Перевірка адекватності представлена на рис. 2.6.

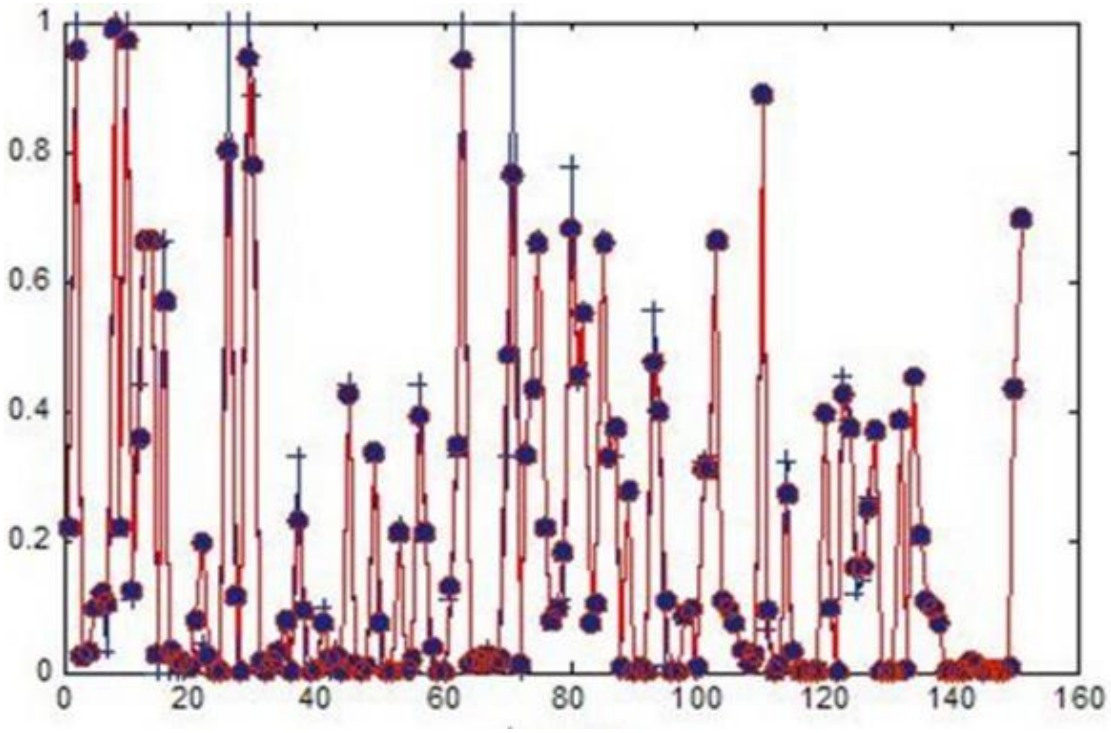

Рис. 2.6 – Перевірка достовірності мережі

Тестування якості навчання нейромережі необхідно проводити на прикладах, які не брали участі в її навчанні. При цьому число тестових прикладів має бути тим більше, чим вище якість навчання. Якщо помилки нейронної мережі мають можливість близьку до однієї мільярдної, то і для підтвердження цієї ймовірності потрібен мільярд тестових прикладів. Виходить, що тестування добре навчених нейронних мереж стає дуже важким завданням.

2.2. Аналіз застосованості до прогнозування подій

Зараз, найперспективнішим методом прогнозування подій є використання нейронних мереж. Можна назвати багато переваг нейронних мереж над іншими алгоритмами, нижче наведено два основні.

При використанні нейронних мереж легко досліджувати залежність прогнозованої величини від незалежних змінних. Наприклад, є припущення, що продажі на наступному тижні якимось чином залежать від наступних параметрів:

- продажів в останній тиждень;
- продажів у передостанній тиждень;
- часу прокручування рекламних роликів (TRP);
- кількості робочих днів;
- температури.

Крім того, продажі носять сезонний характер, мають тренд і якось залежать від активності конкурентів.

Хотілося б побудувати систему, яка б усе це природнім чином враховувала і будувала б короткострокові прогнози.

У такій постановці завдання застосування більшої частини класичних методів прогнозування буде просто неможливою.

Використовуючи ж навіть найпростішу нейромережеву архітектуру (перцептрон з одним схованим шаром) і базу даних (із продажами й усіма параметрами) легко одержати працюючу систему прогнозування. Причому враховувати, чи не враховувати зовнішні параметри системою буде визначатися включенням, або виключенням відповідного входу в нейронну мережу.

Експерт може скористатися яким-небудь алгоритмом визначення важливості і відразу визначити значимість вхідних змінних, щоб потім виключити з розгляду параметри, що мають незначний вплив.

Ще одна серйозна перевага нейронних мереж полягає в тому, що експерт не є заручником вибору математичної моделі поведінки часового ряду. Побудова нейромережевої моделі відбувається адаптивно під час навчання, без участі

експерта. При цьому нейронній мережі пред'являються приклади з бази даних і вона сама підлаштовується під ці дані.

Недоліком нейронних мереж є їхня недетермінованість. Мається на увазі те, що після навчання є "чорний ящик", який якимось чином працює, але логіка прийняття розв'язків нейромережею зовсім схована від експерта. У принципі, існують алгоритми "витягу знань із нейронної мережі", які формалізують навчену нейронну мережу до списку логічних правил, тим самим створюючи на основі мережі експертну систему. На жаль, ці алгоритми не вбудовуються в нейромережеві пакети, до того ж набори правил, які генеруються такими алгоритмами досить об'ємні.

Проте, для людей, що вміють працювати з нейронними мережами й знаючими нюанси налаштування, навчання й застосування, у практичних завданнях непрозорість нейронних мереж не є настільки серйозним недоліком. Використання багатошарових персептронов

Найпростіший варіант застосування штучних нейронних мереж у завданнях бізнес-прогнозування – використання звичайного перцептрона з одним, двома, або трьома прихованими шарами. При цьому на входи нейронної мережі звичайно подається набір параметрів, на основі якого (на думку експерта) можна успішно прогнозувати. Виходом звичайно є прогноз мережі на майбутній момент часу Приклад представлено на рис. 2.7.

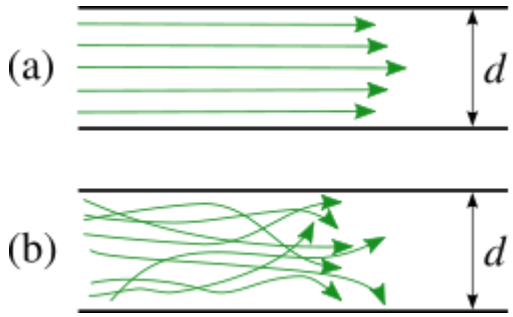

Рис. 2.7 – Приклад прогнозування трафіку

2.3. Модель застосування для прогнозування трафіку

У сучасному світі все з більшою гостротою проявляється інтерес до якісної прогнозування трафіку комп’юторної мережі. Це пов'язано з швидким розвитком

високих технологій і, відповідно, з появою нових інструментів аналізу даних. Однак той технічний аналіз, яким звикли користуватися більшість учасників ринку, не є ефективним. Прогнози на основі експоненційних ковзають середніх, осцилляторах і інших індикаторах не дають відчутний результат, тому що економіка часто буває ірраціональна, тому що рухають ірраціональними мотиваціями людей.

В останні роки, у аналітиків провайдера стали викликати великий інтерес так звані штучні нейронні мережі - це математичні моделі, а також їх програмні або апаратні реалізації, побудовані за принципом організації та функціонування біологічних нейронних мереж – мереж нервових клітин живого організму. Це поняття виникло при вивченні процесів, що протікають в мозку при мисленні, і при спробі змоделювати ці процеси. Згодом ці моделі стали використовувати в практичних цілях, як правило, в задачах прогнозування. Нейронні мережі не програмуються в звичному сенсі цього слова, вони навчаються. Можливість навчання – одне з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Здібності нейронної мережі до прогнозування безпосередньо випливають з її здатності до узагальнення і виділення прихованих залежностей між вхідними та вихідними даними. Після навчання мережа здатна передбачити майбутнє значення якоїсь послідовності на основі декількох попередніх значень і / або якихось існуючих зараз чинників. Слід зазначити, що прогнозування можливо тільки тоді, коли попередні зміни дійсно в якійсь мірі визначають майбутні. Наприклад, прогнозування котирувань акцій на основі котирувань за минулий тиждень може виявитися успішним, тоді як прогнозування результатів завтрашньої лотереї на основі даних за останні 50 років майже напевно не дасть ніяких результатів.

Розглянемо на практиці застосування методу прогнозування за допомогою нейронних мереж. Для прикладу візьмемо дані інтернет провайдера Воля в період з 01.10.2008 по 03.04.2009. Завдання полягає в тому, що на основі

представленої статистичної інформації необхідно зробити прогноз на 10 днів. Як видно з рис. 2.8, з 01.10.08 по 28.10.08 індекс трафіку мережі "просів" приблизно на 534 пункту. Після чого пішло зростання до максимальної позначки в 871 пунктів. Далі, деякий час, користувачі перебували у бічному тренді, потім намітилася висхідна тенденція. В даному прикладі будемо будувати прогноз для однієї змінної (інші аналогічно), але для того, щоб вибрати ту з чотирьох змінних, яка найбільш сильно допоможе спрогнозувати інші, побудуємо кореляційну матрицю.

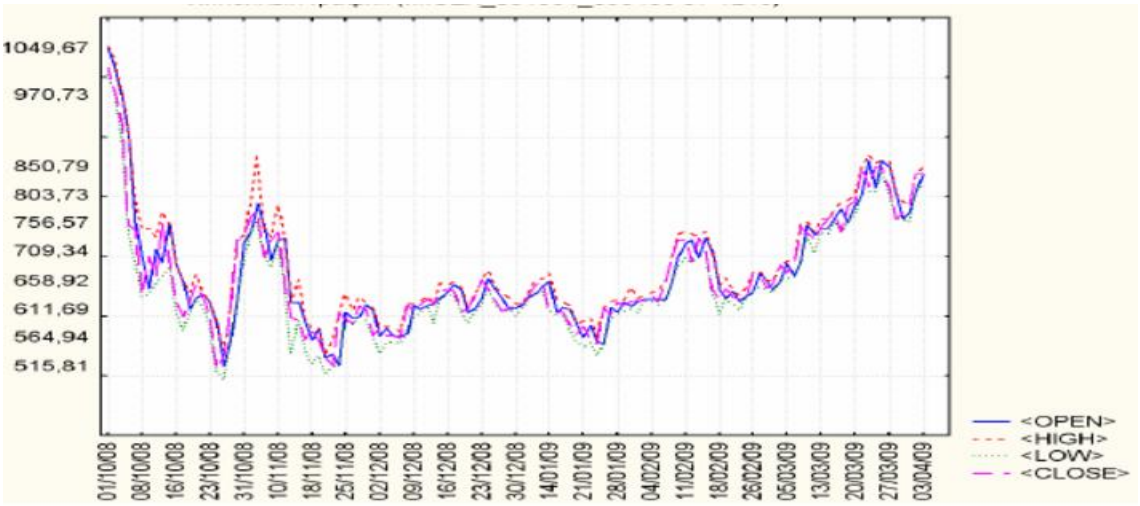

Рис. 2.8 – Індекси трафіку мережі провайдера Воля

Отже, побудувавши матрицю парних кореляцій, робимо висновок про те, що змінна LOW найбільш сильно корелює з іншими. Займемося прогнозом даної змінної, зображена на рис. 2.9.

	OPEN	HIGH	LOW	CLOSE
OPEN	1,00	0,98	0,97	0,93
HIGH	0,98	1,00	0,96	0,96
LOW	0,97	0,96	1,00	0,98
CLOSE	0,93	0,96	0,98	1,00

Рис. 2.9 – Приклад кореляцій параметрів мережі

Нелінійні за своєю суттю нейронні мережі, дозволяють з будь-яким ступенем точності апроксимувати довільну безперервну функцію, не дивлячись на відсутність або наявність будь-якої періодичності або циклічності. Оскільки тимчасової ряд представляє собою безперервну функцію (насправді нам відомо значення цієї функції лише в кінцевому числі точок, але її можна легко

безперервно продовжити на весь аналізований відрізок), то застосування нейронних мереж цілком виправдано і коректно.

Побудуємо тисячу нейронних мереж різної конфігурації в пакеті STATISTICA, навчимо їх, а потім виберемо десять найкращих.

В результаті навчання була знайдена нейронна мережа, відповідна моделі 7 (рис. 2.9) з хорошою продуктивністю (регресійні відношення: 0,253628, помилка: 0,003302). Неважко помітити, що продуктивність мереж з архітектурою Радиально Базисної Функції (РБФ) в середньому гірше продуктивності мереж з архітектурою багатошарового персептрона. Багато в чому це пояснюється тим, що мережі з архітектурою РБФ погано екстраполюють дані (це пов'язано з насиченням елементів прихованої структури). Для оцінки правдоподібності моделі 7 побудуємо гістограму частот (рис.3). Дана гістограма є найбільш симетричною в порівнянні з іншими моделями. Це підтверджує стандартні припущення про нормальність залишків. Отже, модель 7 найбільше підходить для даного часового ряду на рис 2.10.

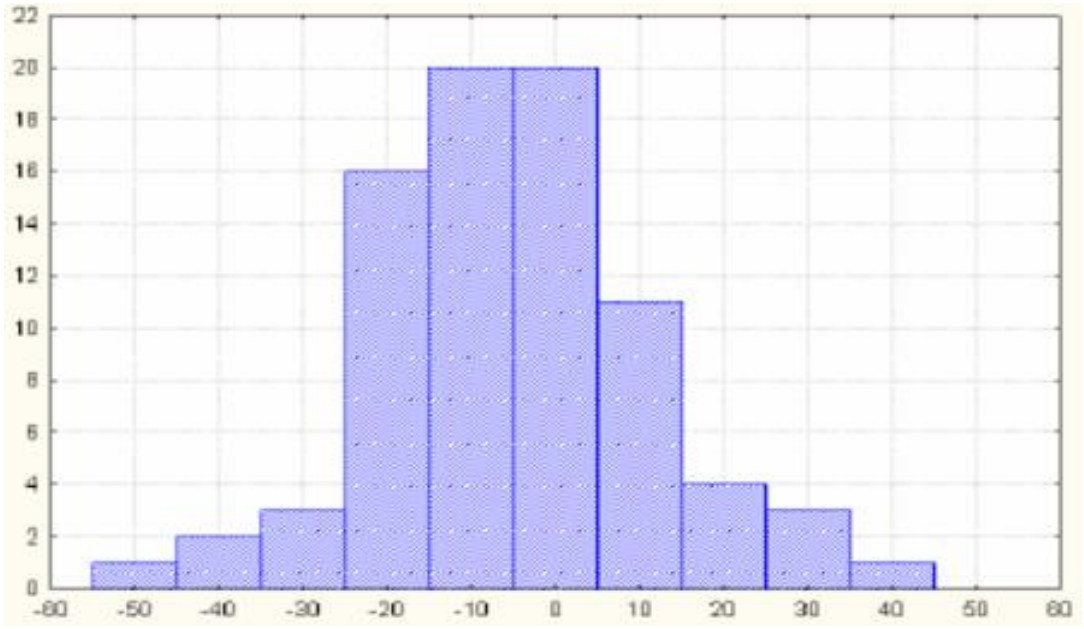

Рис. 2.10 – Центральна гранична теорема для перевірки показників

Здійснимо проекцію для прогнозування часового ряду. В результаті маємо прогноз.

При наявності явної лінійності, простоти структури в задачі, здатність нейронних мереж до узагальнення виявляється слабшою по відношенню до

класичних методів. Пояснюється це як раз нелинейностью мереж за своєю суттю. Результта прогнозування показано на рис. 2.11.

Рис. 2.11 – Результати прогнозування трафіку комп'юторної мережі

У загальному випадку для досягнення найкращого результату необхідно використовувати нейронні мережі вкупі з грамотною стратегією трафіком в мережі та обслуговуванням клієнтів.

2.4. Висновки до розділу

В розділі подано основні структури мереж: функціональні та структурні схеми їх реалізації. Розкрито основні методи розробки підсистеми на основі існуючих технологій.

Проведено архітектурне проектування програмної системи яка складається з модулю прийому, передачі та обробки даних.

В розділі було проаналізовано основні переваги та недоліки нейронних мереж різного типу:

1) швидкість;

2) точність;

3) потужність;

Було розроблено алгоритм аналізу з боку прикладної частини. Описано аналіз прогнозування футбольного мачу.

Проаналізовано етапи розробки проекту та технології, що будуть використовуватися для реалізації.

Описано розробку алгоритму для передачі інформації, особливістю якої, є швидкість та зручність.

Обрано набір технологій для програмної частини сервісу.

На основі наведеної інформації створено нейронну мережу для прогнозування комп’ютерного трафіку.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Процес індивідуалізації мережі

Індивідуалізація - визначення процесу навчання мережі передбачає наступну послідовність подій:

У мережу надходять стимули із зовнішнього середовища.

В результаті першого пункту змінюються вільні параметри мережі.

Після зміни внутрішньої структури вся мережа відповідає на порушення вже іншим чином.

Вищевказаний список чітких правил вирішення проблеми навчання ї мережі називається алгоритмом навчання. Нескладно здогадатися, що не існує універсального алгоритму навчання, відповідного для всіх архітектур.

Існує лише набір засобів, представлений безліччю алгоритмів навчання, кожен з яких має свої переваги. Алгоритми навчання відрізняються один від одного способом налаштування синаптичних ваг нейронів. Ще однією відмінною характеристикою є спосіб зв'язку навченою нейронної мережі з зовнішнім світом. У цьому контексті говорять про парадигму навчання, пов'язаної з моделлю навколишнього середовища, в якому функціонує дана мережа.

Індивідуалізація мережі для передбачення стану трафіку в мережі:

1) Ініціалізація важелів (ваги всіх зв'язків инициализируются випадковими невеликими значеннями). Отже, перш за все ми присвоїли довільні значення можливостям зміни стану трафіка мережі.

2) До тих пір поки умова припинення роботи алгоритму не вірна, виконуються кроки 2 - 9. Тобто, поки нейромережа не буде надавати правельні результати. Вона буде навчатися.

3) Для кожної пари (дані, цільове значення) виконуються кроки 3 - 8. Поширення даних від входів до виходів. Тобто нейрона мережа починає працювати саме з цього пункут. Набір даних який включає інформацію, що до

які саме дані ми маємо та який стан він повинен отримувати зазначені в базі даних.

4) Кожен вхідний нейрон відправляє отриманий сигнал всім нейронам в наступному шарі. Дані для стану мережі циркулюють по всім нейронам та залишають там певні параметри.

5) Кожен прихований нейрон підсумовує зважені вхідні сигнали і застосовує активаційну функцію. Після чого посилає результат всіх елементів наступного шару.

6) Кожен вихідний нейрон підсумовує зважені вхідні сигнали і застосовує активаційну функцію, обчислюючи вихідний сигнал. Зворотне поширення помилки відбувається при не стикані вихідних даних з вхідними.

7) Кожен вихідний нейрон отримує цільове значення – то вихідне значення, яке є правильним для даного вхідного сигналу, і обчислює помилку, так само обчислює величину, на яку зміниться вага зв'язку. Крім цього, обчислює величину коригування зміщення: і її посилає нейронам в попередньому шарі.

8) Кожен прихований нейрон підсумовує помилки (від нейронів в наступному шарі) і обчислює величину помилки, множачи отримане значення на похідну активаційнної функції, так само обчислює величину, на яку зміниться вага зв'язку. Крім цього, обчислює величину коригування зміщення.

9) Кожен вихідний нейрон змінює ваги своїх зв'язків з елементом зміщення і прихованими нейронами. Кожен прихований нейрон змінює ваги своїх зв'язків з елементом зміщення і вихідними нейронами.

10) Перевірка умови припинення роботи алгоритму. Умовою припинення роботи алгоритму може бути як досягнення сумарної квадратичної помилкою результату на виході мережі встановленого заздалегідь мінімуму в ході процесу навчання, так і виконання певної кількості ітерацій алгоритму. В основі алгоритму лежить метод під назвою градієнтний спуск. Залежно від знака, градієнт функції (в даному випадку значення функції - це помилка, а параметри - це ваги зв'язків в мережі) дає напрямок, в якому значення функції зростають (або зменшуються) найбільш стрімко.

В даному дипломному проекті навчання мережі викуонується за допомогою бібліотеки PyBrain. Як саме відбувається дана операція розкрито в розділі 3.2.

В цьому розділі я вирішив також розглянути дані які ми будемо аналізувати та за допомогою них робити припущення, що до стану трафіку комп'юторної мережі. На рис. 3.1 зображено excel файл с даними які ми будемо аналізувати.

28-Oct-16	130.50	132.97	129.93	131.29
27-Oct-16	131.74	131.80	129.27	129.69
26-Oct-16	131.64	132.26	130.94	131.04
25-Oct-16	133.50	133.50	132.22	132.29
24-Oct-16	132.72	133.40	132.15	133.28
21-Oct-16	129.78	132.13	129.70	132.07
20-Oct-16	130.07	130.66	129.50	130.00

Рис. 3.1 – Файл с даними для аналізу

В першій колонці визначена дата провереного сбору даних. Саме вона підскаже нам з якою перебіжністю змінюються параметри трафіку мережі. Інші колоники показують середній рівень передачі даних в мегабайтах. Дані можут здатися дивними, оскільки якісного трафіку не було знайдено, жоден провайдер не викидує в мережу дані про своій трафік.

В кожному дні є певні зміни тараметрів заеложно від часу використання трафіком. Як правило в день, люди використовують менше інтернет можливостей, а під вечір більш активізуються, якщо починаються скачувати фільми, ігри та інші великогабаритноі файли.

За допомогою певних програмних додатків було також обрано прийнятну мережу, яка показана на рис. 3.2.

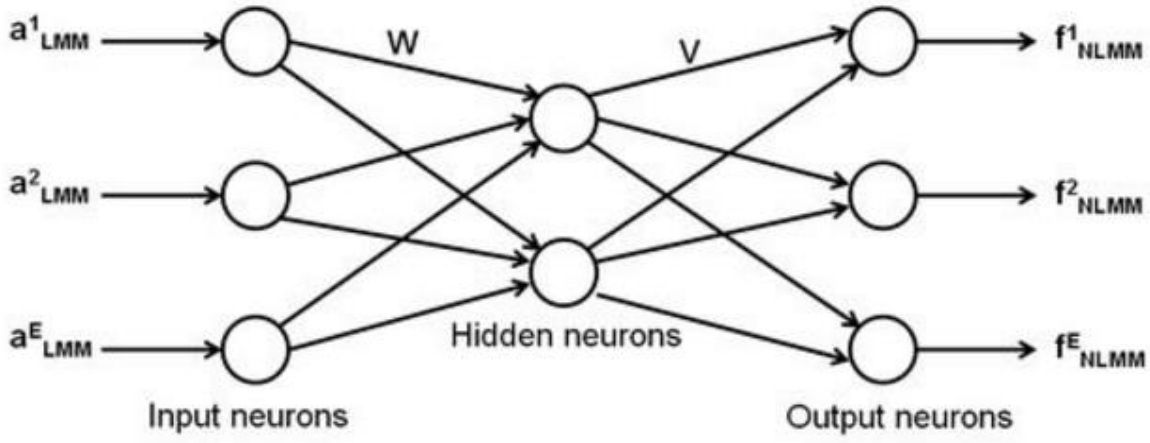

Рис. 3.2– Обрана мережа для прогнозувння трафіку

3.2. Реалізація програмного коду

Перш за все треба розглянути схемау роботи мереж за допомогою представленого алгоритму на рис. 3.3.

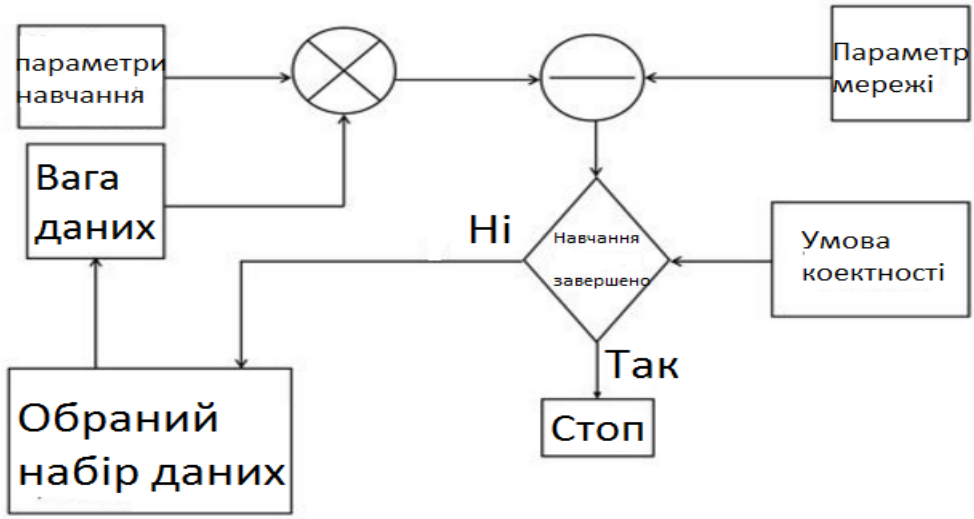

Рис. 3.3 – Алгоритм роботи розробленої мережі

Як можна зрозуміти із алгоритму, обрано стандартний підхід при навчанні мережі і його прогнозуванню.

Діаграма use-case (діаграма прецедентів) в UML - діаграма, що відображає відносини між акторами і прецедентами і є складовою частиною моделі прецедентів, що дозволяє описати систему на концептуальному рівні.

Основне призначення діаграми – опис функціональності і поведінки, що дозволяє замовнику, кінцевому користувачеві і розробнику спільно

обговорювати проектовану або існуючу систему, для коректності, потужності та, головне, успішності програми.

При моделюванні системи за допомогою діаграми прецедентів системний аналітик прагне:

– чітко відокремити систему від її оточення;

– визначити дійових осіб (акторів), їх взаємодію з системою і очікувану функціональність системи;

– визначити в глосарії предметної області поняття, які стосуються детального опису функціональності системи (тобто, прецедентів).

Робота над діаграмою може початися з текстового опису, отриманого при роботі із замовником. При цьому нефункціональні вимоги (наприклад, конкретний мову або система програмування) при складанні моделі прецедентів опускаються (для них складається інший документ).

Для відображення моделі прецедентів на діаграмі використовуються:

1) **Рамки системи** (англ. system boundary) - прямокутник з назвою у верхній частині і еліпсами (прецедентами) всередині. Часто може бути опущений без втрати корисної інформації.

2) **Актор** (англ. actor) - стилізований чоловічок, що позначає набір ролей користувача (розуміється в широкому сенсі: людина, зовнішня сутність, клас, інша система), що взаємодіє з деякою сутністю (системою, підсистемою, класом). Актори не можуть бути пов'язані один з одним (за винятком відносин узагальнення / успадкування).

3) **Прецедент** - еліпс з написом, що позначає виконувані системою дії (можуть включати можливі варіанти), що призводять до спостережуваних акторами результатами. Напис може бути ім'ям або описом (з точки зору акторів) того, «що» робить система (а не «як»). Ім'я прецеденту пов'язано з неперервним (атомарним) сценарієм - конкретної послідовністю дій, що ілюструє поведінку. В ході сценарію актори обмінюються з системою повідомленнями. Сценарій може бути приведений на діаграмі прецедентів у вигляді UML-коментаря. З одним прецедентом може бути пов'язано кілька різних сценаріїв.

В даному дипломному проекті для засобу прогнозування на основі нейронної мережі було вирішено використовувати двух акторів:

– User (користувач програмного забезпечення).

– System (розроблений додок).

На рис. 3.5 представленні актори та їх прецеденти. Для більш детального розуміння розробленого програмного засобу розглянемо кожен прецедент окремо. Почнемо з можливостей користувача (User).

1) Внесення даних (користувач може завантажити набір даних та розпочати системну обробку на основі нейромережі).

2) Параметри мержі та навчання (користувач може додати або замінити параметри для нейронної мержі).

Так як, другим актором запропонованої системи є сам розроблений додаток (System) проаналізуємо його можливості.

1) Моделювання (формується набір даних із запропонованих для обробки та набір даних для нейронної мережі).

2) Отримання параметрів та перевірка достовірності (відбувається обробка сформованих даних та їх конвертація в вид набору даних трафіку за допомогою прецидента Convert, який належить до виконанню Pre-Process. Нейромережа починає робити обробку даних на основі запропонованої моделі в преценденті Binary format. Normalization дозволяє отримати дані після обробки у зрозумілому вигляді).

3) Вивдення на екран (використовується для показання користувачу прогнозованих даних, для розуміння запропонованих дат та їх щогодиного використання трафіку).

4) Прогнозування результатів (прогнозування результатів є та сама обробка, що вибиває інформаціє на дисплей, то відтасовує прогнозовані дані від вже існуючих).

Передачі класів описані за допомогою стрлочок та параметрів користувача і їх ролей у певних методах та функціях. Отже можна зрозумаіти як саме працює розроблений додаток на рис. 3.4.

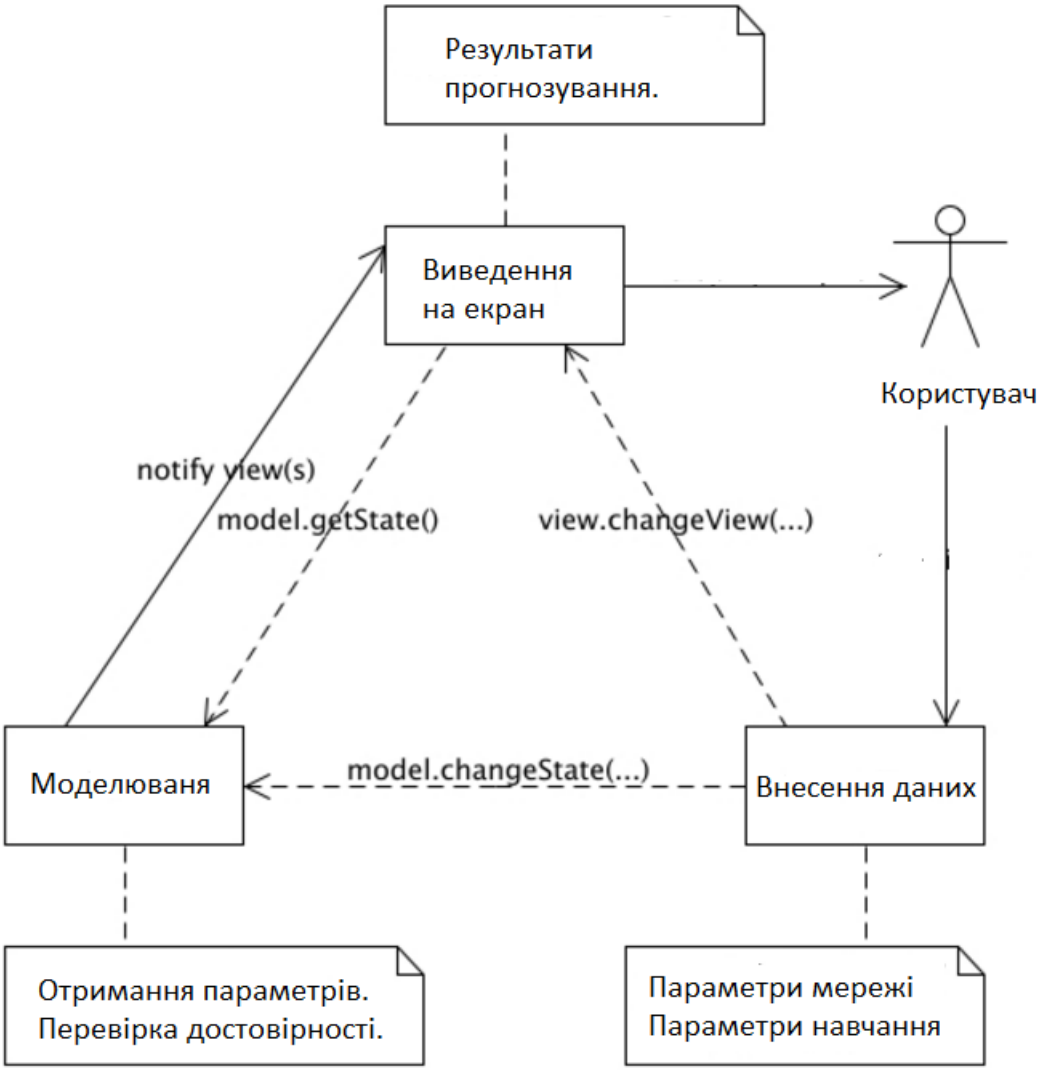

Рис. 3.4 – Use-case діаграма розробленої системи

Після створення діаграми слід проаналізувати чи були дотримані всі правила, що до реалізації:

– кожен прецедент відноситься як мінімум до одної дійової особи;
– кожен прецедент має ініціатора;
– кожен прецедент призводить до відповідного результату.

Діаграмою класів в термінології UML називається діаграма, на якій показаний набір класів (і деяких інших пунктів, які не мають явного відношення до проектування БД), а також зв'язків між цими классам. Коли говорять про дані діаграми, мають на увазі статичну структурну модель проектованої системи. Тому діаграму класів прийнято вважати графічним представленням таких структурних взаємозв'язків логічної моделі системи, що не залежать від часу.

Крім того, діаграма класів може включати коментарі та обмеження. Обмеження можуть неформально задаватися на природній мові або ж можуть формулюватися мовою об'єктних обмежень OCL (Object Constraints Language).

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. Діаграма класів може відбивати, зокрема, різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти і підсистеми, а також описує їх внутрішню структуру і типи відносин. На даній діаграмі не вказується інформація про тимчасові аспекти функціонування системи. З цієї точки зору діаграма класів є подальшим розвитком концептуальної моделі проектованої системи після use-case діаграми.

Діаграма класів є певний граф, вершинами якого є елементи типу "класифікатор", які пов'язані різними типами структурних відносин. Слід зауважити, що діаграма класів може також містити інтерфейси, пакети, відносини і навіть окремі екземпляри, такі як об'єкти зв'язку. Коли говорять про дані діаграми, мають на увазі статичну структурну модель проектованої системи. Тому діаграму класів прийнято вважати графічним представленням таких структурних взаємозв'язків логічної моделі системи, що не залежать від часу. Загалом, діаграма класів складається з безлічі елементів, які в сукупності відображають декларативні знання про предметну область.

В даному дипломному проекті для прогнозування трафіку комп’ютерної мережі на основі нейромережевих технологій було розроблено діаграму класів, яку представлено на рис. 3.5.

Діаграма класів показує те, що ми використовуємо в обьєктно орієнтований підхід для аналізу запропонованих даний.

Цей підхід є найпопулярнішим для аналізу в наш час.

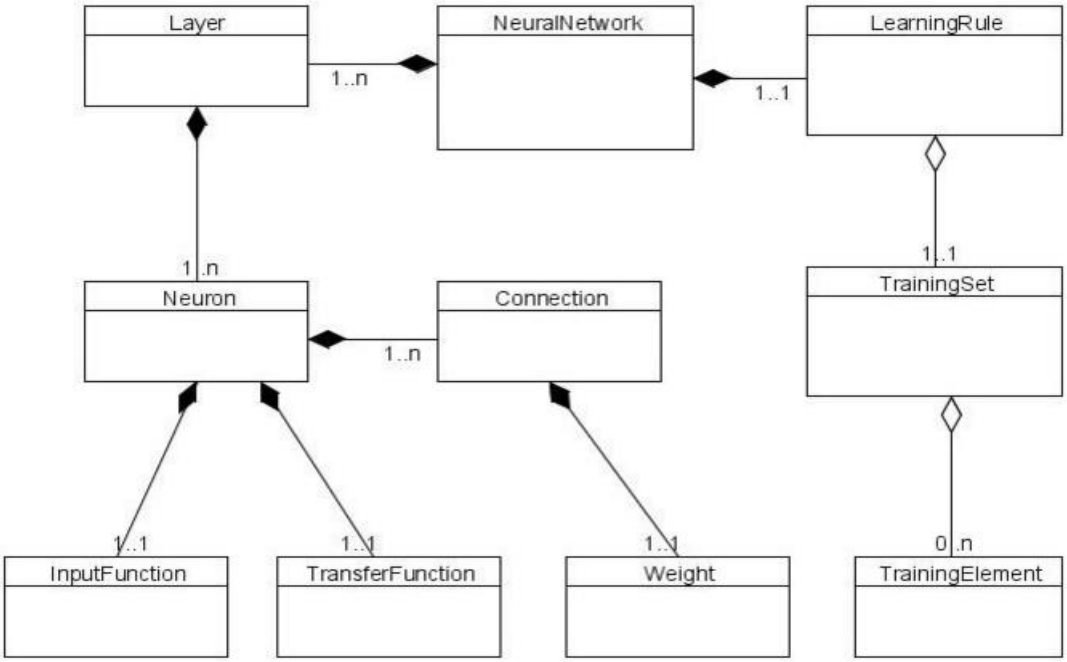

Рис. 3.5 – Діаграма класів розробленої системи

Розглянемо кожен клас та його задачі більш детально.

1) **TrainingElement** – відповідає за навчання системи, що до інтегрованих даних, кожний набір даний який було внесено в системо залишає свій слід на вагах певних факторів при використанні нейромережі, сам тут і відбувається кореляція цієї інформації.

2) **TreiningSet** – відповідає за представлення інформації на основі запропонованих даних, після обробки цих даних нейромережею.

3) **LearningRule** – модуль в якому буде знаходитися набір даних за рахунок якого навчена мережа. Він потрібен для коректної роботи програми, оскільки при використанні всього лише важелів параметрів, запропонований проект буде видавати менш коректні результати.

4) **NeuralNetwork** – клас в якому збираються дані про завантажений файл, усі параметри файлу проходять первинну обробку саме у цьому класі.

5) **Layer** – клас в якому відбувається аналіз запропонованого файлу з частками трафіку та генерується результат прогнозування.

6) **Neuron** – бібліотеки та їх сумісна робота відбувається в даній частині додатку, більш детально цей клас буде розглянуто в додатку.

7) **Connection** – клас в якому зберігається модуль для обробки запропонованого набору даних (модуль визначається за допомогою бібліотеки PyBrain).

Було розглянуто існуючі класи в розробленому додатку, більш детально проаналізовано методи та функції кожного з них буде в розділі присвяченому розробці програмного коду, там буде розглянуто функціонал який пропонує нам бібліотека PyBrain.

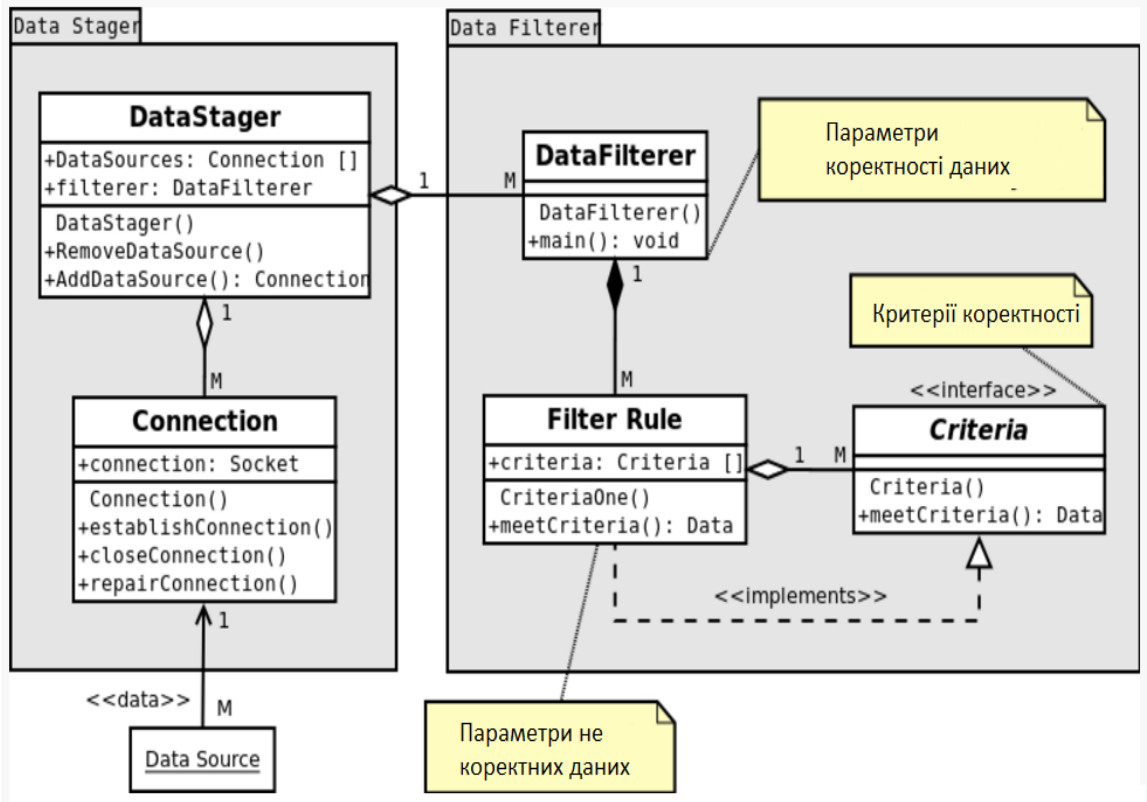

Рис. 3.6 – Клас перевірки коректності даних

Діаграма компонентів (Component diagram) – статична структурна діаграма, показує розбиття програмної системи на структурні компоненти та зв'язку (залежності) між компонентами. Як фізичні компоненти можуть виступати файли, бібліотеки, модулі, виконувані файли, пакети.

Компоненти зв'язуються через залежності, коли з'єднується необхідний інтерфейс одного компонента з наявним інтерфейсом іншого компонента. Таким чином ілюструються відносини клієнт-джерело між двома компонентами. Це дає змогу розуміти як саме програмні частини працюють між собою, та що саме вони утворюють разом.

Залежність показує, що один компонент надає сервіс, необхідний іншому компоненту. Залежність зображується стрілкою від інтерфейсу або порту клієнта до імпортованого інтерфейсу.

Коли діаграма компонентів використовується, щоб показати внутрішню структуру компонентів, що надається і необхідний інтерфейси складеного компонента можна делегувати до відповідних інтерфейсів внутрішніх компонентів.

Делегація показується зв'язок зовнішнього контракту компонента з внутрішньої реалізацією цієї поведінки внутрішніми компонентами.

В даному дипломному проекті для засобу прогнозування трафіку на основі нейронної мережі було розроблено наступну діаграму компонентів, яку представлено на рис. 3.7.

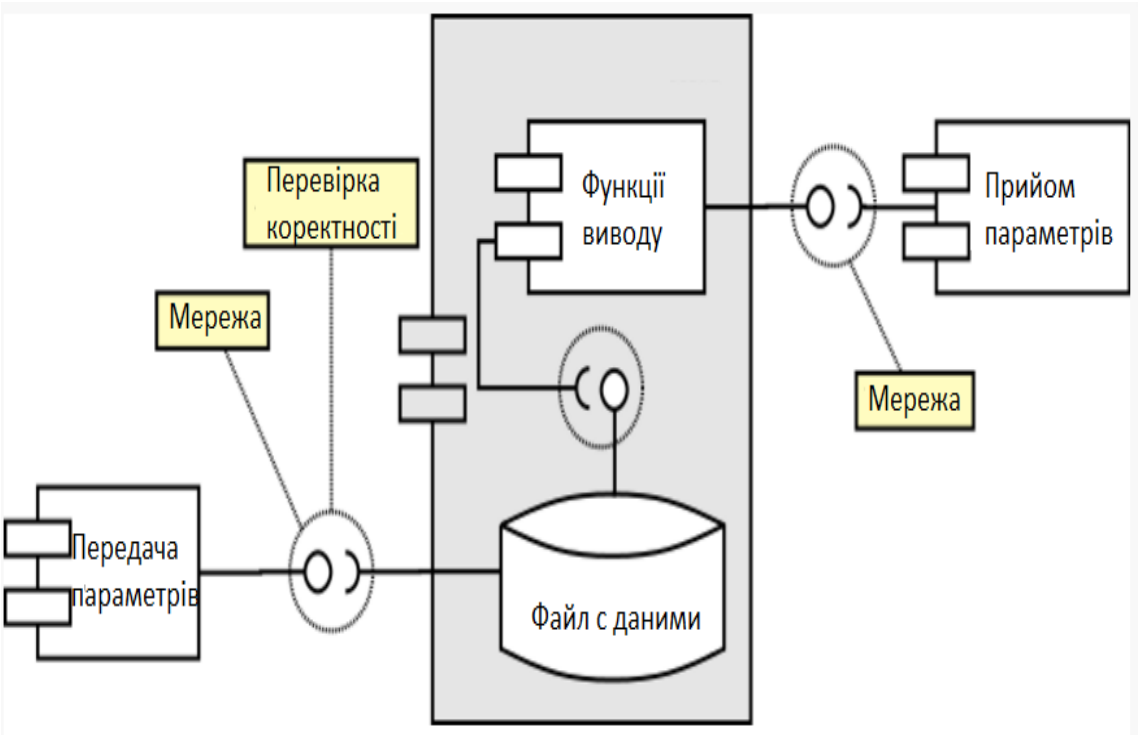

Рис. 3.7 – Діаграма компонентів розробленої системи

Розглянемо кожен компонент та його функції більш детально:

1) **Передача параметрів** – в даному дипломному проекті це модулі за рахунок яких можлива робота з програмою, тобто функціонал такий як загрузка даних в систему, отримання результату обробки. Цей компонент відповідає за той набір опцій, що відповідає за комунікацію з користувачем.

2) **Прийом параметрів** – даний компонент відповідає за налаштування прогнозуванню за роздачу важелів для певних параметрів система. За кількість використаних даних із запропонованого набору. За видалення, модифікацію та додавання даних.

3) **Файл с даними** – компонент дає змогу порівняти набір даних який ми використовуємо для навчання та для опрацювання. Знаходить закономірності, відповідності та послідовності.

4) **Файл виводу** – даний компонент містить нейромережеві методи та функції реалізовані за допомогою запропонованих бібліотек. Сам це і є основною частиною розробленого додатку в якому виконується обробка та розпізнавання запропонованого набору даних трафіку при використанні нейронних технологій.

Розробка програмного коду (Software engineering, software development) - це рід діяльності (професія) та процес, спрямований на створення та підтримку працездатності, якості та надійності програмного забезпечення, використовуючи технології, методологію та практики з інформатики, керування проектами, математики, інженерії та інших областей знаннь.

В даному дипломному проекті для прогнозуванню трафіку мережі на основі нейромереж було створено діаграму класів з різноманітними параметрами та методам, функціями та константами. Кожну з них треба описати для повного розуміння створеного продукту.

Запрограмувати нейронну мережу без використання різноманітних бібліотек, що полегшують нам написання коду, дуже складно. В даному дипломному проекті були використано бібліотеку PyBrain. Вона і являє собою основний механізм програми. Блок схему роботи даної бібліотеки показано на рис. 3.8.

PyBrian являє собою модульну бібліотеку призначену для реалізації різних алгоритмів машинного навчання на мові Python. Основною його метою є надання досліднику гнучких, простих у використанні, але в той же час потужних інструментів для реалізації завдань з області машинного навчання, тестування і порівняння ефективності різних алгоритмів.

Назва PyBrain є абревіатурою від англійського: Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network Library.

Як сказано на одному сайті: PyBrain - swiss army knife for neural networking (PyBrain – це швейцарський армійський ніж в області нейро-мережевих обчислень).

Рис. 3.8 – Блок схема роботи бібліотеки PyBrain

Виходячи з інформації показаної на блок схемі роботи бібліотеки PyBrain. Можна зробити висновок, що основний модуль навчання системи закладено сам в сюди. Тут відбувається левова частина обробки та перевірки заданих параметрів.

Бібліотека побудована за модульним принципом, що дозволяє використовувати її як студентам для навчання основам, так і дослідникам, які потребують реалізації більш складних алгоритмів.

Але жодна бібліотека не дасть нам бажаних результатів в роботі, тому необхідно створювати певні частини програми самотушки. Розглянемо створені частини нашого програмного засобу.

NeurophClassificationOperator – клас в якому збираються дані про завантажений файл, усі параметри файлу проходять первинну обробку саме у цьому класі. Цей клас містить чотири атрибути та чотири методи, розглянемо їх більш детально.

Атрибути які містяться в класі NeurophClassificationOperator.

– PARAMETR_NN_FILE – параметр, що виконує первинну обробку даних та визначає потрібну кількість ітерацій при навчанні програмного засобу.

– PARAMETR_MAX_ITERETIONS – параметр, що зберігає потрібну кількість ітерацій при навчанні програми та надає доступ до цих даних.

– nnFilePath – параметр, що надає доступ до обробленого файлу існуючим класам та функції вводу та виводу.

– fileIteretions – пакет для зберігання результатів ітерацій файлу.

Методи які містяться в класі NeurophClassificationOperator.

– NeurophClassificationOperator – метод для класифікації даних та роздачі їм відповідної ваги, залежно від внесених параметрів.

– Model_learn – метод, що дозволяє виконувати навчання опрацьованого набору даних. Перевірка поставлених вимог, щодо набору даних.

– Boolean_supportsCapability – знаходження параметрів, що не відповідають дійсності представленого набору (дуже великі числа, або нулі).

– ParametrType – присвоєння відповідних типів та опцій, залежно від якостей запропонованих параметрів.

NeurophClassificationAdapter – клас в якому відбувається аналіз запропонованого файлу з частками спектру та генерується результат розпізнавання. Цей клас містить один атрибут та сім методів, розглянемо їх більш детально.

Атрибути які містяться в класі NeurophClassificationAdapter.

– Maxiteretions – кількість ітерацій, залежно від чистоти даних та кількості.

Методи які містяться в класі NeurophClassificationAdapter.

– NeurophClassificationAdapter – класифікація запропонованих частин спекральних даних.

– trainNNModel – тренування нейронної мережі на основі запропонованих частин трафікових даних.

– TreiningSet – представлення натренованого набору даних.

– CreateOutputVectorMap – створення файлу для виводу на екран.

– getOutputVectorMap – виклик файлу для виводу на екран.

– getLabelIndex – отримання параметрів, що будуть впливати на набір навчання.

– checkNNCapability – перевірка отриманих параметрів, що будуть впливати на набір даних залежно від існуючого набору.

NeurophClassificationModel – бібліотеки та їх сумісна робота відбувається в даній частині додатку, цей клас відповідає за навчання мережі та даних. В даному класі міститься два атрибути та два методи, розглянемо їх більш детально.

Атрибути які містяться в класі NeurophClassificationModel.

– UID – атрибут який позначає певний набір та показує його номер.

– NeuralNetworknnnet – параметри заданої нейронної мережі.

Методи які містяться в класі NeurophClassificationModel.

– NeurophClassificationModel – модель, що класифікує параметри залежно від вимог PyBrain.

– Predict – результат які потрібно отримати залежно від вимог до набору.

Запропоновані методи та атрибути чітко та якісно синхронізуються з набором функціоналу представленою бібліотекою PyBrain.

3.3. Оцінювання програмного коду

Верифікація (підтвердження правильності) - складається в перевірці і підтвердженні правильності розробленої програми по відношенню до сукупності формальних претензій, викладених в специфікації і повністю визначає зв'язок між вхідними та вихідними даними цієї програми. При цьому відносини між змінними на вході і виході програми аналізується не у вигляді значень, як при

тестуванні, а у вигляді описів їх властивостей, що виявляються при будь-яких процесах обробок цих змінних контрольованої в програмі.

Верифікація програми в принципі виключає необхідність її тестування і налагодження, так як при цьому на більш високому рівні понять і описів всіх змінних, встановлюється коректність процесів, їх обробка і перетворення. Для своїх проектів я вирішив зекономити трохи часу і людських ресурсів, і використовувати автоматизовану перевірку коду проекту на відповідність PEP 8, а також якості коду (pyflakes) і якість повідомлень в Git. Повноцінний Code Review при цьому робити все рівно потрібно, але вже в менших обсягах, а іноді і зовсім можна їм знехтувати.

Отже, для роботи нам знадобиться встановити pep8, pyflakes, а також відредагувати на робочих станціях розробників для Git: "pre-commit" і "commit-msg".

Серверні частини спеціально не використовуються, так як вже буде проведений комміт і розробнику доведеться відкочуватися, це зовсім не економить час розробника.

У майбутньому планувалося додати туди перевірку докстрінгів (процентне співвідношення документації до коду, або будь-які інші метрики) і запуск тестів (але це сумнівно, так як позначиться на продуктивності виконання коммітів).

Для таких важких операцій як тести є зручний інструмент - Continuous Integraton (CI), такі як Hudson, Jenkins (django-jenkins), CruiseControl і Atlassian Bamboo.

```
$ git commit -m "Something I"
**********************************
***** CODE REVIEW IS FAILED *****
**********************************
Check files: <project>/<app>/views.py,
<project>/common/context_processors.py, <project>/common/decorators.py,
<project>/settings.py, <project>/<app>/models.py, <project>/urls.py
pep8:
./<project>/urls.py:14:19: E241 multiple spaces after ','
```

./<project>/<app>/models.py:18:1: E302 expected 2 blank lines, found 1

pyflakes:

./<project>/urls.py:3: 'include' imported but unused

./<project>/urls.py:4: undefined name 'User'

Модуль "pre-commit" перевіряє тільки staged зміни (тому після кожного фікса вам доведеться робити git-add), для того щоб перевіряти весь індекс відразу потрібно використовувати --debug:

$ .git/hooks/pre-commit --debug

Приклад перевірки повідомлення коммітів:

$ git commit -m "Something II"

****

***** CHECK COMMIT IS FAILED *****

****

> check_task_tracking_identificator:

>> Wrong Issue ID! Use format: [{project_id}-{issue_number}], example: [PROJ-1027]

Якщо додати до команди git-commit опцію --no-verify, то ви зможете пропустити перевірку модулів.

-n, --no-verify

This option bypasses the pre-commit and commit-msg hooks. See also githooks(5).

Такого виду контроль підвищує якості коду на вході в репозиторій, розробники залишають менше неточностей в коді. В цілому це мінімум, який потрібен для підвищення якості представленого дипломного проекту.

PyFence – самопальна утиліта-бібліотека, яка дозволяє стежити за відповідністю типів під час налагодження вашого проекту. PyFence бере інформацію про типах з docstring'ов функцій в стандартному форматі Sphinx. Тобто, якщо у вас вже є документація, більше нічого робити для використання PyFence не потрібно.

Наприклад, візьмемо наступний клас:

class RationalFormatter (object):

```
    def format(self, number):
        """
        Stringifies a number to numerator/denominator format
        Example::
            >>> print(RationalFormatter().format(1.25))
            5/4
        :param number: input number
        :type  number: float
        :raises      : None
        :rtype       : str
        """
        return '%i/%i' % number.as_integer_ratio()
    def display(self, number):
        print(str(number) + ' = ' + self.format(number))
```

Метод format представляє число у вигляді дробу, використовуючи метод float.as_integer_ratio().

Використаємо це:

```
from formatter import RationalFormatter
f = RationalFormatter()
f.display(1.25)
1.25 = 5/4
```

Начебто працює? Однак все зламається, якщо передати int, адже int, на жаль, as_integer_rati() не містить.

```
>>> f.display(5)
Traceback (most recent call last):
  File "example.py", line 5, in <module>
    f.display(5)
  File "/home/eugeny/Work/pyfence/example_formatter.py", line 18, in display
    print(str(number) + ' = ' + self.format(number))
```

Втім, при використанні PyFence таку проблему можна було б помітити набагато раніше. Крім того, PyFence може перевіряти типи, що виникають у функціях або методах виключень і повертаються типам. Можна також імпортувати модуль pyfence в самому проекті, в цьому випадку не доведеться використовувати окрему утиліту fence. Зрозуміло, використовувати fence варто тільки при розробці, а не в продакшн, так як через перевірки можливе падіння продуктивності.

Оцінка надійності програмного забезпечення є важливою частиною процесу розробки. Існуючі методи аналізу надійності програм засновані на використанні результатів тестування програми, програмних метрик, характеристик процесу розробки або програмної архітектури. Недоліком цих методів є використання непрямої інформації про програмні помилки, які є основною причиною недостатньої надійності програми.

Емпіричні моделі засновані на аналізі накопиченої інформації про функціонування раніше розроблених програм.

Найбільш проста емпірична модель пов'язує число помилок в програмному забезпеченні з його об'ємом. Досвідчені дані свідчать, що до початку системного тестування в програмному забезпеченні на кожні 1000 операторів припадає приблизно 10 помилок. Рівень надійності програмного забезпечення вважається прийнятним для початку експлуатації, якщо того ж обсягу операторів буде відповідати одна помилка.

Переваги та недоліки емпіричних моделей. Перевага емпіричних моделей в тому, що вони не містять складних формул і обчислення по ним прості.

До недоліків емпіричних моделей відноситься те, що вони дуже грубі, дуже приблизні. Крім того, вони не відображають динаміки обчислювального процесу при експлуатації програм.

Таким чином, в даний час в розпорядженні фахівців є достатня кількість емпіричних і аналітичних моделей, що забезпечують з тим або іншим ступенем точності розрахунок числових оцінок показників надійності програмного забезпечення на різних стадіях його життєвого циклу.

Аналізуючи моделі надійності програмного забезпечення, приходимо до висновку, що більшість з них визначає надійність програмного забезпечення на початкових стадіях життєвого циклу. Застосування розглянутих моделей для оцінки завершальних стадій життєвого циклу програмного забезпечення обмежено з наступних причин:

– на фазах виробництва та використання програмного забезпечення інформації про процес налагодження, виявленні та усуненні помилок, як правило, недоступні;

– відмови при приймально-здавальних випробуваннях малоінтенсивні або відсутні.

Тому для визначення надійності програмного забезпечення на всіх стадіях його життєвого циклу доцільно застосовувати, як мінімум, дві моделі надійності програмного забезпечення. Модель надійності програмного забезпечення для фази розробки вибирається для кожної конкретної програми. Для цього потрібно зібрати дані про помилки, на підставі наявних даних вибрати модель надійності, а потім виконати тести, які показують, наскільки ця модель підходить. Для визначення надійності програмного забезпечення на завершальних стадіях найбільш ефективно застосовувати моделі надійності з системно-незалежним аргументом, наприклад, модель Нельсона.

3.4. Висновки до розділу

Під час виконання програмної реалізації дипломного проекту було розроблено:

1) логіку роботи Інтернет-мережі;

2) програмний модуль обробки даних;

3) програмний модуль системи зв’язку;

Створено:

1) модель перевірки мережі Інтернет-трафіку;

2) алгоритм передачі даних від користувача до программи моніторингу

3) алгоритм збережння оброблених даних .

Дану систему розроблено за допомогою C++ та Python.

Впровадження розробленого сервісу дозволяє добитися економічного ефекту та зберегти час провайдерів внаслідок:

1) спрощення процесу передбачення помилок;

2) зниження часу на аналіз стану трафіку;

3) можливість прямої комунікації з системою, додаткова платформа для майбутніх оптимізацій системи.

Створену систему можуть використовувати як користувачі мережі інтернет, так і провайдери великих компаній.

ВИСНОВКИ

Атестаційна робота бакалавра присвячена актуальній тематиці створення підсистеми прогнозування трафіку комп'ютерної мережі.

Під час виконання атестаційнох роботи бакалавра було проаналізовано:

1) існуючі види мереж;
2) способи створення мережі;
3) етапи створення мережі.

Розроблено:

1) Алгоритм роботи мережі та її схему;
2) Діаграму класів та use-case діаграму.
3) Діаграму компонентів та блок схему роботи бібліотек.

При реалізації прогнозування було використано стек технологій, який дав змогу нам створити сервіс. Дану систему розроблено за допомогою Python та C++

Впровадження розробленого сервісу дозволяє добитися економічного ефекту та зберегти час провайдерів і користувачів за рахунок:

1) спрощення процесу передбачення помилок;
2) зниження часу на аналіз стану трафіку;
3) можливість прямої комунікації з користувачем.

Створену систему можуть використовувати як користувачі мережі інтернет, так і провайдери великих компаній.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Dover D., Dafforn E. Search Engine Optimization Secrets. Indianapolis: Wiley Publishing, Inc., 2011. 456 p.

2) «Анатомия беспроводных сетей» / Сергей Пахомов. – Компьютер-Пресс, №7,2002

3) Cardoso, J., Sheth, Amit (Eds.) ,. Semantic Neural Services, Processes and Applications. - Springer, 2006. - ISBN 0-387-30239-5.

4) Томас Коннолли. Базы данных. Проектирование, реализация и сопровождение. Теория и практика.

5) Роберт К. Элсенпитер, Тоби Дж. Велт. Нейронные сети строим сами. 2006. - 256 с.

6) ГОСТ Р 55060-2012 Системы управления трафик и сеть автоматизированные. Термины и определения.

7) Весоловский Кшиштоф. Системы подвижной радиосвязи. Горячая линия –Телеком, 2006

8) Григорьев В.А., О.И. Лагутенко, Ю.А. Распаев. Сети и системы радиодоступа. Эко-Трендз, 2005

9) WiMAX Forum [Електронний ресурс] / wimaxforum.org/

10) «WLAN: практическое руководство для администраторов и профессиональных пользователей» / Томас Мауфер. – М.: КУДИЦ-Образ, 2005

11) «Беспроводные сети. Первый шаг» / Джим Гейер. – М.: Издательство: Вильямс, 2005

12) Документы, созданные при участии АВОК - Ассоциация встроеного нахождения комп’ютерного трафика 12 дек 2016 - 320 с.

13) Марк Лутц. Программирование на Python / Пер. с англ. - Четвёртый изд. - СПб .: Символ-Плюс, 2011. - Т. I. - 992 с. - ISBN 978-5-93286-210-0.

14) Neural Network in school and house — Scout Blog, 7 дек 2016- 482 с. - ISBN 978-5-97060-315-4.

15) КоАП - Административные правонарушения в области связи и информации - Гарант, 7 дек 2016- СПб .: Символ-Плюс, 2010. - 560 с.

16) Кулик М.С., Полухін А.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2011. – 72 с.

17) ГОСТ .19.701-90 ЕСКД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения

18) ГОСТ 19.003-80 ЕСКД. Схемы алгоритмов и программ. Обозначения условные графические.

19) Берлин А.Н. Цифровые сотовые системы связи. М.: Эко-Трендз, 2007

20) В. Вишневский, С. Портной, И. Шахнович. Энциклопедия WiMAX. Путь к 4G. Техносфера, 2009

21) «Секреты беспроводных технологий» / Джек Маккалоу. – М.: НТ-Пресс, 2005

Додаток А

Лістинг вихідних кодів

sudo apt-get install pyflakes pep8

cd ~/work

git clone git://github.com/adw0rd/pre-code-review.git pre-code-review

ln -s /home/<username>/work/pre-code-review/pre-commit.py /home/<username>/work/<project>/.git/hooks/pre-commit

ln -s /home/<username>/work/pre-code-review/commit-msg.py /home/<username>/work/<project>/.git/hooks/commit-msg

chmod +x /home/<username>/work/<project>/.git/hooks/pre-commit

chmod +x /home/<username>/work/<project>/.git/hooks/commit-msg

$ git commit -m "Something I"

****

***** CODE REVIEW IS FAILED *****

****

Check files: <project>/<app>/views.py, <project>/common/context_processors.py, <project>/common/decorators.py, <project>/settings.py, <project>/<app>/models.py, <project>/urls.py

pep8:

./<project>/urls.py:14:19: E241 multiple spaces after ','

./<project>/<app>/models.py:18:1: E302 expected 2 blank lines, found 1

pyflakes:

./<project>/urls.py:3: 'include' imported but unused

./<project>/urls.py:4: undefined name 'User'

$ .git/hooks/pre-commit --debug

Приклад перевірки повідомлення коммітів:

$ git commit -m "Something II"

****

***** CHECK COMMIT IS FAILED *****

****

> check_task_tracking_identificator:

>> Wrong Issue ID! Use format: [{project_id}-{issue_number}], example: [PROJ-1027]

-n, --no-verify

This option bypasses the pre-commit and commit-msg hooks. See also githooks(5).

class RationalFormatter (object):

def format(self, number):

"""

Stringifies a number to numerator/denominator format

Example::

>>> print(RationalFormatter().format(1.25))

5/4

:param number: input number

:type number: float

:raises   : None

:rtype   : str

"""

return '%i/%i' % number.as_integer_ratio()

def display(self, number):

print(str(number) + ' = ' + self.format(number))

float.as_integer_ratio().

from formatter import RationalFormatter

f = RationalFormatter()

f.display(1.25)

1.25 = 5/4

>>> f.display(5)

Traceback (most recent call last):

File "example.py", line 5, in <module>

f.display(5)

File "/home/eugeny/Work/pyfence/example_formatter.py", line 18, in display

print(str(number) + ' = ' + self.format(number))

```
File "/home/eugeny/Work/pyfence/example_formatter.py", line 15, in format
    return '%i/%i' % number.as_integer_ratio()
AttributeError: 'int' object has no attribute 'as_integer_ratio'
```