Зміст

ВСТУП

Детектування та аналіз емотиконів є необхідною задачею в системах аналізу повідомлень. Зміст висловленого користувачем може мінятися в залежності від вживання наборів символів котрі відображають емоційний стан. Наявність певної невизначеності у вихідних даних дає змогу говорити, що вказане завдання доцільно вирішувати стохастичними алгоритмами .

Актуальність теми. Системи розпізнавання повинні вибрати емотикони з тексту і визначити емоції котрі вони виражають. Вирішення цього завдання є важливою для використання в системах аналізу повідомлень для зменшення омонімії. Існуючі на даний момент розробки та дослідження не виконують це завдання в повній мірі.

Об’єктом дослідження стали символи емоційного забарвлення текстових повідомлень.

Метою наукової роботи є дослідити використання прихованої марківської моделі для використання в системах детектування емотиконів, завдання доцільне через те, що детектування смайлів вписується в модель в котрій наявні приховані та видимі стани параметрів.

Незважаючи на те, що є величезна кількість інших можливих варіантів виконання даного завдання, майже всі вони враховують використання регулярних виразів, а отже *науковою новизною* є використання прихованої марківської моделі в детектуванні символів емоційного наповнення.

В результаті виконання кваліфікаційної роботи на здобуття рівня магістра буде розроблено алгоритм та програма детектування емотиконів з використанням прихованої марківської моделі. Розроблена система буде давати змогу зменшувати омонімію вхідних в системах семантичного та лексичного аналізу.

Тема даного кваліфікаційної роботи була апробована на на VI студентській науково-технічній конференції ТНТУ ім. І. Пулюя “Природничі та гуманітарні науки. Актуальні питання”

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД

1.1. Вступ в предметну область

Кінесика – сукупність рухів (жестів, міміки), що застосовуються в процесі людського спілкування (за винятком рухів мовного апарату). Емотикони представляють мову тіла в он-лайн текстових системах зв’язку. Саме тому кінесика підходить для дослідження смайлів також.

Кінесика вивчає відображення поведінки людини в її невербальних проявах , до яких відносяться міміка (рух м'язів обличчя) , пантоміміка (рухи всього тіла) , " вокальна міміка " (інтонація , тембр , ритм , вібрато голосу) , просторовий малюнок (зона , територія , власність і переміщення) , експресія (виразність , сила прояву почуттів , переживань) , яка може бути вирішальною в інтерпретації вимовлених висловлювань.

В зв’язку з поширенням електронних засобів комунікацій проблема аналізу повідомлення та уникнення омонімії спричиненою емотиконами повідомлення стає дуже нагальною.

Смайли, або емотикони умовно можна розділити на три типи 1) західного зразку, 2) східного(японські смайли), 3) багаторядковий ASCII-арт. Виявлення та аналіз ASCII-арту відноситься до галузі аналізу зображень тому такий вид смайлів не буде розглядатися. Тому, відповідно, можна характеризувати смайл, як однорядкову стрічку котра має хоча б одну – семантичну ділянку, наприклад: «Рот»[M], «Очі»[E_L][E_R], «Межі смайлу»[B_1][B_2](вуха, щоки), «Додаткові ділянки» [S_1]-[S_4] розміщені всередині, або зовні. Кожна ділянка може мати будь-яку кількість символів. Також деякі з них можуть бути пустими.

1.2. Аналіз існуючих систем

Потрібність проведення дослідження та розробки системи в обраному напрямку підтверджується тим, що на даний час ведеться наукова діяльність в даному напрямку, але кількість розроблених систем остається не великою.

Візьмемо до прикладу алгоритм розроблений в Oregon Health & Science University пропонує новий метод виявлення меж смайлів, а саме – метод графічної схожості. Такий підхід є логічним так, як емотикони використовуються якраз для створення графічного ефекту, а не для передачі текстової інформації. Розробники алгоритму перетворювали кожний гліф котрий був отриманий з інтернету у бітову карту(bitmap), після чого, виконавши масштабування щоб два гліфи співпадали, порівнювали два символи. Шкала збігу пікселів була наступною: 1 – повний збіг, 0 – нема суміжних пікселів. Символи порівнювались, як в простому вигляді так і в вертикально відображеному.

Система CAO (emotiCon Analysis and decOding of affective information) – розроблена Міхалем Птяжинським та іншими науковцями в університеті Хокайдо, опирається на алгоритм з поступовою релаксацією в своїй системі поступового аналізу. Після збору бази даних емотиконів і класифікації їх відповідно до типу емоцій в системі CAO, виконується виокремлення всіх семантичних ділянок. Спочатку знаходяться межі емотикона (котрі можуть мати значення ([{<|та їх інверсії) і виділяється триплет (ELMER). Слід зауважити, що межі виступають своєрідними якорями для визначення центру емотикону. Після цього з цього триплету видобуваються рот (M) і пара очей (EL,ER). Для майже всіх каомоджі виконується наступне: якщо очі складаються з декількох символів то послідовність цих символі для кожного ока є однакова і якщо очі – одно символьні, то вони можуть бути, як однакові так і різні. Після того, як триплети та межі смайлу виокремлено видобуваються додаткові ділянки (S1-S4)

використовуючи регулярний вираз S1B1S2ELMERS3B2S4. Розділивши емотикони на семантичні ділянки, розраховується частота повторення цих ділянок в базі даних типу емоцій. Всі унікальні ділянки додаються в порядку частоти повторень в базі для кожного типу емоцій. Частота повторень кожної ділянки вказує на ймовірність виражати певну емоцію.

Слід сказати, що в системі CAO було зібрано 1654 унікальних триплетів EL,ER , унікальних ротів M 1654, різних додаткових ділянок відповідно 20 184. Комбінацій які можливо згенерувати тільки з EL ,ER *M=3 175 680. З додатковими ділянками Sn така кількість зростає до 1,382613544823877 × 1021. На даний час існує небагато систем котрі здатні детектувати та аналізувати емотикони будь-якої складності. Більшість діючих та впроваджених систем включать недоліки попередніх та впроваджують новий функціонал. Слід зазначити, що всі існуючі системи мають похибки в детектуванні і практично неможливо розробити алгоритм котрий повністю детектував всі емотикони в вхідних даних, в зв’язку з креативністю користувачів та наявністю в вхідних даних інших символів пунктуації крім тих, що відносяться до смайлів.

У системі аналізу CAO першим ділом, при отримані даних з входу, аналізується присутність емотиконів. Присутність емотиконів визнається, якщо що найменше три символи котрі найчастіше використовуються в смайлах, попадаються в ряд. Для таких цілей в системі CAO, складається набір 455 символів котрі відібрані статистично. Слід згадати, що система CAO має майже ідеальну точність 99.5%.

Видобування емотикона з вхідних даних здійснюється в три кроки вхідні дані порівнюються з:

Звичайною базою даних смайлів, якщо немає збігів то порівнюються триплети (око, рот/ніс,око) з триплетами з бази даних, якщо триплет знайдено то система використовує всі бази даних з межами та додатковими ділянками

порівнюючи дані з них за допомогою регулярного виразу m/[S1?][B1?][S2?][EL M ER][S3?][B2?][S4?]/.

Якщо триплет не знайдено то система шукає:

Збіжності між будь якими комбінаціями очей, ротів, носів([EL][M][ER],[EL][ER],[M][ER],[EL][M]) і останній метод: шукати збіжності між всіма елементами окремо.

Процедуру видобування також можна б було використати, як процедуру виявлення, але вона забирає більше часу, тому для опрацювання великих об'ємів даних він не підходить. В аналізі процедури впливу, система оцінює який тип емоцій найбільш вірогідно виражає смайл. Аналіз здійснюється порівнянням розпізнаних емотиконів до тих, котрі анотовані в базі даних, і звіряючи статистику їх поширеності. Ця процедура виконується в додаток до процедури видобування. Спочатку вхідний смайл порівнюється із зразками бази даних емоцій неопрацьованих смайлів. Якщо такої емоції не знайдено, то перевіряється база даних з триплетами ELMER. Якщо тут немає збігів, то семантичні ділянки з бази даних для очей ELER і роту M порівнюються окремо.

Незважаючи на простоту системи CAO вона має ряд суттєвих недоліків: виявленні смайлів існує похибка у 2,4%, також система може неправильно виявити межі емотикону на етапі виокремлення семантичних ділянок у випадку, якщо використані нестандартні символи замість дужок різного виду,також така проблема буде спостерігатися з очима із різних символів.

Таїчі Ямада та ін. використали статистику N-Грам для визначення типу емоцій котрі передаються емотиконом. Для класифікації смайлів вони використали просту статистику символів котрі появляються в смайлах, без розбиття їх на семантичні ділянки. З часом такий підхід почав давати помилки, а саме виявляти ділянки невірно, як от вказувати на «очі» замість «роту». Однак такий метод демонстрував високі показники точності (від 76% до 83%).

Метод раціонального зерна для видобування та класифікації емотиконів був запропонований Юкі Танакою та ін. Вони використали популярний в Японії інструмент для обробки речень – POS tagger ChaSen і програму yamcha (котра працює за методом опорних векторів), щоб поділити речення і виділити частини мови від «інших елементів», котрі скоріше всього будуть являти собою емотикони.

Цей метод може працювати тільки з реченнями котрі поділені на частини мови і також деколи вважає за емотикони інші символьні позначення. Незважаючи на це, він показує 85.5% точності.

РОЗДІЛ 2
ОБГРУНТУВАННЯ ВИБРАНОГО НАПРЯМКУ РОБОТИ

2.1. Обґрунтування вибору математичного апарату

2.1.1. Опис прихованих марківських моделей

Прихована марківська модель (ПММ) – частковий випадок Байесової мережі довіри. Дана модель підходить для моделювання об’єкту дослідження через наявність двох станів для символів терміналів у створюваній системі. Іншими словами прихована марківська модель являється статистичною моделлю, котра імітує роботу процесу, схожого на Марківський процес, який містить невідомі параметри, і завданням ставиться розгадування невідомих параметрів на основі спостережуваних. Отримані параметри можуть бути використані в подальшому аналізі, наприклад, для розпізнавання образів.

Ма́рківський процес – це випадковий процес, конкретні значення якого для будь-якого заданого часового параметру t+1 залежать від значення у момент часу t, але не залежать від його значень у моменти часу t-1, t-2 і т. д. (дискретний випадок марківського процесу). Іншими словами «майбутнє» процесу залежить лише від «поточного» стану, але не залежить від «минулого» (за умови, коли «поточний» стан процесу відомий).

Властивість Маркова – це властивість яка характеризує процес як марківський. Вперше цю властивість було сформульовано російським математиком Марковим А. А., який 1907 року поклав початок вивченню послідовностей залежних випробувань і пов'язаних із ними сум випадкових величин. Цей напрямок досліджень відомий зараз під назвою теорії ланцюгів Маркова. Основне застосування ПММ отримали в області розпізнавання мовлення, письма, рухів та біоінформатики. Крім того, ПММ застосовуються в криптоаналізі, машинному перекладі.

У звичайній марківській моделі стан видно спостерігачеві, тому ймовірності переходів – єдиний необхідний параметр.

У прихованій марківській моделі ми можемо стежити лише за змінними, на які впливає цей стан. Кожен стан має ймовірнісний розподіл серед усіх можливих вихідних значень. Тому послідовність символів, згенерована ПММ, дає інформацію про послідовність станів.

Діаграма, подана нижче, показує загальну структуру ПММ (див. рис. 2.1). Овали – це змінні з випадковим значенням. Випадкова змінна відповідає значенню прихованої змінної в момент часу . Випадкова змінна – це значення змінної, за якою ми спостерігаємо, в момент часу . Стрілки на діаграмі символізують умовні залежності.

Із діаграми можна дізнатись, що значення прихованої змінної (в момент часу) залежить тільки від значення прихованої змінної (в момент часу). Це називається властивістю Маркова. Хоча в той же час значення змінної , за якою ми спостерігаємо, залежить лише від значення прихованої змінної (в момент часу).

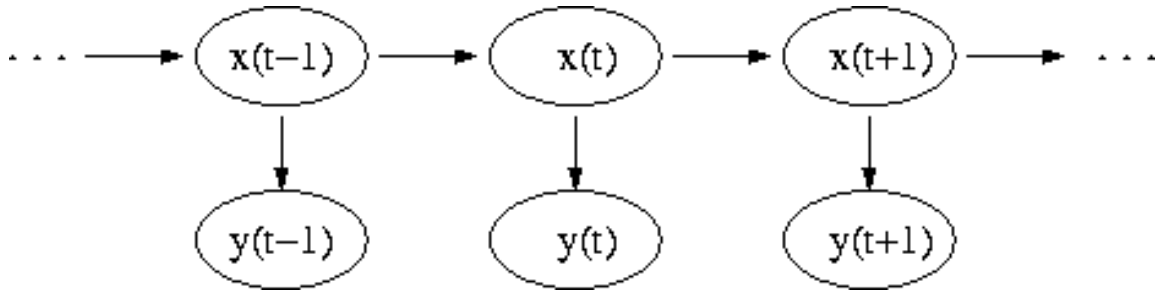

Рисунок 2.1 – Діаграма ПММ

В Прихованій марківській мережі ймовірність спостерігати послідовність $Y = y(0), y(1), ..., y(L-1)$ довжини L дорівнює:

$$P(Y) = \sum_{X} P(Y \mid X) P(X), \qquad (2.1)$$

тут сума пробігає по всіх можливих послідовностях прихованих вузлів $X = x(0), x(1), ..., x(L-1).$

Метод підрахунку повним перебором значень $P(Y)$ – дуже трудомісткий для багатьох реальних задач внаслідок того, що кількість можливих послідовностей прихованих вузлів дуже велика. Але застосування процедури вперед-назад дозволяє істотно збільшити швидкість обчислень.

Основні приховані марківські моделі можна описати за допомогою таких змінних:

- N – кількість станів;
- T – кількість спостережень;
- $\theta_{i=1...N}$ – параметр для спостереження за зв'язками між станами;
- $\phi_{i=1...N,j=1...N}$ – ймовірність переходу із стану до стану ;
- $\boldsymbol{\phi}_{i=1...N}$ – N -мірний вектор, що складається із ;
- $x_{t=1...T}$ – стан спостереження за час t;
- $y_{t=1...T}$ – результат спостереження за час t;
- $F(y|\theta)$ – функція розподілу ймовірності спостережень, параметризованих по θ.

2.1.2. Означення поняття стохастичної контекстно-вільної граматики

Слід вказати на те, що для розбору емотикони потрібно провести індукцію стохастичної контекстно-вільної граматики.

Формальна граматика або просто граматика в теорії формальних мов — спосіб опису формальної мови, тобто виділення деякої підмножини з множини всіх слів деякого скінченного алфавіту.

Ідея формальних мов полягає в тому, аби створити точний математичний опис для подібних мов. На основі цього опису можна будувати засоби для автоматичної обробки цих мов.

Розрізняють породжувальні і аналітичні граматики — перші ставлять правила, за допомогою яких можна побудувати будь-яке слово мови, а другі дозволяють по даному слову визначити, входить воно в мову чи ні. Формальні граматики були введені американським вченим, математиком та філософом, Н. Хомським в 50-тих роках XX сторіччя.

Стохастична контекстно-вільна граматика (СКВ, також імовірнісна контекстно-вільна граматика, ІКВ) – це контекстно-вільна граматика, в якій кожному правилу висновку відповідає ймовірність. Імовірність висновку визначається як добуток ймовірностей використовуваних в ньому правил виводу, таким чином, деякі висновки краще відповідають стохастичній граматиці ніж інші.

СКВ-граматики розширюють КВ(контекстно-вільної)-граматики так само як приховані марківські моделі розширюють регулярні граматики. СКВ-граматики широко застосовуються в науці: від обробки природних мов до вивчення молекул РНК. СКВ-граматики є особливою формою зважених контекстно-вільних граматик.

Для контекстно-вільних граматик визначено різні нормальні форми. В нормальних формах Чомскі (НФЧ)скорочують праву частину правил виводу, тобто, права частина може складатись або з одного термінального символу, або з двох нетермінальних. Якщо в лівій частині знаходиться початковий символ, права частина може породжувати порожнє слово. Існує алгоритм, який переводить довільну контекстно-вільну граматику в НФЧ.

Контекстно-вільна граматика визначена в нормальній формі Грейбах, якщо вона не породжує порожнього слова та в права частина правил виводу починається з щонайбільше одного термінального символу, слід за яким йдуть нетермінальні символи. Кожна контекстно-вільна граматика, яка не породжує порожнє слово, може бути перетворена в нормальну форму Грейбеха алгоритмом.

2.2 Обґрунтування технічних засобів виконання завдання

2.2.1 Обґрунтування вибору середовища виконання додатку

Додаток розроблюваний для вирішення поставленої задачі буде виконуватись віддалено взаємодіючи з користвачем через мережу Internet .

Серверне програмне забезпечення (сервер, англ. Server від to serve – служити; множина сервери) – в інформаційних технологіях – програмний компонент обчислювальної системи, що виконує сервісні (обслуговуючі) функції по запиту клієнта, надаючи йому доступ до певних ресурсів або послуг.

Поняття сервер і клієнт і закріплені за ними ролі утворюють програмну концепцію «клієнт-сервер».

Для взаємодії з клієнтом (або клієнтами, якщо підтримується одночасна робота з декількома клієнтами) сервер виділяє необхідні ресурси між процесами взаємодії (колективна пам'ять, пайп, сокет і т. п.) і очікує запити на відкриття з'єднання (чи, власне, запити на наданий сервіс). Залежно від типу такого ресурсу, сервер може обслуговувати процеси в межах однієї комп'ютерної системи або процеси на інших машинах через канали передачі даних (наприклад, COM-порт) або мережеві з'єднання.

Формат запитів клієнта і відповідей сервера визначається протоколом. Специфікації відкритих протоколів описуються відкритими стандартами, наприклад протоколи Інтернету визначаються в документах RFC.

Залежно від виконуваних завдань одні сервери, за відсутності запитів на обслуговування, можуть простоювати в очікуванні.

Інші можуть виконувати якусь роботу (наприклад, роботу по збору інформації), у таких серверів робота з клієнтами може бути другорядною завданням.

Апаратними серверами називаються вузькоспеціалізовані рішення з вбудованим програмним забезпеченням (англ. firmware; на відміну від комп'ютерів, де програмне забезпечення необхідно встановлювати), визначальним спеціалізацію та можливі надані послуги. Апаратні сервера, як правило, більш прості і надійні в експлуатації, споживають менше електроенергії і, іноді, більш дешеві. Але разом з тим вони менш гнучкі (так як спочатку обмежені у виконуваних завданнях) і часто обмежені в ресурсах.

Важливо розуміти, що сервер, в тому значенні, яке мається на увазі в даній кваліфікаційній роботі(тобто сервер, що надає сервіс, наприклад проксі-сервер), завжди є програмою (або програмним модулем), що виконується на якомусь апаратному забезпеченні. Без цієї програми апаратне забезпечення не може нічого надавати. Навіть «апаратні сервера» (або роутери) не виключення, тому що в них сервіс також надається (вбудованим) програмним забезпеченням. Іноді, для простоти, сервером послуги (наприклад тим же проксі-сервером) називають програмне і апаратне забезпечення в цілому, особливо якщо цей програмно-апаратний комплекс виконує тільки одну задачу.

Теоретично на одній одиниці апаратного забезпечення може одночасно виконуватися довільна кількість серверів (за винятком серверів, конфліктуючих між собою по ресурсах або їх кількістю), вони будуть ділити між собою апаратні ресурси. Практично, між крайнощами «один комп'ютер – одна послуга» і «один комп'ютер – всі послуги» кожен знаходить свій компроміс.

Сервери послуг можна запускати на робочій станції, щоб вони працювали у фоновому режимі, розділяючи ресурси комп'ютера з програмами, що запускаються користувачем. Такий режим роботи називається «невиділеним»,

на відміну від «виділеного» (англ. dedicated), коли комп'ютер виконує тільки сервісні функції. Строго кажучи, на робочій станції (для прикладу, під керуванням Windows XP) і без того завжди працює кілька серверів – сервер віддаленого доступу (термінальний сервер), сервер віддаленого доступу до файлової системи і системі друку та інші віддалені і внутрішні сервера.

Як правило, кожен сервер обслуговує один або декілька схожих протоколів. Сервера можна класифікувати за типом послуг, які вони надають.

Універсальні сервера – особливий вид серверної програми, який не надає жодних послуг самостійно. Замість цього універсальні сервера надають серверів послуг спрощений інтерфейс до ресурсів між процесами взаємодії та уніфікований доступ клієнтів до різних послуг. Існують кілька видів таких серверів:

1. inetd (від англ. internet super-server daemon – демон сервісів IP) – стандартний засіб UNIX-систем – програма, що дозволяє писати сервера TCP / IP (і мережевих протоколів інших родин), що працюють з клієнтом через переслані inetd потоки стандартного введення і виведення (stdin і stdout).
2. RPC (від англ. Remote Procedure Call – віддалений виклик процедур) – система інтеграції серверів у вигляді процедур, доступних для виклику віддаленим користувачем через уніфікований інтерфейс.
3. Прикладні клієнт-серверні технології Windows:

   3.1. (D-) COM (англ. (Distributed) Component Object Model – модель складових об'єктів) та ін. – Дозволяє одним програмами виконувати операції над об'єктами даних, використовуючи процедури інших програм. COM працює лише в межах одного комп'ютера, DCOM доступна віддалено через RPC.

   3.2. Active-X – Розширення COM і DCOM для створення мультимедійних додатків. Універсальні сервера часто використовуються для написання всіляких інформаційних серверів

– серверів, що не потребують специфічної роботи з мережею і не мають ніяких завдань, крім обслуговування клієнтів. Наприклад, в ролі серверів для inetd можуть виступати звичайні консольні програми та скрипти.

Більшість внутрішніх та мережевих специфічних серверів Windows працюють через універсальні сервера (RPC, (D-) COM).

Строго кажучи, сервіс маршрутизації не є сервісом в класичному сенсі, а є базовою функцією підтримки мережі операційною системою.

Для TCP / IP маршрутизація є базовою функцією стека IP (коду підтримки TCP / IP). Маршрутизацію своїх пакетів до місця призначення виконує будь-яка система в мережі, маршрутизацію ж чужих пакетів (форвардного) виконують тільки маршрутизатори (також відомі як роутери або шлюзи). Завдання маршрутизатора при Форвардінг пакету:

1) прийняти пакет;
2) знайти машину, на яку слід цей пакет, або наступний маршрутизатор за маршрутом до неї (у таблиці маршрутів);
3) передати пакет або повернути ICMP-повідомлення про неможливість його доставки з причин:
    1) призначення недосяжно (англ. Destination unreachable) – у пакета скінчилося «час життя» перш ніж він досяг місця призначення;
    2) хост недосяжний (Host unreachable) – комп'ютер або наступний маршрутизатор вимкнено або не існує;
    3) мережа недосяжна (Network unreachable) – маршрутизатор не має маршруту в мережу призначення.
4) якщо пакет не може бути доставлений через перевантаження маршрутизатора (або мережі) – відкинути пакет без сповіщень.

Розв'язки динамічної маршрутизації покликані збирати інформацію про поточний стан складної мережі і підтримувати таблицю маршрутів через цю

мережу, щоб забезпечити доставку пакета за найкоротшим і найефективнішим маршрутом.

З цих рішень клієнт-серверну модель використовує тільки BGP (англ. Border Gateway Protocol – протокол прикордонного шлюзу), застосовуваний для глобальної маршрутизації. Локальні рішення (RIP OSPF) використовують у своїй роботі бродкастові та мультикастові розсилки.

Мережеві служби забезпечують функціонування мережі; наприклад, сервера DHCP і BOOTP забезпечують стартову ініціалізацію серверів і робочих станцій, DNS – трансляцію імен в адреси і навпаки.

Сервера тунелювання (наприклад, різні VPN-сервера) і проксі-сервера забезпечують зв'язок з мережею, недоступною роутингом.

Сервера AAA і Radius забезпечують у мережі єдину аутентифікацію, авторизацію та ведення логів доступу.

До інформаційних служб можна віднести як найпростіші сервера, що повідомляють інформацію про хост (time, daytime, motd) і користувачах (finger, ident), так і сервера для моніторингу, наприклад SNMP. Більшість інформаційних служб працюють через універсальні сервера.

Особливим видом інформаційних служб є сервера синхронізації часу – NTP; крім інформування клієнта про точний час NTP-сервер періодично опитує кілька інших серверів на предмет корекції власного часу. Крім часу, аналізується і коректується швидкість ходу системних годин. Корекція часу здійснюється прискоренням або уповільненням ходу системних годин (залежно від напрямку корекції), щоб уникнути проблем, можливих при простій перестановці часу.

Файлові сервера являють собою сервера для забезпечення доступу до файлів на диску сервера.

Перш за все це сервера передачі файлів за замовленням, по протоколах FTP, TFTP, SFTP і HTTP. Протокол HTTP орієнтований на передачу текстових

файлів, але сервера можуть віддавати як запитаних файлів і довільні дані, наприклад динамічно створені веб-сторінки, картинки, музику і т. п.

Інші сервера дозволяють монтувати дискові розділи сервера в дисковий простір клієнта і повноцінно працювати з файлами на них. Це дозволяють сервера протоколів NFS і SMB. Сервери NFS і SMB працюють через інтерфейс RPC.

Недоліки файл-серверної системи:

- Дуже велике навантаження на мережу, підвищені вимоги до пропускної здатності. На практиці це робить практично неможливою одночасну роботу великої кількості користувачів з великими обсягами даних.
- Обробка даних здійснюється на комп'ютері користувача. Це тягне підвищені вимоги до апаратного забезпечення кожного користувача. Чим більше користувачів, тим більше грошей доведеться витратити на оснащення їх комп'ютерів.
- Блокування даних при редагуванні одним користувачем робить неможливою роботу з цими даними інших користувачів.
- Безпека. Для забезпечення можливості роботи з такою системою Вам буде необхідно дати кожному користувачеві повний доступ до цілого файлу, в якому його може цікавити тільки одне поле.

Сервера доступу до даних обслуговують базу даних і віддають дані за запитами. Один з найпростіших сервісів подібного типу – LDAP (англ. Lightweight Directory Access Protocol – полегшений протокол доступу до списків).

Для доступу до серверів баз даних єдиного протоколу не існує, проте всі сервери баз даних об'єднує використання єдиних правил формування запитів – мови SQL (англ. Structured Query Language – мова структурованих запитів).

Служби обміну повідомленнями дозволяють користувачеві передавати і отримувати повідомлення (зазвичай – текстові).

У першу чергу це сервера електронної пошти, що працюють по протоколу SMTP. SMTP-сервер приймає повідомлення і доставляє його в локальну поштову скриньку користувача або на інший SMTP-сервер (сервер призначення або проміжний). На багатокористувацьких комп'ютерах користувачі працюють з поштою прямо на терміналі (або у веб-інтерфейсі). Для роботи з поштою на персональному комп'ютері пошта забирається з поштової скриньки через сервери, що працюють по протоколах POP3 або IMAP.

Для організації конференцій існує сервера новин, що працюють по протоколу NNTP.

Сервера віддаленого доступу, через відповідну клієнтську програму, забезпечують користувача аналогом локального терміналу (текстового або графічного) для роботи на віддаленій системі.

Для забезпечення доступу до командного рядка служать сервера telnet, RSH і SSH.

Графічний інтерфейс для Unix-систем – X Window System – має вбудований сервер віддаленого доступу, так як з такою можливістю розроблявся спочатку. Іноді можливість віддаленого доступу до інтерфейсу X-Window неправильно називають «X-Server» (цим терміном в X-Window називають відеодрайвер).

Стандартний сервер віддаленого доступу до графічного інтерфейсу Microsoft Windows називається термінальний сервер.

Деякий різновид управління (точніше, моніторингу та конфігурування) також надає протокол SNMP. Комп'ютер або апаратний пристрій для цього повинно мати SNMP-сервер.

2.2.2 Обґрунтування вибору парадигми програмування

PHP (англ. PHP:Hypertext) – скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-серверу. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. Проект за яким був створений PHP — проект з відкритими програмними кодами.

PHP інтерпретується веб-сервером в HTML-код, який передається на сторону клієнта. На відміну від таких скриптових мов програмування, як JavaScript, користувач не має доступу до PHP-коду, що є перевагою з точки зору безпеки але значно погіршує інтерактивність сторінок. Але ніщо не забороняє використовувати PHP для генерування і JavaScript-кодів які виконаються вже на стороні клієнта.

PHP - мова, яка може бути вбудованою безпосередньо в html-код сторінок, які, в свою чергу коректно будуть оброблені PHP - інтерпретатором. Механізм PHP просто починає виконувати код після першої екрануючої послідовності (<?) і продовжує виконання до того моменту, коли він зустріне парну екрануючу послідовність (?>).

Велика різноманітність функцій PHP дають можливість уникнути написання багаторядкових призначених для користувача функцій на C або Pascal.

Наявність інтерфейсів до багатьох баз даних:

– в PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase;
– через стандарт відкритого інтерфейсу зв’язку з базами даних (Open Database Connectivity Standard — ODBC) можна підключатися до всіх баз даних, до яких існує драйвер.

Мова PHP має низькій поріг входу. Багато конструкцій мови запозичені з C, Perl. Код PHP дуже схожий на той, який зустрічається в типових програмах на C або Pascal. Це помітно знижує початкові зусилля при вивченні PHP. PHP - мова, що поєднує переваги Perl і C і спеціально спрямована на роботу в Інтернеті, мова з універсальним і зрозумілим синтаксисом. І хоча PHP є досить молодою мовою, вона знайшла таку популярність серед веб-програмістів, що на даний момент є мало не найпопулярнішою мовою для створення веб-додатків (скриптів).

Стратегія Open Source, і розповсюдження початкових текстів програм в масах, зробило поза сумнівом благотворний вплив на багато проектів, в першу чергу - Linux хоч і успіх проекту Apache сильно підкріпив позиції прихильників Open Source. Сказане відноситься і до історії створення PHP, оскільки підтримка користувачів зі всього світу виявилася дуже важливим чинником в розвитку проекту PHP. Ухвалення стратегії Open Source і безкоштовне розповсюдження початкових текстів PHP надало неоціниму послугу користувачам. Додатково, чуйне співтовариство користувачів PHP є свого роду колективною службою підтримки, і в популярних електронних конференціях можна знайти відповіді навіть на найскладніші питання.

Ефективність є виключно важливим чинником при програмуванні для розрахованих на багато користувачів середовищ, до яких належить і веб. Дуже важлива перевага PHP полягає в його транслюючому інтерпретаторі. Такий пристрій дозволяє обробляти сценарії з достатньо високою швидкістю. За деякими оцінками, більшість PHP-сценаріїв (особливо не дуже великих розмірів) обробляються швидше за аналогічні їм програми, написані на Perl. Проте, щоб не робили розробники PHP, виконувані файли, що відкомпілювалися, працюватимуть значно швидше - в десятки, а іноді і в сотні разів. Але продуктивність PHP цілком достатня для створення цілком серйозних веб-додатків.

З точки зору типізації, PHP є мовою програмування з динамічною типізацією. Немає необхідності явного визначення типу змінних, хоча така можливість існує. В разі звернення до змінної, ядро PHP трактує її тип відповідно до контексту. За необхідності можливе приведення змінної до певного типу за допомогою відповідних конструкцій мови. Це може знадобитись, якщо зважити, що значення змінної можуть трактуватись по-різному в залежності від її типу. Також можливе визначення типу відповідної змінної на певному етапі виконання сценарію. Імена змінних чутливі до регістру символів.

До базових типів належать булеві дані, цілі та дійсні числа із плаваючою комою, а також строки. Булеві дані виражають істинність значення. Цілі числа можуть бути подані у вісімковому, десятковому та шістнадцятковому вигляді. Розмір цілого числа може змінюватись залежно від платформи, як правило, розрядність становить 32 біти. PHP не підтримує беззнакові цілі числа. Дійсні числа із плаваючою комою можуть бути подані в десятковій або експоненційній формі. Строки розділяють на два класи — строки, що підлягають аналізу, та строки, що не підлягають аналізу. Перший клас досліджується інтерпретатором на наявність посилань на інші змінні і за умови їх наявності, відбувається підстановка значень у відповідне місце. Крім того, клас дозволяє проводити маніпуляції із керуючими символами. Символ строки може мати лише одне із 256 значень, але існує можливість працювати із багатобайтовими символами. Доступ до символів строки можливий з використанням синтаксису, схожого на доступ до елементів масивів.

PHP надає широкий спектр функцій для пошуку та заміни тексту в строках. Для цього використовують як традиційний підхід, так і спеціальний підхід, що базується на використанні регулярних виразів. При цьому в мові реалізована підтримка двох видів регулярних виразів — Perl-сумісні та POSIX-сумісні, що розрізняються за синтаксисом та особливостями роботи.

До змішаних типів належать масиви, хеші та об’єкти. Масиви в сенсі мови є наборами змінних, що згруповані в єдину змінну. Вимога однотипності наповнення масивів не ставиться. Технічно, масиви являють собою впорядковані карти, що відображають ключові значення на позиції змінних даних. Вмістом значення, на яке вказує ключ може бути будь-чим, що можна подати у вигляді змінної. Не існує жодних обмежень, крім об’єму пам’яті, що накладаються на кількість ключів масиву.

Особливістю мови є відмова від рівномірного розподілу ключів масивів. Знайшла реалізацію і модель багатовимірних масивів, причому без явного обмеження глибини вкладеності. Корисною властивістю PHP є можливість асоціації масивів із функцією зворотного виклику. Ці функції дозволяють проводити дії над одним чи кількома масивами в пакетному режимі.

Також, існують два спеціальні типи даних — ресурс та NULL. Ресурс — спеціальна змінна, що містить посилання на зовнішній ресурс. Ресурси створюються та використовуються в спеціалізованих функціях. Оскільки тип містить спеціальні вказівники на відкриті файли, під’єднання та інше, то будь-які дії над значенням ресурсу не мають сенсу.

Область видимості змінної — середовище, в якій вона визначена. Розрізняють локальні та глобальні змінні. За замовчуванням, всі змінні мають локальний характер дії. Виділяють особливий тип змінних — статичні, що дозволяє повторне звернення до змінної в певному сегменті коду, причому змінна буде зберігати попередньо отримане значення. Існує також поняття суперглобальних змінних, які служать місцем збереження даних оточення або даних, отриманих ззовні. Підтримується концепція динамічних змінних та функцій.

Константи в PHP — ідентифікатори простих значень. Можливе визначення константи, причому після її оголошення стає неможливою зміна її значення чи анулювання. Константи можуть мати лише скалярні значення. Підтримується можливість отримання значення константи за динамічним ім’ям.

Область видимості констант буде глобальною для сценарію та всіх під'єднаних компонентів. Також в ядрі мови визначено чимало системних констант.

Оператори в сенсі мови дозволяють виконувати відповідну дію над одним чи кількома операндами. Оператори бувають трьох типів — унарні, бінарні та тернарні. Оператори, як і в інших мовах характеризуються не лише дією, а й асоціативністю та пріоритетністю. Особливістю булевих операцій порівняння — розрізнення двох класів — з врахуванням типу і без врахування типу, при якому відбувається приведення до відповідного типу. Заокруглення відбуваються завжди в меншу сторону. В мові реалізовані особливі класи операторів — виконання, управління помилками та перевірки приналежності до класу.

Функції в сенсі мови є контейнерами коду, причому можливе поміщення інших функцій та класів. На цьому і базується можливість умовного визначення функції. В цьому випадку висувається вимога попередньої декларації викликаної функції, що не обов’язково в інших випадках. Можливості перевизначення чи деактивації функції не існує. Результат, який повертає функція може мати будь-який тип.

В мові реалізована функціональність посилань. Можливо створити скільки завгодно псевдонімів, що посилаються на єдиний сегмент даних. При вивільненні будь-якого з псевдонімів, сегмент даних залишається в пам’яті до моменту завершення сценарію або вивільнення усіх посилань.

Що стосується функцій в PHP, то замість прийнятого в багатьох мовах принципу перевантаження функцій, що дозволяє змінити хід виконання певної функції в залежності від типу та кількості переданих параметрів, використовується метод динамічних аргументів. Це дає змогу не визначати кількість параметрів для функцій при їх оголошенні, а працювати із тими аргументами, які були отримані на момент виклику функції. У тілі функції можливо отримати кількість переданих їй аргументів і проводити відповідні маніпуляції. При оголошенні функції звичайним чином, можливе задання

значень аргументів за замовчуванням. Функції можуть повертати лише одне значення, проте це обмеження можна оминути, використавши не лише масиви, а й посилання. Передача аргументів за посиланням неможлива під час виконання та оголошення функції.

Після виконання сценаріїв, простір пам’яті, займаної ними очищується збирачем сміття. Проте, за необхідності можливе виконання очищення пам’яті від надлишкових сегментів даних під час виконання скриптів. Використання функцій очищення пам’яті є невиправданим, хоча така можливість існує. Для побудови програмних комплексів можливо використовувати модульний підхід, виконуючи розділення різнорідного коду. При потребі, можливе виконання під’єднання необхідних модулів, причому операція виконання може бути і умовною. Під'єднанні до скрипта файли можуть повертати значення. Мова явно підтримує HTTP cookies відповідно до специфікацій Netscape. Це дозволяє проводити встановлення та читання невеликих сегментів даних на стороні клієнта.

Протокол HTTP, засобами якого, як правило, обмінюються інформацією клієнт та веб-сервер не надає змогу зберегти стан сеансу взаємодії. Це випливає із того, що між клієнтом та сервером не встановлюється постійне з'єднання і клієнт не надає жодних відомостей, що можуть виділити його з поміж інших активних в деякому околі часу. Альтернативою cookies є концепція сесій, яка знайшла свою реалізацію в PHP. В сесії можна зберігати різні дані, включаючи об’єкти.

2.2.3 Обґрунтування вибору системи управління базами даних

База даних – сукупність даних, організованих за визначеними правилами, що передбачає загальні принципи опису, збереження і маніпулювання даними, незалежно від прикладних програм. СКБД – комплекс програм і мовних засобів для створення, ведення і використання БД. Часто для роботи з БД

використовуються не СКБД, а створені з їхньою допомогою інформаційні системи, що забезпечують роботу з інформацією, регламентуючи доступ до структури БД.

СКБД – одні з найбільш розповсюджених програмних продуктів. Вони розрізняються швидкістю обробки даних (виконання запитів, пошуку в таблицях), можливостями збереження різних типів даних, способами підтримки цілісності і несуперечності даних у таблицях. Основна відмінність між СКБД полягає в реалізації моделі даних БД. Найбільш поширені в даний час СКБД dBase, FoxPro, Paradox, Clarion, Access, MySQL що можуть вирішити практично всі задачі користувача.

В даний час існує багато вирішень для розробки баз даних та систем управління ними, також є уже готові для використання СКБД.

Сучасні БД ґрунтуються на використанні моделей даних (МД), що дозволяють описувати об’єкти предметних областей і взаємозв'язки між ними, існують три основні МД і їх комбінації, на яких ґрунтуються БД:

- реляційна модель даних (РМД);
- мережева модель даних (ММД);
- ієрархічна модель даних (ІМД).

В даний час широкого використання набула реляційна модель даних. Реляційною називається база даних, у якій всі дані, доступні користувачу, організовані у вигляді таблиць, а всі операції над даними зводяться до операцій над цими таблицями. Поняття реляційний пов’язано з розробками відомого американського фахівця у області систем баз даних Е.Кодда.

Реляційна модель орієнтована на організацію даних у вигляді двовимірних таблиць. Кожна реляційна таблиця є двовимірним масивом і володіє наступними властивостями:

- кожен елемент таблиці - один елемент даних;

- всі стовпці в таблиці однорідні, тобто всі елементи в стовпці мають однаковий тип (числовий, символьний і т.д.) і довжину;
- кожен стовпець має унікальне ім'я;
- однакові рядки в таблиці відсутні;
- порядок проходження рядків і стовпців може бути довільним.

В даному випадку для реалізації проекту вибрано реляційний підхід.

Значна кількість інформації, можлива віддаленість місця її збереження від джерел інформації – все це зумовлює необхідність розподілу введення даних. Водночас значна кількість користувачів, які територіально розкидані, вимагає багатьох місць доступу до обробленої інформації. Це спричиняє широке використання інформаційних мереж в інформаційних системах (ІС).

Дещо умовно, за принципом використання ІС, їх можна поділити на персональні ІС та ІС для роботи у мережі.

Персональна ІС розрахована на одночасну роботу з даними лише одного користувача. Це не означає, що дані не можуть бути розташовані на деякому сервері у мережі, а лише говорить про те, що кожен набір даних може використовуватись у певний час лише одним користувачем.

Головна відмінність ІС для роботи у мережі – можливість одночасного доступу до даних (введення/перегляд) декількох користувачів або застосувань. Водночас ці застосування можуть бути запущені на одному комп’ютері під багатозадачною ОС. У такій ІС необхідно передбачити та запропонувати вирішення проблеми блокування даних на час їхнього використання. Майже всі сучасні ІС необхідно проектувати для роботи в інформаційних мережах.

MySQL це реляційна база даних, робота з даними в якій здійснюється за допомогою SQL запитів. Основними перевагами цього типу БД є швидкість та простота у використанні. За допомогою MySQL можна робити операції над даними, які з текстовими файлами важко реалізовані. Дана система керування базами даних (СКБД) була створена як альтернатива деяким комерційним

системам, які за функціональністю не дуже поступаються СКБД з відкритим кодом. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL - одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, в веб-рішеннях, оскільки має чудову підтримку з боку різноманітних мов програмування для веб.

Переваги MySQL:

- швидкість(одна з найшвидших СКБД);
- простота у встановленні та використанні;
- безкоштовні засоби адміністрування від виробників.

2.2.3. Обґрунтування вибору операційної системи

Найбільш доцільним методом обґрунтування вибру операційної системи буде приведення порівняльною характеристики Microsoft Windows і GNU/Linux (двох надсімейств операційних систем). Windows 7 – найбільш поширена система являється закритим програмним продуктом, GNU/Linux найпопулярніша операційна система з вільного програмного забезпечення.

Переваги обох систем:

– Швидкі темпи розвитку даних ОС, в настільних комп'ютерах, на серверах, в державних установах і офісах, суперкомп'ютерах і вбудованих системах;

– Windows 7 – домінуюча система на настільних і персональних системах (близько 90 % настільних комп'ютерів), GNU/Linux популярна на веб-серверах, обчислювальних кластерах і в суперкомп'ютерах (50 – 80 %);

– Вони різняться у вартості, простоті використання, зручності і стабільності і з часом удосконалюються в кожній з цих областей. При

їх порівнянні доводиться брати до уваги їх коріння, історичні чинники і спосіб розповсюдження.

Традиційно слабким місцем GNU/Linux вважалося складність в установці, а слабким місцем Windows – недостатня стабільність, але в обох системах ведеться активна робота над поліпшенням. Сильною стороною GNU/Linux вважається пошана до свобод користувача: можливість запускати на будь-яких системах, досліджувати і змінювати початковий код, поширювати початкові або змінені версії. Колективна розробка цієї операційної системи зробила її надзвичайно відкритою і такою, що настроюється, при цьому підтримуючи міжнародні стандарти ISO і IEEE, тоді як сильні позиції Windows – заявлена чуйність до запитів так званого «середнього користувача» і встановлення власних стандартів в IT-області завдяки своєму домінуючому положенню.

Windows і Linux важко порівнювати на рівних в зв’язку з наступними чинниками:

- обидві системи поставляються в різних конфігураціях. Особливо GNU/Linux, для якої існує величезна кількість варіантів, деякі з них призначені для вузького кола завдань;
- ціна і широта технічної підтримки розрізняються у різних постачальників, а також залежно від версії і дистрибутива;
- виробники устаткування можуть встановлювати додаткове ПО з операційною системою, яке робить доступні функції системи різноманітнішими. Іноді вони навіть спонсорують продавця, знижуючи ціну продукту для користувача;
- дані, отримані від маркетингових підрозділів і результати тестування можуть розходитися;

– Microsoft поширює Windows під різними ліцензіями (як правило, закритими, але у виняткових випадках з відкритим початковим кодом). Дистрибутиви Linux, зі свого боку, можуть містити платні компоненти.

Порівнюючи Windows та Linux не можна однозначно сказати, котра операційна система краща. Для нормального функціонування веб-сайту потрібно: веб-сервер Apache, логіку динамічних Інтернет сторінок PHP та систему керування базами даних MySql. Всі перечисленні вище компоненти інсталюються як на Windows так і на Linux системах. З цього випливає висновок, що веб-сайт може виконуватися на будь-якій операційній системі (тобто в залежності від операційної системи хостинг провайдера, будь то Windows чи Linux).

Те ж саме можна сказати і про користувацьку операційну систему. Для перегляду веб-сайту потрібно мати в наявності браузер (Opera, Internet Exprorer, Maxtron, Mozilla Firefox та ін.), котрий також є в наявності у всіх операційних системах сімейства Windows та Linux.

РОЗДІЛ 3

ПРОЕКТУВАННЯ СИСТЕМИ АНАЛІЗУ ЕМОТИКОНІВ

3.1 Розробка алгоритму детектування емотиконів

3.1.1 Опис процесу проведення індукції граматики

Розроблювана система виконує індукцію СКВ (Стохастична контекстно-вільна граматика) граматики окремо в кожну послідовність з можливими емотиконами, базуючись на простому наборі методів шаблонів правил для присвоєння значимості правил. Індукуючи невеликі, окремі для кожного прикладу СКВ граматики, переконуємося, що кожний приклад має правильний розбір без збільшення граматики до розмірів котрі будуть впливати на ефективність аналізатора.

В розроблюваній системі детектування смайлів використовується система котра за своїми характеристиками схожа на просту Марківську модель, котра ділить стрічку графем Юнікоду на суміжні ділянки, котрі в основному бувають цілі емотікони, або такі що складаються з декількох частин (знаки пунктуації та інші символи).

Підхід забезпечує високу чутливість і повертає майже всі речення у котрих є емотікони, але має низьку точність тому що може прийняти за смайл ділянки з великою кількістю символів пунктуації. Проста ПММ (Прихована марківська модель) складається з двох станів: A (головним чином лінгвістичний) та @ (головним чином нелінгвістичний). Зважаючи на те, що існує два класи результуючих символів ПММ повинен мати дві можливості результату, одну для їхніх основних класів символів(лінгвістичних L і нелінгвістичних N) і одну для іншого класу символів.

Нелінгвістичні символи часто з’являються в лінгвістичних послідовностях. Однак, послідовність з трьох таких символів зустрічається

рідко, лінгвістичні символи також часто зустрічаються в емотиконах, але кількістю не більшою за три. Так як, для сегмента в суміжних послідовностях з певною кількістю символів в ряд, можливість переходу зі стану A в стан @ має бути набагато нижча чим можливість результату з однієї чи двох N зі станів A чи L зі станів @. Таким чином ми отримуємо ПММ з вісьмома параметрами (чотири для переходу і чотири для результату), котра була грубо параметризована, щоб мати властивості перераховані вище. Потрібно звернути увагу, на те, що такий підхід має обмеження, яке полягає в тому, що будуть обрізатися певні лінгвістичні символи котрі будуть розміщуватись на периферії емотикону.

3.1.2 Формальне подання основ роботи алгоритму

Сформована СКВ буде містити нетермінальний X , та змінні a та b будуть вибрані з вхідної послідовності . Після вибору першої змінної вхідних даних є можливість вибору елементу котрий репрезентує середній сегмент, тобто «ніс». Тобто, можна зазначити, що СКВ містить два правила, виконання першого з них є необов'язковим. Отже визначення елементу терміналу для участі в СКВ відбувається тільки збігом його з великим набором терміналів. Це дозволяє виконувати формування емотиконів, як от таких, що складаються тільки з двох нетерміналів, а також таких котрі являються одним не терміналом. Всі можливі комбінації емотиконів виявляються за таким шаблоном.

Таблиця 3.1

Схема правил для СКВ

Правила	Результат
X→a	Повний емотикони
X→aX	Емотикон з двох символів
X→abX	Емотикон з трьох символів

Описана базова індукція граматки може бути покращена кількома способами не жертвуючи надійністю цього методу. Один з них це розбиття вхідних символів на окремі слова. Другий – збільшення кількості символів нетерміналів у граматиці.

3.2 Розробка програми детектування емотиконів

3.2.1 Постановка задачі на розробку програмного забезпечення

В результаті виконання магістерської роботи було створено до можливостей котрого входять виконання детектування та аналіз символів емоційного наповнення. Розробка програма була виконана в середовищі Oracle NetBeans.

В процесі наукової роботи над предметом даної магістерської роботи планується розробка програмного продукту котра складається з таких кроків:

- вибір мови програмування для реалізації програмного забезпечення;
- вибір додаткових технологій, які будуть використовуватися;
- проектування бази даних;
- створення структури бази даних;
- створення структури програмного продукту;
- створення алгоритмів роботи програмного продукту;
- написання кодів програмного забезпечення;
- тестування створеного програмного забезпечення;
- створення допоміжної документації для користувачів та обслуговуючого персоналу.

Програмування — процес створення комп'ютерних програм та/або програмного забезпечення. Програмування поєднує в собі елементи інженерії, фундаментальних наук і мистецтва.

У більш вузькому значенні програмування розглядається як кодування — реалізація у вигляді програми одного чи кількох взаємопов'язаних алгоритмів (у сучасних умовах це здійснюється з застосуванням мов програмування). У ширшому сенсі процес програмування охоплює і створення, тобто розробку, алгоритмів, і аналіз потреб майбутніх користувачів програмного забезпечення.

У широкому значенні програмування використовується у значенні створення алгоритмів та навчання людей або пристроїв діяти за алгоритмами.

Алгоритм — система правил виконання обчислювального процесу, що обов'язково приводить до розв'язання певного класу задач після скінченного числа операцій. При написанні комп'ютерних програм алгоритм описує логічну послідовність операцій. Кожен алгоритм є списком добре визначених інструкцій для розв'язання задачі. Починаючи з початкового стану, інструкції алгоритму описують процес обчислення, які відбуваються через послідовність станів, які, зрештою, завершуються кінцевим станом. Перехід з одного стану до наступного не обов'язково детермінований — деякі алгоритми містять елементи випадковості. Наприклад алгоритми нечіткої логіки котрі використовуються при виконанні даного дослідження.

3.2.2. Визначення інформаційних зв’язків програмних компонентів

Робоче поле програми розбите на декілька блоків котрі виконують певні функціональні призначення. В інтерфейсі присутня кнопка за допомогою котрої здійснюється початок операції детектування емотикони в стрічці.

Програма забезпечує самі прості засоби ітерації з користувачем, а саме введення даних у текстові поля і отримання значень у вигляді тексту. Такий

підхід дозволяє забезпечити легкість виконання задачі розробки системи детектування та аналізу

Для виконання даного сайту була використана мова серверних скриптів PHP і мову запитів Mysql.

MySQL – вільна система керування реляційними базами даних.

Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

У процесі виконання дослідження були використані реляційні таблиці бази даних. Схема організації таблиць у базі даних можна зобразити у вигляді діаграми (див. рис. 3.1).

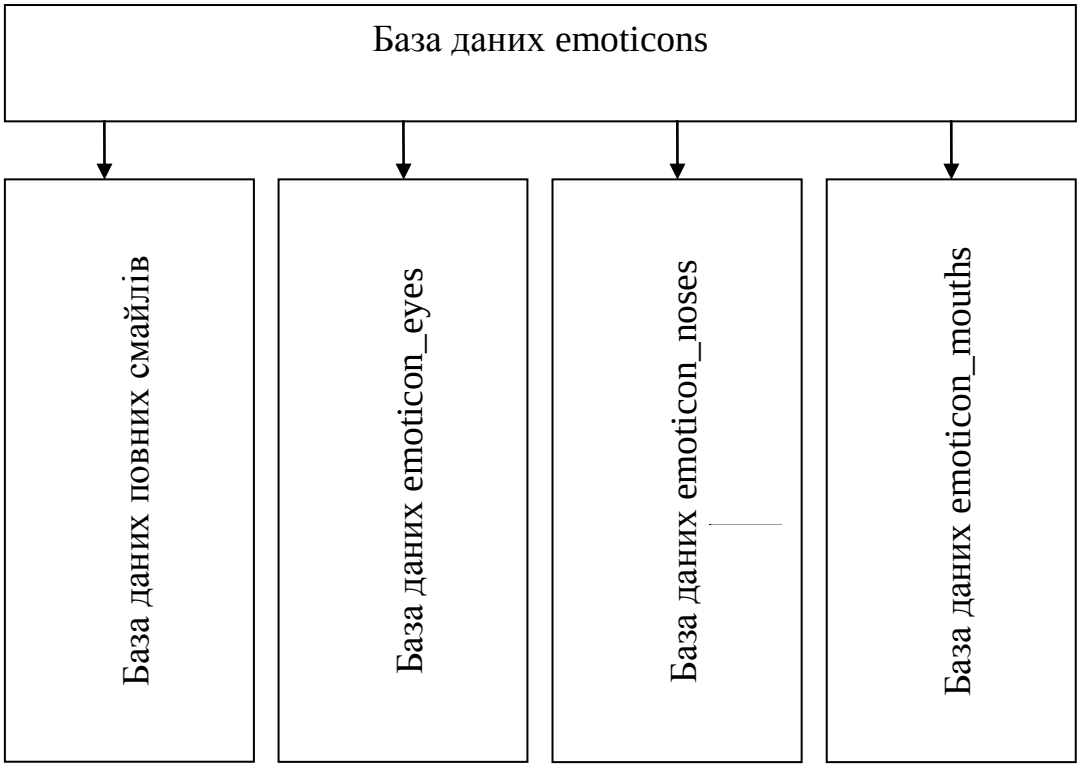

Рисунок 3.1 – Загальна схема організації таблиць в базі даних

В якості візуального оформлення вибрано чіткий дизайн з використанням блоків.

При роботі з частиною сайту котра передбачає використання тільки скриптів PHP. користувач має змогу бачити сторінку, що складається з блоків та клітчатого тла (див. рис. 3.2).

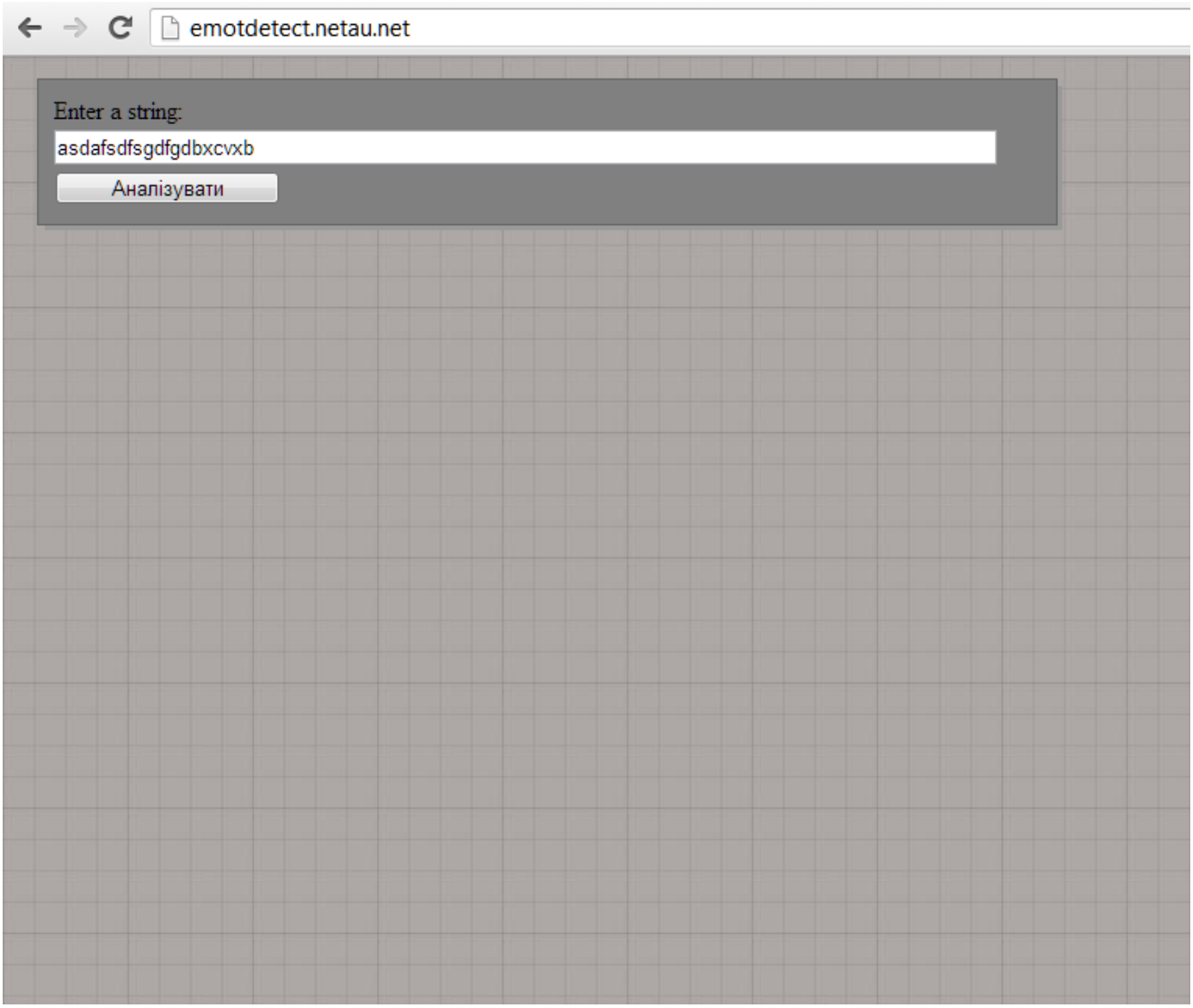

Рисунок 3.2 – Інтерфейс програми

Видача фінальних результатів здійснюється у вигляді обрахованих емоцій виражених смайлами (див. рис. 3.3).

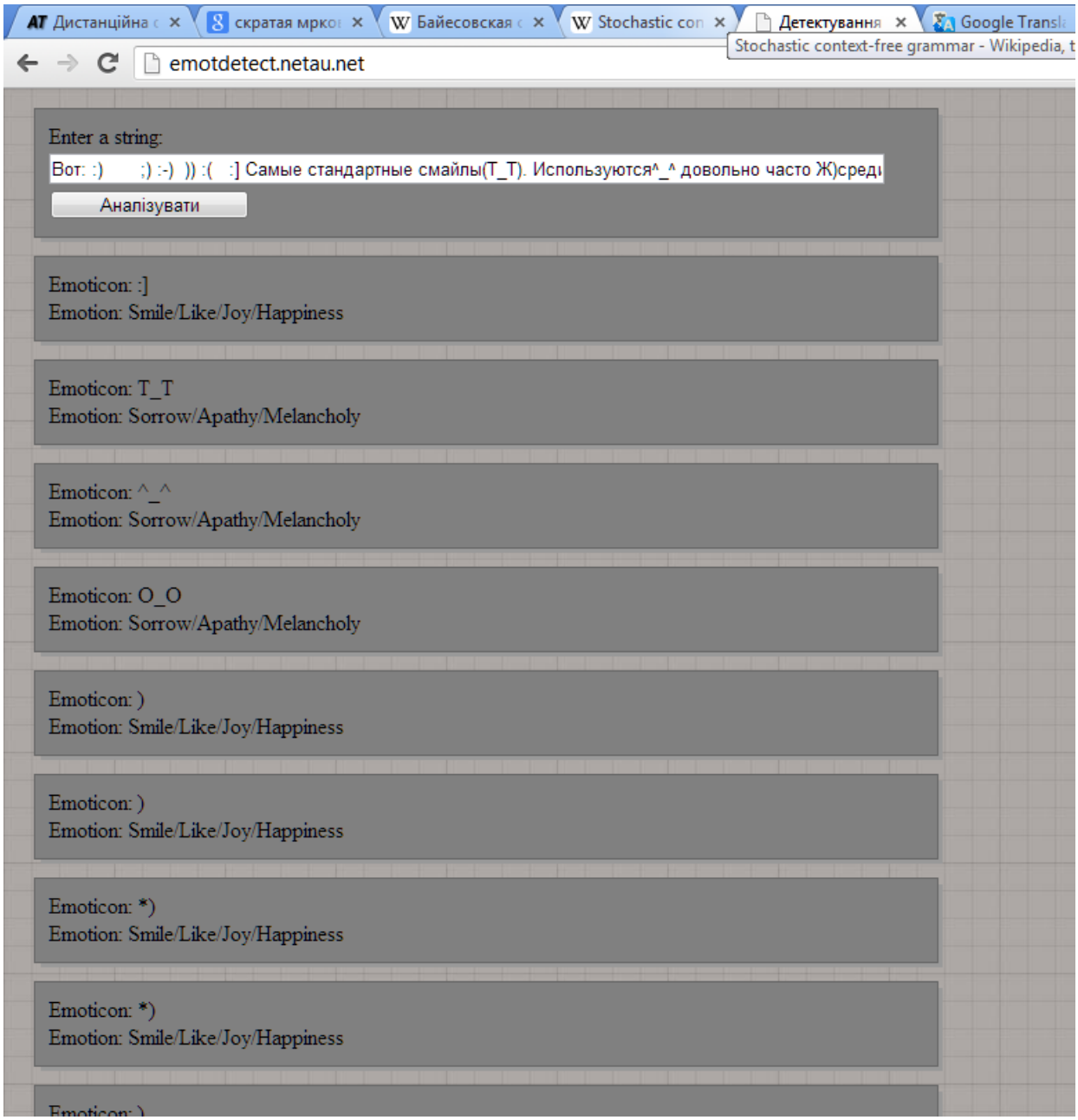

Рисунок 3.3 – Видача результатів

3.2.3. Визначення функції ініціалізації

На початку роботи програми відбувається зчитування змінною $_POST['input_string'] котра репрезентує собою текстове поле інтерфейсу програми. Відповідно до значення змінною відпивається вивід відповідного тексту на екран.

Після виконання вищезгаданого відбувається виклик функції котра видаляє з вхідних даних симетричні символи на зразок дужок (див. рис 3.4).

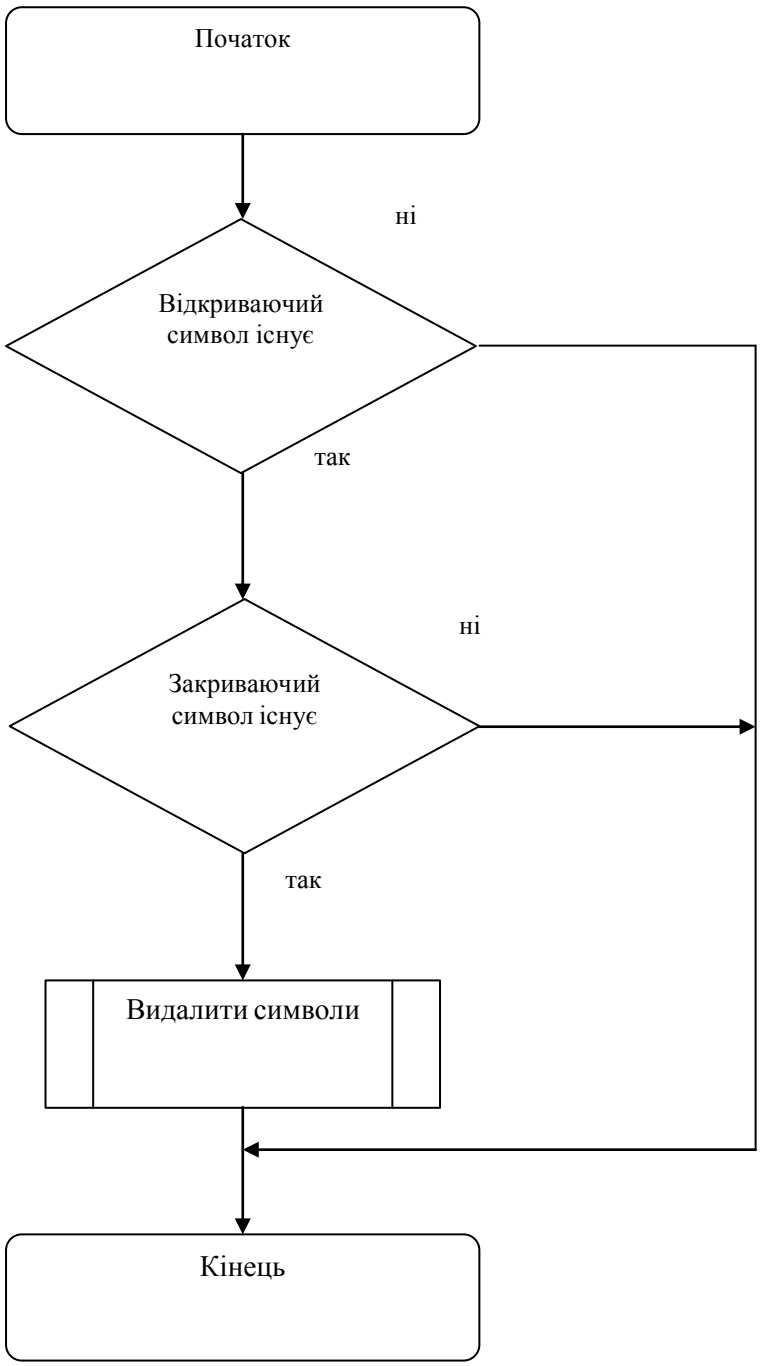

Рисунок 3.4 – Робота функції deleteSymetrical()

3.2.4 Написання текстів програми

В першому сегменті коду програми виконується отримання початкових даних із поля input. Форма для введення інформації виглядає наступним чином:

```
<div id="enter_string" class="border sub1">
    Enter a string: </br>
<form id="form1" name="form1" method="post"  >
    <input name="input_string" type="text" id="name"
    <?php
    if (isset($_POST['input_string'])) {
        echo 'value="' . $_POST['input_string'] . '"';
    }
    ?> />
    </br>
    <input type="submit" name="submit" id="submit" value="Аналізувати" />
</form>
</div>
```

Подальшим кроком є здійснення зчитування даних з форми розбиття на слова за допомогою спеціальних функцій та передачі в якості аргументу в функцію котра буде детектувати емотикони присутні в цьому слові. Для здійснення такої операції використовується оператор foreach котрий слугує заміною оператору for і здійснює перебір членів масиву без використання додаткової змінною-лічильника. Код для вищезгаданих операцій виглядає наступним чином:

```
if (isset($_POST['input_string'])) {
        $emoticon = $_POST['input_string'];
        $emoticon = deleteSymetrical($emoticon, "(", ")");
```

```
$mas = explode(" ", $emoticon);
$emoticons = array();

foreach ($mas as $string) {
   $temp = checkWholeEmoticon($string);
   foreach ($temp as $mas2) {
      $emoticons[] = $mas2;
   }
}
```

Всі необхідні операції по детектуванню емотиконів здійснюються з використанням функцій checkWholeEmoticon() та checkEmoticonbyParts(), котрі в якості параметрів приймають слово з можливим емотіконом у ньому.

Для виводу даних про наявність емотикони та даних про емоційну складову емотикона використовується наступна конструкція:

```
foreach ($emoticons as $mas){
      echo "<div class='border'>";
      echo "Emoticon: ".$mas[1]."</br>";
      switch ($mas[2]) {
         case "1":
            echo "Emotion: Anger</br>";

            break;
         case "2":
            echo "Emotion: Like/Joy</br>";

            break;
         case "3":
            echo "Emotion: Sorrow</br>";
```

```
            break;
        case "4":
            echo "Emotion: Dislike</br>";

            break;
        default:
            echo "NO EMOTION FOUND";
            break;
    }
    echo "</div>";
}
```

В даній конструкції використовується оператор switch для підбору текстової характеристики емотикона відповідно до цифрового його позначення взятого із бази даних.

3.2.5 Інструкція з користування тестовими наборами

Призначенням даної програми є детектування емотиконів. Програма виконана у вигляді Інтернет сайту і дозволяє виконати детектування емотикону зручно з будь-якого комп’ютера.

Програма володіє інтерфейсом в котрому блоки котрі потребують введення даних виділені кольором та межею , що дозволяє користувачу зорієнтуватися в інтерфейсі відразу після запуску сайту.

Дана програма виконує такі функції:

1) дозволяє ввід даних користувачем;
2) вивід фінальних розрахунків.

РОЗДІЛ 4
ДОСЛІДНИЦЬКИЙ РОЗДІЛ

4.1. Мета проведення досліджень

Метою наукового дослідження є створення математичної моделі, алгоритму, а також програми і бази даних знань для розв’язання проблеми розпізнавання графічних елементів формованих в тексті.

Науковою новизною даного дослідження являється використання марківської моделі у системах детектування емоційного наповнення повідомлення.

Система та алгоритми котрі розробляються в процесі наукового дослідження можна використати інфраструктурі систем лексичного аналізу.

4.2. Математична обробка експериментальних даних

В результаті роботи розроблюваної системи отримуємо статистичний ряд нерівноточних вимірів кількості віднайдених символів емоційного наповнення в тексті:

15,16,14.

Знаходимо ваги вимірів за формулою:

$$p_i = \frac{n_i}{N_i},$$

де N_i – кількість виміряних величин;

n - кількість вимірів (прийомів) однієї шуканої величини.

$$p_1 = \frac{3}{1}.$$

Вираховуємо загальне арифметичне за формулою:

$$\bar{X} = \frac{[xp]}{[p]}$$

$$\bar{X} = \frac{[15,16,14]}{3} = 15$$

Оскільки випадкова похибка змінюється під час проведення повторних вимірювань навіть за однакових умов, то це призводить до мінливості результатів вимірювань. Таким чином, випадкова похибка може бути виявлена на основі аналізу результатів повторних вимірювань, проведених за умов збіжності (за одних і тих же умов).

Непрогнозований характер мінливості випадкових похибок призводить до практичної неможливості коригування результатів вимірювань, тобто до внесення поправок на величину випадкової похибки, оскільки для конкретного результату її значення залишається невідомим. Тим не менше, часто існує необхідність зменшити випадкову похибку результату вимірювання, наприклад, якщо вона перевищує прийнятне значення. Для цього можуть бути використані такі методи:

1. Усунення причин виникнення випадкових похибок.

Наприклад, якщо відомо, що джерелом істотної випадкової похибки є випадкові коливання напруги в мережі живлення приладу, можна його живлення здійснювати через стабілізатор напруги. Зрозуміло, що усунення всіх ефектів, які призводять до появи випадкової похибки неможливе або через фізичні причини, або через економічні причини.

2. Математична обробка результатів повторних вимірювань, спрямована на зменшення випадкової похибки.

Усереднення результатів повторних вимірювань завдяки другій властивості випадкових похибок дозволяє зменшити випадкову похибку кінцевого результату вимірювання. В ідеалі, за нескінченно великої кількості

вимірювань, усереднення дозволило б звести випадкову похибку до нуля, оскільки однакові за модулем, але протилежні за знаком випадкові похибки, які за другою властивістю зустрічаються однаково часто, при усередненні себе повністю б компенсували. На практиці, звичайно, через обмежену кількість результатів вимірювань повної компенсації добитися не вдається (слід зазначити, що і практичної потреби в цьому немає), однак все ж випадкова похибка середнього значення зменшується зі зростанням кількості результатів повторних спостережень.

Вибір методу зменшення випадкової похибки результату вимірювання залежить як від можливості впливу на ті чи інші джерела виникнення випадкових похибок, так і від економічних чинників, оскільки і заходи з усунення причин, і проведення повторних вимірювань вимагають додаткових затрат часу та ресурсів.

Абсолютні похибки знаходимо за наступною наступним чином:

$$\Delta_i = x_i - X;$$
$$\Delta_1 = 16 - 14 = 2;$$
$$\Delta_2 = 15 - 14 = 1;$$
$$\Delta_3 = 14 - 14 = 0.$$

Систематичну похибку λ, при відомому істинному значенні X=14 знаходимо за формулою:

$$\lambda = \bar{X} - X;$$
$$\lambda = 15 - 14 = 1.$$

Середню квадратичну похибку одиниці ваги знаходимо за формулою:

$$\mu = \sqrt{\frac{[p\Delta^2]}{n}};$$
$$\mu = \sqrt{\frac{[3 \cdot 1]}{3}} = 1.$$

Середню похибку загального середнього арифметичного знаходимо за формулою:

$$M = m_{\bar{X}} = \frac{\mu}{\sqrt{[p]}};$$

$$M = m_{\bar{X}} = \frac{1}{\sqrt{1}} = 1.$$

Середню квадратичну похибку середньоквадратичної похибки ваг знаходимо за формулою:

$$m_{\mu} = \sqrt{\frac{\mu}{2(n-1)}};$$

$$m_{\mu} = \sqrt{\frac{1}{2}} = 0{,}7.$$

Середню квадратичну похибку середньої квадратичної похибки загального середнього арифметичного обраховуємо за формулою:

$$m_{M} = \frac{\mu}{\sqrt{2[p](n-1)}}$$

$$m_{M} = \frac{1}{\sqrt{12}} = 0{,}28$$

4.3 Верифікація математичної моделі

В якості параметрів скористаємось кількістю емотиконів в тексті, номером емоції відображуваної ним та кількість символів котрі входять у нього. Для визначення відносної похибки скористаємося формулою:

$$\xi_j = \left(\tilde{y}_j - y_j\right) / y_j,$$

де $\tilde{y}_j$ - j-й вихідний параметр, розрахований за допомогою моделі;

y_j - той же вихідний параметр, що має місце в об'єкті, який моделюється.

$$\xi_1 = (20-19)/19 = 0{,}05;$$

$$\xi_2 = (6-4)/4 = 0.50;$$

$$\xi_3 = (4-3)/3 = 0{,}33.$$

Похибка обчислень $\boldsymbol{\xi_M}$ за сукупністю вихідних параметрів визначається за формулою:

$$\xi_M = \sqrt{\sum_{j=1}^{M} \xi_j^2};$$

$$\xi_M = 0{,}36.$$

Отже $\xi_M < \xi_{пред}$, тому модель можна вважати адекватною а вихідні параметри верифіковані лежать в *області адекватності*.

5 ТЕХНІЧНИЙ РОЗДІЛ

5.1 Реалізація високої доступності розроблюваної системи

5.1.1 Робота з Amazon Elastic Compute Cloud

Для реалізації високої доступності будемо використовувати Amazon Web Services. Наш додаток буде складатись з двох серверів, які будуть обслуговувати запити і балансера навантаження, який буде перенаправляти запити на другий сервер, якщо перший недоступний. Базовим сервером проекту буде Amazon Elastic Compute Cloud.

Amazon Elastic Compute Cloud (Amazon EC2) — веб-сервіс, котрий надає обчислювальні потужності в хмарі. Сервіс входить в інфраструктуру Amazon Web Services.

Простий веб-інтерфейс сервісу дозволяє отримати доступ до обчислювальних потужностей і налаштувати ресурси з мінімальними затратами. Від надає користувачам повний контроль над обчислювальними ресурсами, а також доступне середовище для роботи. Сервіс скорочує час, необхідний для отримання і завантаження нового сервера.

За допомогою EC2 дозволяється:

- створювати Amazon Machine Image (AMI), котрий буде утримувати ваші програми, бібліотеки, дані та пов'язані з ними функціональні параметри або використовувати раніше налаштовані шаблони образів та робіт;
- завантажити AMI в Amazon S3. Amazon EC2 пропонує інструменти, для зберігання AMI. Amazon S3 забезпечує захищене, надійне та швидке сховище для зберігання образів;
- використовувати Amazon EC2 Веб-сервис для налаштування безпеки та мережевого доступу;

- можливість вибору типу операційної системи, за вашими потребами, запуск, закінчення або контроль декількох AMI по мірі необхідності, використовується API веб-сервісу, або різних інструментів управління, які раніше були передбачені;
- визначити необхідність працювати в декількох місцях, використовуючи статичні IP-адреси або інші варіанти;
- оплачувати тільки за ресурси, котрі ви збираєтесь використовувати, такі як час або кількість переданої інформації (кількість трафіку).

Для початку запустимо основні сервери проекту. Для цього після реєстрації в Amazon Web Services перейдемо на сторінку налаштувань EC2 серверів і запустимо сервер з ОС Linux (рисунок 5.1).

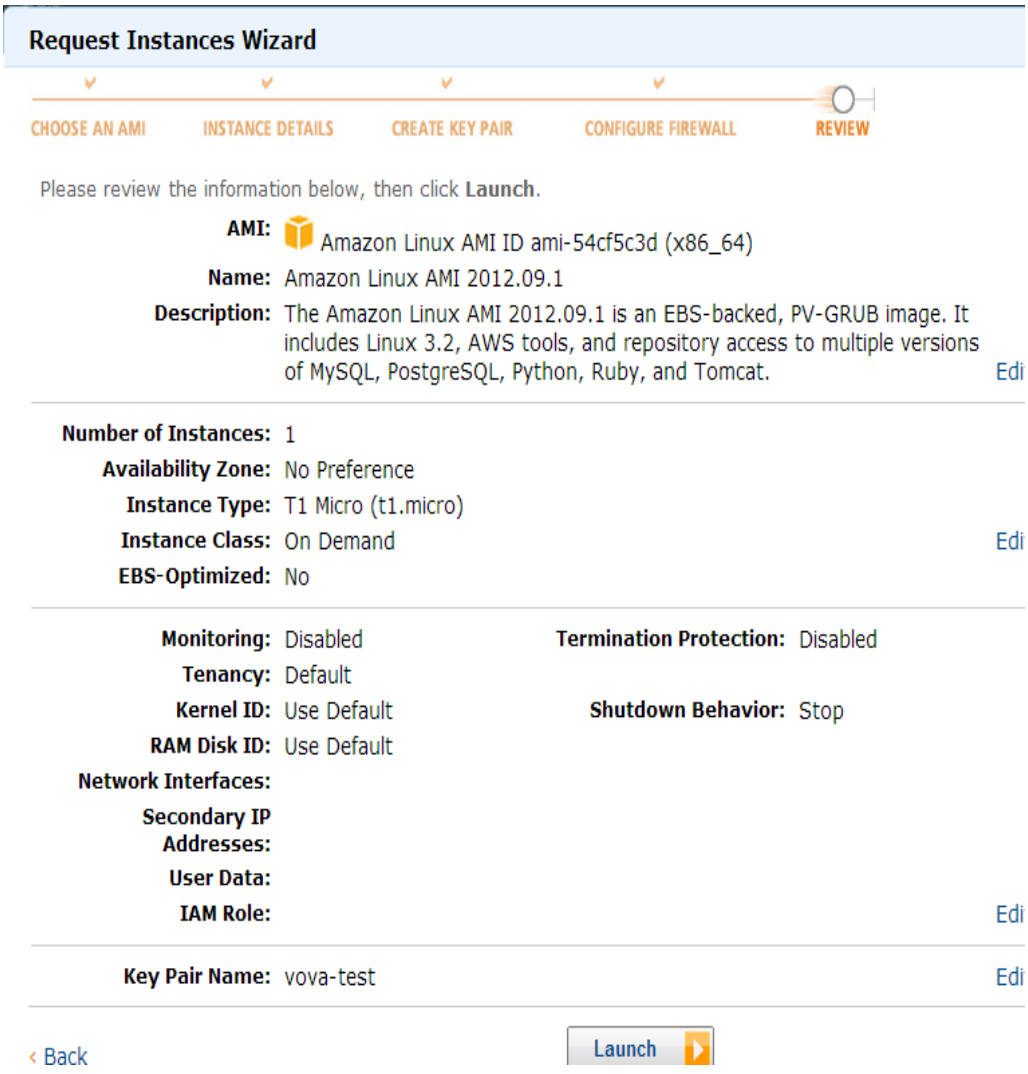

Рисунок 5.1 – Запуск EC2 сервера в Amazon Web Services

Після хвилини очікування, ми отримаємо зображення, показане на рисунку 5.2, де ми можемо бачити, що в нас є 2 сервери, які функціонують.

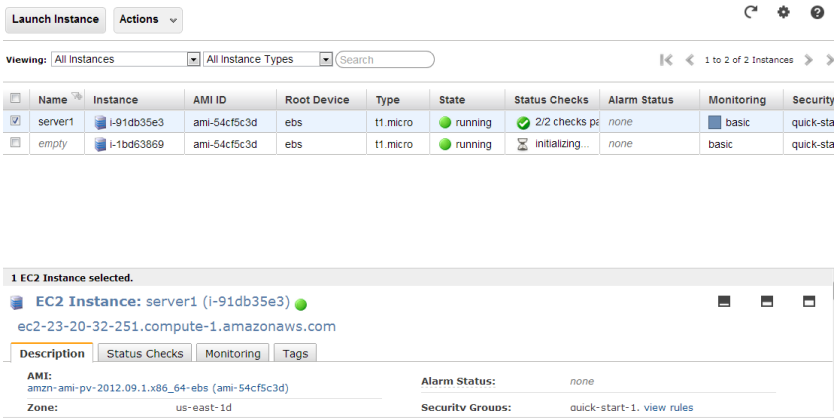

Рисунок 5.2 – Перегляд існуючих EC2 серверів проекту

Для можливості перевірки роботи сервера, створимо файл index.html в який помістимо контент «Hello, World! This is server #1» для першого сервера і «Hello, World! This is server #2» для другого. Створені файли помістимо в кореневу директорію веб-сервера. При звертанні до IP-адреси відповідного сервера через веб-браузер, ми бачимо відповідний контент.

Наступним кроком є налаштування балансувальника навантаження.

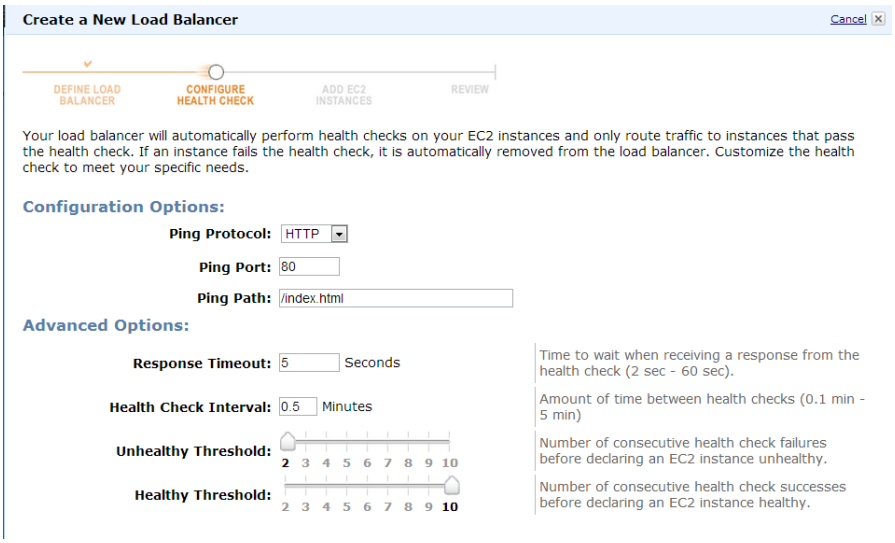

Рисунок 5.3 – Створення балансувальника навантаження

Як ми бачимо на рисунку 5.3, ми можемо виставляти різні параметри. Давайте встановимо переадресацію на інший сервер при відсутності відповіді 5 секунд. Період перевірки – 0.5 секунд. Після цього вся система є налаштована.

Load Balancer дав нам IP адресу http://my-910462096.us-east-1.elb.amazonaws.com/, результат звертання до неї зображено на рисунку 5.4

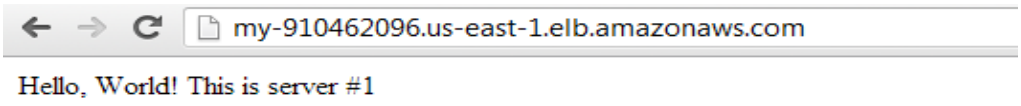

Рисунок 5.4 – Звертання до налаштованої системи по HTTP протоколу

Тепер необхідно протестувати чи справляється балансувальник з поставленими задачами. Для цього зупиняємо apache сервер на першому хості і знов звертаємся до IP-адреси нашого проекту. (рисунок 2.5)

Рисунок 5.5 – Перевірка роботи системи при зупиненому основному сервері

Отже, в результаті ми отримали систему високої готовності. При несправності одного з серверів, користувач буде автоматично перенаправлений на працюючий сервер. В результаті в системі не буде присутній час простою.

Тепер ми можемо сміливо розгорнути наш проект на даних серверах. Високу доступність досягнуто.

5.1.2 Конфігурування веб-сервера Apache

На кожному з запущених ec2-серверів проекту нам необхідно налаштувати веб-сервер Apache.

Apache установлює свої файли в наступні каталоги:

- /etc/httpd/conf – цей каталог містить всі файли конфігурації сервера Apache;
- /etc/red/ – у цьому каталозі знаходяться інші каталоги, які містять сценарії запуску системи. Apache установлює їхній повний набір для Web-сервера. Ці сценарії можуть бути використані для запуску й зупинки сервера у ручному режимі, а також будуть автоматично запускати й зупиняти сервер, коли робоча станція зупинена, запускається або перезавантажується;
- /home/httpd – у цей каталог встановлений стандартний набір піктограм для сервера й CGI-сценаріїв;
- /usr/doc й /usr/man – у цих каталогах містяться керівництво manual pages і файлиreadme. Більшість пакетів розміщає файли readme й іншу аналогічну документацію в спеціально названому для версії сервера каталозі /usr/doc;
- /usr/sbin – у цей каталог розміщено всі виконувані програми.

Apache зчитує конфігураційні установки із трьох файлів: access.conf, httpd.conf й srm.conf. Спочатку така організація була розроблена для підтримки зворотної сумісності із сервером NCSA. Файли конфігурації знаходяться у підкаталозі conf сервера. Резервні копії цих файлів також включено в програмний дистрибутив. Вони відповідно названі access.conf-dist, httpd.conf-dist й srm.conf-dist.

Налаштування роботи сервера відбувається за допомогою конфігураційних директив. Директиви – це спеціальні команди, що встановлюють деякі опції. Ними потрібно скористатися для включення таких необхідних опцій роботи сервера, як розміщення файлів, важливих для

конфігурації й роботи сервера. Конфігураційні директиви мають наступний синтаксис: директива опція.

Директиви розміщуються по одній у рядку. Одні директиви встановлюють тільки значення імені файлу, інші дають можливість визначити кілька опцій. Деякі спеціальні директиви, названо секціями (section) і вони дуже схожі на теги HTML. Секції розміщено в кутових дужках, наприклад: <директива>. Після секції звичайно йде група директив, дія яких поширюється на зміст каталогу, відповідного до секції:

```
<Деякий_каталог_нашого_дерева_каталогів>
директива опція опція
директива опція опція
</директива>
```

Всі секції закриваються спеціальним секційним вираженням, що має вигляд </директива>. Ми побачимо деякі із цих конструкцій у файлах conf/access.conf й conf/httpd.conf. Помітьте, що спеціальні секційні вираження схожі на інші директиви й визначаються по одному в рядку.

Файл httpd.conf містить конфігураційні директиви, що управляють роботою сервера та встановлюють місце розташування файлів реєстрації, ID(UID) користувача, під яким він працює, номер використовуваного порту й багато чого іншого. Необхідно відредагувати значення початкових установок, що важливі для конфігурації вашого сайту. Звичайно залишають без зміни більшість установок, за винятком перерахованих нижче.

Директиву ServerAdmin варто настроїти на адресу Web-майстра, що керує сервером. Це повинна бути діюча адреса e-mail або псевдонім, наприклад: webmaster@emodetect.net. User й Group. Ці директиви встановлюють UID і груповий ID(GID), які будуть використані сервером при формуванні запитів. Краще застосовувати параметри, установлені початково, тобто: nobody та nogroup. Переконайтеся, що імена nobody й nogroup існують відповідно у ваших файлах /etc/passwd і /etc/group. Повноваження для цих UID й GID повинні бути обмежені, тому що у випадку виникнення «дір системного захисту», ці програми

будуть продовжувати виконуватися із призначеним UID. Якщо сервер запущений у режимі супер-користувача, хтось може скористатися виниклими «дірами в системному захисті».

ServerName встановлює ім'я вузла, що сервер буде повертати. Встановіть повне ім'я домену. Якщо це значення не визначене, сервер спробує визначити його сам й установить директиву ServerName з використанням імені в канонічному виді. Директива ServerName повинна бути дійсним ім'ям, визначеним службою Domain Name System (DNS) для вашої мережі. Якщо хтось інший управляє DNS, то попросіть його встановити це ім'я.

Директива ServerRoot встановлює абсолютний шлях до каталогу сервера. Вона вказує серверу, де шукати всі файли ресурсів і конфігурації. Більшість цих ресурсів визначено у файлі конфігурації, що відносяться до каталогу ServerRoot. Ваша директива ServerRoot може виглядати так: ServerRoot /etc/httpd.

Файл srm.conf – це файл конфігурації ресурсів. Його керуючі установки віднесено до місця розташування вашого дерева Web-документів, до каталогів CGI-програм і до проблем конфігурації інших ресурсів, які мають відношення до вашого Web-сайту. Звичайно всі установки у файлі srm.conf залишають такими, якими їх завдано на початок. Найбільш важливими директивами в цьому файлі конфігурації є наступні:

DocumentRoot - встановлює абсолютний шлях до дерева документів. Ваше дерево документів – це головний каталог, з якого Apache буде обслуговувати файли. За умовчуванням він установлений як /home/httpd/html.

UserDir - визначає вихідний каталог локального користувача, де він буде поміщати загальні HTML-документи. Тобто, кожен користувач буде мати власний HTML-каталог. Стандартне значення для цієї директиви – public_html, так що кожен користувач може створити каталог з ім'ям public_html у вихідному каталозі, і HTML-документи, розміщені в цьому каталозі, будуть доступні за адресою http:// ім'я_сервера/~username, де username – зареєстроване ім'я звичайного користувача.

access.conf - глобальний файл керування доступом. У ньому задається тип доступу користувачів до вашого сайту й документів, якими ми наповнюємо його, визначаючи рівень системного захисту й іноді дозволяючи іншим користувачам змінювати деякі параметри системного захисту. За умовчуванням, система встановлює необмежений доступ до документів у вашому DocumentRoot. Але рекомендується залишити всі установки так, як вони задані за умовчуванням.

Якщо хочете обмежити доступ до сайту, то варто переконатися, що всі секції, що вказують «Шлях_до_каталогу», відповідають обговореним при інсталяції каталогам. У секціях Directory на кожен каталог визначається ряд опцій, що звичайно залежать від вимог системного захисту. Зокрема можливо видалити опцію Indexes, яка йде за директивою Options у секції, що має вид:

```
<Directory /home/httpd/cgi-bin>
Options Indexes FollowSymLinks
```

Опції, які встановлюються в глобальних файлах конфігурації, можуть бути скасовані за допомогою файлу.htaccess. Файли.htaccess дають можливість установити директиви сервера на кожен каталог. Ця можливість зручна для каталогів, звідки користувач не повинен мати доступу до головних файлів конфігурації сервера. Можна заблокувати всі можливості.htaccess по заміні ваших опцій, якщо задати директиві AllowOverride значення None, як показано нижче. За умовчуванням ця директива дозволяє всі можливості.htaccess: AllowOverride None.

WEB-сервер Apache працює або в автономному режимі, або в режимі демона. Робота сервера залежить від значення директиви ServerType у файлі conf/httpd.conf. Автономний сервер краще справляється зі своєю роботою, чим сервери, на яких виконується inetd, тому що його процес звичайно завжди готовий обслужити запит. Працюючи як inetd (Internet-демон), новий сервер запускається щораз, коли приходить запит на HTTP-порт. Значна кількість

накладних витрат пов'язане із запуском нового сервера для кожного нового запиту.

За умовчуванням директива ServerType визначає автономний тип сервера. Якщо пропускна здатність сайту досить низька, варто дотримуватися автономного режиму роботи. Щоб запустити сервер inetd, потрібно модифікувати файл conf/httpd.conf і змінити значення директиви ServerType з автономного типу на inetd, як показано нижче:

```
ServerType inetd
```

Директива Port не викликає ніякого ефекту на сервері inetd, оскільки inetd зв'язує порт із програмним забезпеченням, ця установка не має значення для конфігурації inetd. Автономний сервер використає цю конфігураційну інформацію, щоб знати, від якого порту потрібно чекати сигналу.

5.1.3. Налаштування MySQL-реплікації типу Master – Master

Будемо вважати, що сервера мають такі IP адреси:

server1 - 192.168.1.23

server2 - 192.168.1.24

Редагуємо /etc/my.cnf сервера server1. Додаємо до вже існуючих параметрів такі:

```
[mysqld]
# Номер сервера, що реплікується. У кожного має бути унікальний id.
server-id = 1

# Конфігурація сервера, як майстра. Указуємо, де будуть зберігатись бінлоги.
# Бінлог - це список SQL-запитів (окрім SELECT та SHOW запитів ), що були виконані на сервері задля подальшої їх передачі на слейв.
log-bin = /var/lib/mysql/mysql-bin

# Конфігурація сервера, як слейва. Relay-log - лог дій, котрі були виконані на слейві за ініціативи майстра.
relay-log = /var/lib/mysql/mysql-relay-bin
relay-log-index = /var/lib/mysql/mysql-relay-bin.index
#База, що буде реплікуватись.
replicate-do-db = dbwordpress
```

Для сервера server2 необхідно прописати ідентичні параметри, окрім значення 'server-id' (замінюємо на цифру 2).

Рестартуємо mysql на кожному сервері.

Наступне, що необхідно зробити - створити користувача, від імені якого буде проходити реплікація та прописати певні параметри, щоб стартувати реплікацію

Тож на серверах server1 і 2 відповідно виконуємо:

```
mysql@server1 > GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'replication'@'192.168.1.24' IDENTIFIED BY 'password';

mysql@server2> GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'replication'@'192.168.1.23' IDENTIFIED BY 'password';
```

Далі необхідно прив'язати слейви до своїх майстрів.

На server1 виконуємо:

```
mysql@server1 > show master status;

+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000006 | 7984     |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0,00 sec)
```

Використовуючи значення колонки "File" та "Position", пишемо запит в консолі mysql на сервері server2:

```
mysql@server2> CHANGE MASTER TO MASTER_HOST = "192.168.1.23", MASTER_USER = "replication", MASTER_PASSWORD = "password_of_user_replication", MASTER_LOG_FILE = "mysql-bin.000006", MASTER_LOG_POS = 7984;
mysql@server2> slave start;
```

Далі, навпаки на сервері 2 виконуємо:

```
mysql@server2> show master status;
```

Аналогічно, використовуємо отримані дані для запиту, проте вже на сервері1:

```
mysql@server1 > CHANGE MASTER TO MASTER_HOST = "192.168.1.24", MASTER_USER = "replication", MASTER_PASSWORD = "password_of_user_replication", MASTER_LOG_FILE = "mysql-bin.000011", MASTER_LOG_POS = 106;

mysql@server1 > slave start;
```

Перевірити статус реплікації можна виконавши в консолі MySQL:

```
show slave status \G

*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.1.23
                  Master_User: replication
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000011
          Read_Master_Log_Pos: 106
               Relay_Log_File: mysql-relay-bin.000083
                Relay_Log_Pos: 251
        Relay_Master_Log_File: mysql-bin.000011
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB: dbwordpress
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 106
              Relay_Log_Space: 551
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
```

```
             Master_SSL_CA_Path:
                Master_SSL_Cert:
              Master_SSL_Cipher:
                 Master_SSL_Key:
          Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                 Last_IO_Errno: 0
                 Last_IO_Error:
                Last_SQL_Errno: 0
                Last_SQL_Error:
1 row in set (0.00 sec)
```

На таблиці 2.1 показані команди, які можна використовувати при конфігуруванні реплікації.

Таблиця 5.1

Команди конфігурування реплікації

Команда	Опис
SLAVE START	Запустити підлеглий (Slave) потік.
SLAVE STOP	Вимкнути підлеглий (Slave) потік.
SET SQL_LOG_BIN=0	Відключає реєстрацію модифікації, якщо користувач має привілей process.
SET SQL_LOG_BIN=1	Заново запускає реєстрацію модифікації, якщо користувач має привілей process.
SET SQL_SLAVE_SKIP_COUNTER=n	Пропустити наступні n подій від головної системи.

Продовження табл. 5.1

Команда	Опис
RESET MASTER	Видаляє всі виконавчі файли реєстрації, перераховані в індексному файлі, очищаючи індексний файл binlog. У версіях до 3.23.26 називалася FLUSH MASTER
RESET SLAVE	Підлеглий забуває позицію реплікації в головних файлах реєстрації. У версіях до 3.23.26 називалася FLUSH SLAVE, тепер вже ні.
LOAD TABLE tblname FROM MASTER	Завантажити копію таблиці з головної системи на підлеглу
CHANGE MASTER TO master_def_list	Змінює параметри для значень, визначених у master_def_list і перезапускає підлеглий процес. master_def_list являє собою розділений комами список master_def, де master_def один з елементів наступного переліку: MASTER_HOST, MASTER_USER, MASTER_PASSWORD, MASTER_PORT, MASTER_LOG_FILE, MASTER_LOG_POS. Наприклад: CHANGE MASTER TO MASTER_HOST= 'master2.mycompany.com', MASTER_USER = 'replication', MASTER_PASSWORD = 'bigs3cret',

Продовження табл. 5.1

Команда	Опис
	MASTER_PORT = 3306, MASTER_LOG_FILE = 'master2-bin.001', MASTER_LOG_POS = 4; Необхідно визначити лише значення, які повинні бути змінені. Значення, які вносяться, залишаться тими ж самими, за винятком того випадку, коли змінюється головний комп'ютер або порт. У цьому випадку підлеглий вважає, що, так як Ви з'єднуєтеся з іншим головним комп'ютером або іншим портом, головна система змінилася. Отже, старі значення файлу реєстрації і позиції більше незастосовні і будуть автоматично скинуті до порожньої рядку і 0 відповідно (це значення початку). Необхідно звернути увагу, що коли перезапускається підлеглий сервер, то він буде пам'ятати останній головний сервер. Якщо це не потрібно, ми повинні видалити файл master.info перш, ніж виконати перезапуск, і підлеглий буде читати дані на головний сервер з файлу my.cnf або з командного рядка.
SHOW MASTER STATUS	Забезпечує інформацію стану binlog.

Продовження табл. 5.1

Команда	Опис
SHOW SLAVE STATUS	Забезпечує інформацію стану істотних параметрів підпорядкованої системи.
SHOW MASTER LOGS	Вносить до списку виконавчі протоколи на головній системі. Доступно з версії 3.23.28. Ви повинні використовувати цю команду до PURGE MASTER LOGS TO.
PURGE MASTER LOGS TO logname	Видаляє всі файли реєстрації реплікацій, які перераховані в індексі файлу реєстрації як знаходяться до певного файлу реєстрації, і потім видаляє їх з індексу файлу реєстрації так, щоб даний файл реєстрації тепер став першим. Доступно з версії 3.23.28. Приклад: PURGE MASTER LOGS TO mysql-bin.010 Ця команда нічого не буде робити, якщо Ви маєте справу з активним підлеглим сервером, який в даний час читає один з файлів реєстрації, які Ви пробуєте видаляти. Команда безпечна для виконання в той час, коли всі підлеглі копіюють дані. Спочатку необхідно з’ясувати, які файли обробляє кожен підлеглий командою SHOW SLAVE STATUS

Також можна налаштовувати сервери в файлах конфігурації my.ini. Опції, доступні для конфігурування основного сервера представлені в таблиці 5.2

Таблиця 5.2

Опції конфігурування основного серверу

Команда	Опис
log-bin=filename	Вказує місце розташування двійкового журналу оновлень, в якому будуть вестися запису. Зверніть увагу: якщо переданий параметр має розширення (Наприклад log-bin = / mysql / logs / replication.log), то в разі виклику команди FLUSH LOGS версії MySQL нижче 3.23.24 не будуть правильно працювати під час реплікації. Ця проблема усунена у версії 3.23.25. Тепер, якщо використовується такий спосіб визначення імені журналу, команда FLUSH LOGS для двійкових журналів буде ігноруватися. У версії 3.23.26 або вище потрібно використовувати для цього команди RESET MASTER і RESET SLAVE можна використовувати ці опції якщо буде потреба мати ім'я, яке буде незалежне від імені хоста
log-bin-index=filename	Так як користувач може виконувати команду FLUSH LOG, потрібно знати, який журнал є активним в даний час, а також які журнали використовувалися раніше і в якій

Продовження табл.. 5.2

Команда	Опис
	послідовності вони змінювалися. Ця інформація збережена в індексному файлі двійкового журналу, ім'я якого типово буває ім'я_хоста.index. Ім'я та вміст даного файлу не слід змінювати. Приклад: log-bin-index = db.index
sql-bin-update-same	Якщо включена ця опція, то при встановленні значення змінної SQL_LOG_BIN це ж значення буде автоматично встановлено і для змінної SQL_LOG_UPDATE, і навпаки.
binlog-do-db=database_name	Вказує головному серверу, що він повинен вести записи про оновлення в журналі, якщо поточна база даних - database_name. Всі інші бази даних ігноруються. Зверніть увагу: при використанні цієї опції ви повинні бути впевнені, що оновлення будуть проводитися тільки в поточній базі даних. Приклад: binlog-do-db = sales
binlog-ignore-db=database_name	Вказує головному серверу, що якщо поточна база даних - database, то запису про оновлення не повинні вестися в довічним журналі. При використанні цієї опції ми повинні бути впевнені, що оновлення будуть проводитися тільки в поточній базі даних.

Опції конфігурування підлеглого сервера представлені в таблиці 5.3

Таблиця 5.3

Опції конфігурування підлеглого серверу

Команда	Опис
master-host=host	Ім'я хоста головного сервера або IP-адресу для реплікації. Якщо значення цієї опції не встановлено, потік підлеглого серверу не буде запущений. Зверніть увагу: установка master-host буде ігноруватися, якщо існує коректний файл master.info. Можливо, краще було б назвати ці опції як-небудь інакше, щось на зразок bootstrap-master-host, але міняти їх імена вже пізно.
master-user=username	Ім'я користувача, яке підлеглий сервер буде використовувати для аутентифікації при приєднанні до головного сервера. Користувач повинен мати привілей FILE. Якщо користувач головного сервера не встановлений, буде використано ім'я користувача test. Якщо вдасться вважати значення з файлу master.info, то воно буде мати більший пріоритет.
master-password=password	Пароль, який буде використовуватися при під'єднанні підлеглого сервера до головного серверу. Якщо вдасться вважати значення з файлу master.info, то воно матиме більший пріоритет.

Продовження табл. 5.3

Команда	Опис
master-port=portnumber	Порт, який слухає головний сервер. Якщо він не встановлений, використовується Відкомпілювати установка MYSQL_PORT. Це має бути значення 3306, якщо воно не було змінено за допомогою опцій configure. Якщо вдасться вважати значення з файлу master.info, то воно буде мати більший пріоритет
report-host	Ім'я хоста або IP-адреса підлеглого сервера, який передається головному серверу під час реєстрації підлеглого сервера. Може бути виведений командою SHOW SLAVE HOSTS. Не встановлюйте цю опцію, якщо не хочете, щоб підлеглий сервер реєструвався на головному сервері. Для того, щоб головний сервер встановив з'єднання з підлеглим сервером, йому недостатньо просто отримати IP-адресу підлеглого сервера із з'єднання. Через NAT і інших проблем маршрутизації IP-адреса може бути неприпустимим для під'єднання головного сервера або інших хостів до підлеглого сервера.
report-port	Доступна для версій вище 4.0.0. Порт для з'єднання з підлеглим сервером, ім'я

Продовження табл. 5.3

Команда	Опис
	хоста або IP-адреса якого були передані головному серверу при реєстрації підлеглого сервера. Порт потрібно встановлювати лише в тому випадку, коли підлеглий сервер слухає порт, який заданий не по замовчуванню, або якщо є спеціальний тунель від головного сервера або інших клієнтів до підлеглому сервера. Не використовуйте цю опцію, якщо не впевнені у своїх діях.
master-info-file=filename	Розташування файлу, в який записується інформація про те, де на головному сервері сталася зупинка під час виконання реплікації. За умовчанням це файл master.info в директорії даних. Змінювати це місце розташування немає необхідності.

РОЗДІЛ 6
ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА

6.1 Розрахунок норм часу на виконання науково-дослідної роботи

Час є важливими ресурсом при виконанні науково-дослідної роботи. Розподіляти час слід ефективно оскільки коефіцієнт корисної дії залежить від оптимального використання часу.

Роботу над дослідницькою роботи необхідно розділити на кілька етапів, що дозволить полегшити і структурувати виконання дослідження.

Основні етапи при виконанні розробки системи детектування емотиконів наступні:

1) Підготовка опису задачі;
2) Збір необхідної інформації по розробці програмного комплексу;
3) Розробка алгоритму індукції ПММ;
4) Вибір парадигми програмування;
5) Розробка бази знань предметної області;
6) Вибір програмного забезпечення для роботи системи;
7) Розробка основних модулів програмного продукту;
8) Тестування роботи додатку.

Показником тривалості виконання окремих робіт використовують нормативи часу або попередній досвід. Як результат планується отримати якісну систему, яка буде зручною в використанні та матиме функціональні можливості, що повністю покриватимуть досліджувану предметну область і матиме високі показники продуктивності, це зробить систему конкурентоспроможною на ринку та актуальною для використання.

Виконавцем усіх операцій по розробці додатку являється інженер.

Витрати часу по окремих операціях технологічного процесу відображені в таблиці 6.1.

Таблиця 6.1

Операції технологічного процесу та час їх виконання

№ з/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1.	Підготовка опису задачі.	Інженер	8
2.	Збір необхідної інформації по розробці програмного комплексу	Інженер	24
3.	Розробка алгоритму індукції ПММ	Інженер	24
4.	Вибір парадигми програмування	Інженер	8
5.	Розробка бази знань предметної області	Інженер	48
6.	Вибір програмного забезпечення для роботи системи.	Інженер	12
7.	Розробка основних модулів програмного продукту	Інженер	48
8.	Тестування роботи додатку	Інженер	24
Разом			**196**

6.2. Визначення витрати на оплату праці та соціальні заходи

Відповідно до Закону України “Про оплату праці” заробітна плата – це “винагорода, обчислена, як правило, у грошовому виразі, яку власник або уповноважений ним орган виплачує працівникові за виконану ним роботу”.

Розмір заробітної плати залежить від складності та умов виконуваної роботи, професійно-ділових якостей працівника, результатів його праці та господарської діяльності підприємства. Заробітна плата складається з основної та додаткової оплати праці.

Основна заробітна плата нараховується на виконану роботу за тарифними ставками, відрядними розцінками чи посадовими окладами і не залежить від результатів господарської діяльності підприємства.

Додаткова заробітна плата – це складова заробітної плати працівників, до якої включають витрати на оплату праці, не пов'язані з виплатами за фактично відпрацьований час.

Для розрахунку заробітної плати кількість робочих днів у місяці в середньому приймаємо – 24,5 дні/міс., або 196 год./міс. (тривалість робочого дня – 8 год.).

Місячний оклад кожного працівника враховується згідно існуючих на даний час тарифних окладів. Тарифні ставки: керівник проекту – 4,5 грн./год., інженер – 3,0 грн./год., консультант – 3,5 грн./год., технік – 3,0грн./год., лаборант – 2,0 грн./год.

Основну заробітну плату розраховуємо за формулою:

$$З_{осн.} = Т_с \cdot К_г \text{ ,} \qquad (6.1)$$

де $Т_с$ – тарифна ставка, грн.;

$К_г$ – кількість відпрацьованих годин.

Під час розробки програмного продукту був задіяний тільки інженер, отже основна заробітна плата буде розраховуватись наступним чином:

$$З_{осн.} = 3 \cdot 196 = 588 \text{ грн.}$$

Додаткова заробітна плата становить 10–15 % від суми основної заробітної плати.

$$З_{дод.} = З_{осн.} \cdot К_{допл.}, \qquad (6.2)$$

де $К_{допл}$ – коефіцієнт додаткових виплат працівникам, 0,1–0,15 (в даному випадку візьмемо 0,15).

$$З_{дод} = 588 \cdot 0,15 = 88,2 \text{ грн.}$$

Звідси загальні витрати на оплату праці ($В_{о.п.}$) визначаються за формулою:

$$В_{о.п.} = З_{осн.} + З_{дод.}, \quad (6.3)$$

$$В_{о.п.} = 588 + 88,2 = 676,2 \text{ грн.}$$

Крім того, слід визначити відрахування на соціальні заходи:

1) фонд страхування на випадок безробіття – 1,3 %;
2) фонд по тимчасовій втраті працездатності – 2,9 %;
3) пенсійний фонд – 32,3 %.

У сумі зазначені відрахування становлять 37,5 %.

Отже, сума відрахувань на соціальні заходи буде становити:

$$В_{с.з.} = Ф_{оп} \cdot 0,375 , \quad (6.4)$$

де $Ф_{оп}$ – фонд оплати праці, грн.

$$В_{с.з.} = 588 \cdot 0,375 = 220,5 \text{ грн.}$$

Проведені розрахунки витрат на оплату праці зведемо у таблицю 6.2.

Таблиця 6.2

Зведені розрахунки витрат на оплату праці

№ з/п	Категорія працівни–ків	Основна заробітна плата, грн.			Додаткова заробітна плата, грн.	Нарахув. на ФОП, грн.	Всього витрати на плату праці, грн. 6=3+4+5
		Тарифна ставка, грн.	К–сть відпра–цьов. год.	Фактично нарах. з/пл., грн.			
А	Б	1	2	3	4	5	6
1.	інженер	3	588	676,2	88,2	220,5	676,2

6.3. Визначення матеріальних витрат

Матеріальні витрати це добуток кількості матеріалів та їх ціни:

$$M_{ві} = q_i \cdot p_i \,, \qquad (6.5)$$

де q_i – кількість витраченого матеріалу і–го виду;

p_i – ціна матеріалу і–го виду.

Звідси, загальні матеріальні витрати можна визначити:

$$З_{м.в.} = \sum M_{ві} \,. \qquad (6.6)$$

Проведені розрахунки занесемо у таблицю 3.3.

Таблиця 6.3

Зведені розрахунки матеріальних витрат

Найменування матеріальних ресурсів	Один. виміру	Норма витрат	Ціна за один., грн.	Затрати матер., грн.	Транспортно-заготівельні витрати, грн.	Загальна сума витрат на матер., грн.
1. Основні матеріали						
Площадка для розміщення веб–додатку	штук	1	200	–	–	200
2. Допоміжні витрати						
Використання мережі Internet	години	–	124	75	–	75
Разом:						**275**

6.4. Визначення витрат на електроенергію

Затрати на електроенергію 1–ці обладнання визначаються за формулою:

$$З_в = W \cdot T \cdot S\,, \qquad (6.7)$$

де W – необхідна потужність, кВт;

T – кількість годин роботи обладнання;

S – вартість кіловат-години електроенергії.

Вартість кіловат-години електроенергії слід приймати згідно існуючих на даний час тарифів (0,203 грн. + 20% ПДВ за 1 кВт). Отже, 1 кВт з ПДВ коштує 0,2436 грн.

Потужність комп'ютера для створення проекту – 500 Вт, кількість годин роботи обладнання згідно таблиці 6.1 – 124 години.

Тоді,

$$З_в = 0{,}5 \cdot 124 \cdot 0{,}2436 = 15{,}10 \text{ грн.}$$

6.5. Розрахунок амортизаційних відрахувань

Характерною особливістю застосування основних фондів у процесі виробництва є їх відновлення. Для відновлення засобів праці у натуральному виразі необхідне їх відшкодування у вартісній формі, яке здійснюється шляхом амортизації.

Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення.

Для визначення амортизаційних відрахувань застосовуємо формулу:

$$A = \frac{Б_В \cdot Н_А}{100\%}, \qquad (6.8)$$

де A – амортизаційні відрахування за звітний період, грн.;

$Б_В$ – балансова вартість групи основних фондів на початок звітного періоду, грн.;

$Н_А$ – норма амортизації, %.

Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %).

Для даного проекту засобом розробки є комп'ютер. Його сума становить 5400 грн. Отже, амортизаційні відрахування будуть рівні:

$$А = 5400 \cdot 5\% / 100\% = 270 \text{ грн.}$$

Оскільки робота виконувалась 124 години, то амортизаційні відрахування будуть становити:

$$А = 270 \cdot 124 / 124 = 270 \text{ грн.}$$

6.6. Обчислення накладних витрат

Накладні витрати пов'язані з обслуговуванням виробництва, утриманням апарату управління спілкою та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$$Н_в = В_{о.п.} \cdot 0{,}2 \ldots 0{,}6 , \qquad (6.9)$$

де $Н_в$ – накладні витрати.

Отже, накладні витрати:

$$Н_в = 676{,}2 \cdot 0{,}4 = 270{,}48 \text{ грн.}$$

Результати проведених вище розрахунків зведемо у таблицю 6.4.

Таблиця 6.4

Кошторис витрат на НДР

Зміст витрат	Сума, грн.	В % до загальної суми
Витрати на оплату праці (основну і додаткову заробітну плату)	676,2	39,1
Відрахування на соціальні заходи	220,5	12,7
Матеріальні витрати	275	15,9
Витрати на електроенергію	15,10	0,8
Амортизаційні відрахування	270	15,6
Накладні витрати	270,48	6,3
Собівартість	1727,28	100

Собівартість ($С_в$) програмного продукту розрахуємо за формулою:

$$С_в = В_{о.п.} + В_{с.з.} + З_{м.в.} + З_в + А + Н_в \text{ .} \qquad (6.10)$$

Отже, собівартість програмного продукту дорівнює:

$С_в$ = 676,2 + 220,5+ 275 + 15,10+ 270 + 270,48= 1727,28 грн.

6.7. Складання кошторису витрат та визначення собівартості НДР

Ціну НДР можна визначити за формулою:

$$Ц = \frac{С_В \cdot (1 + Р_{рен}) + К \cdot В_{н.і.}}{К} \cdot (1 + ПДВ) \text{ ,} \qquad (6.11)$$

де $Р_{рен.}$ – рівень рентабельності, 30 %;

$К$ – кількість замовлень, од. (встановлюється лише при розробці програмного продукту та мікропроцесорних систем);

$B_{н.і.}$ – вартість носія інформації, грн. (встановлюється лише при розробці програмного продукту);

ПДВ – ставка податку на додану вартість, (20 %).

Оскільки розробка є прикладною, і використовуватиметься тільки для одного підприємства, то для розрахунку ціни не потрібно вказувати коефіцієнти *К* та $B_{н.і}$, оскільки їх в даному випадку не потрібно.

Тоді, формула для обчислення ціни розробки буде мати вигляд:

$$Ц = C_B \cdot (1 + P_{рен}) \cdot (1 + ПДВ). \quad (6.12)$$

Звідси ціна на проект складе:

$$Ц = 1727{,}28 \cdot (1 + 0{,}3) \cdot (1 + 0{,}2) = 2694{,}55 \text{ грн.}$$

6.8. Визначення ефективності виробництва

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність ($E_р$) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_р = \frac{П}{C_B}, \quad (6.13)$$

де *П* – прибуток;

C_B – собівартість.

Плановий прибуток ($П_{пл}$) знаходимо за формулою:

$$П_{пл} = Ц - C_в . \quad (6.14)$$

Розраховуємо плановий прибуток:

$$П_{пл} = 2694{,}55 – 1727{,}28 = 967{,}27 \text{ грн.}$$

Отже, формула для визначення економічної ефективності набуде вигляду:

$$E_p = \frac{П_{пл}}{C_в}. \qquad (6.15)$$

Тоді,

$$E_p = 967{,}27 / 1727{,}28 = 0{,}55 .$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень (T_p):

$$T_p = \frac{1}{E_p} , \qquad (6.16)$$

Термін окупності дорівнює:

$$T_p = 1 / 0{,}55 = 1{,}8 \text{ роки}$$

Висновок:

В організаційно-економічній частині розробки системи детектування емотиконів проекту було розраховано основні техніко-економічні показники розробки веб-сайту (таблиця 6.5).

Розраховане значення економічної ефективності, яке становить 0,55, що є високим значенням.

Так самими нормальним є термін окупності, який повинен коливатися від 1 до 3 років, тоді розробка вважається доцільною і економічно вигідною. Для даного продукту він становить 1,8 років.

Таблиця 6.5

Техніко–економічні показники НДР

№ п/п	Показник	Значення
1.	Собівартість, грн.	1727,28
2.	Плановий прибуток, грн..	967,27
3.	Ціна, грн.	2694,55
4.	Економічна ефективність	0,55
5.	Термін окупності, рік	1,8

Отже, даний проект може бути впроваджений та мати подальший розвиток, оскільки він є економічно вигідним за всіма основними техніко-економічними показниками.

РОЗДІЛ 7
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1. Охорона праці

Охорона праці є важливою при розробці системи аналізу та детектування емотиконів. Робота над даною системою передбачає використання засобів обчислювальної техніки, а отже необхідно враховувати вимоги, що до забезпечення безпечних умов праці.

Закон України «Про охорону праці» — закон, що визначає основні положення щодо реалізації конституційного права громадян на охорону їх життя і здоров'я у процесі трудової діяльності, регулює за участю відповідних державних органів відносини між власником підприємства, установи і організації або уповноваженим ним органом і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в Україні.

Також варто згадати міжнародне законодавство про охорону праці, що являє собою систему міжнародно-правових актів, спрямованих на захист працівників від професійних ризиків. Право на охорону праці належить до невід'ємних прав людини, записаних у фундаментальних міжнародних документах, таких, як Загальна декларація прав людини ООН та Міжнародний пакт ООН про економічні, соціальні та культурні права.

Принципи охорони праці також відображені в законодавстві Євросоюзу, про охорону праці, зокрема в директиві 89/391/ЄЕС "Про впровадження заходів для поліпшення безпеки та охорони здоров'я працівників під час роботи". Варто згадати, що вимоги з міжнародного законодавства є більш пріоритетні чим українські у випадку, якщо вони є жорсткішими.

Важливим фактором ефективної та безпечної роботи з обчислювальною технікою є правильно облаштоване робоче місце.

Робоче місце – це частина простору, у якому дослідник здійснює наукову діяльність, і проводить більшу частину робочого дня. Робоче місце дослідника котрий виконував розробку системи детектування та налізу емотиконів було добре пристосоване до праці, правильно і доцільно впорядковане, щодо простору, форми, розміру забезпечувало зручне положення під час роботи і високу продуктивність праці при найменшому фізичному і психічному напруженні.

Слід також згадати що при правильній організації робочого місця продуктивності праці дослідника зростає з 8 до 20 відсотків.

Згідно з ГОСТ 12.2.032-78 конструкція робочого місця та взаємне розташування усіх її елементів відповідали антропометричним, фізичним і психологічним вимогам. Важливе значення було надано характеру роботи.

Головними елементами робочого місця дослідника є стіл та крісло. Робоче місце при виконанні робіт у положенні сидячи було організоване згідно з ГОСТ 12.2.032-78. Планування робочого місця передбачало чітке розміщення предметів, інструментів праці та задоволення вимог фізіології, що випливають. Те, що використовується частіше, лежить у зоні легкої досяжності робочого простору.

Параметри робочого місця обирались відповідно до антропометричних характеристик.

Важливим елементом робочого місця дослідника є крісло. Воно обиралося згідно ГОСТ 21.889-76. Вибір крісла був продиктований тим, що за будь-якого робочого положення дослідника його поза мусить бути фізіологічно правильно обґрунтованою, тобто положення частин тіла має бути оптимальним. Для задоволення вимог фізіології, що випливають із аналізу стану тіла людини у положенні сидячи, конструкція робочого сидіння відповідала основним вимогам:

- допускала можливість зміни розташування тіла, тобто. Забезпечувала вільне пересування корпуса та кінцівок тіла одної відносно іншої;
- допускала регулювання висоти;
- володіла увігнутою поверхнею,
- володіла невеликим нахилом.

Важливим моментом було раціональне розміщення на робочому місці документації, канцтоварів, що забезпечило працюючому зручну робочу позу.

Слід також згадати, що створення сприятливих умов праці та правильне естетичне оформлення робочого місця дослідника мало велике значення для полегшення роботи над розробкою системи детектування та аналізу емотиконів, а підвищення його ергономічних характеристик, позитивно впливало на продуктивність праці.

7.2. Безпека в надзвичайних ситуаціях

7.2.1. Вступ

Правовою основою діяльності в галузі пожежної безпеки є Конституція, постанови Верховної Ради України, укази і розпорядження Президента України, декрети, постанови та розпорядження Кабінету Міністрів України, рішення органів виконавчої влади, місцевого та регіонального самоврядування, прийняті в межах їх компетенції.

Основним документом щодо захисту населення від наслідків надзвичайних ситуацій є Закон «Про цивільну оборону України» »(від 02.10.2012). Відповідно до цього Закону громадяни України мають право на захист свого життя і здоров`я від наслідків аварій, катастроф, значних пожеж, стихійного лиха. Держава як гарант цього права створює систему цивільної оборони, метою якої захист населення від небезпечних наслідків аварій і катастроф техногенного та воєнного характеру.

Згідно цього вищезгаданого закону можна дати визначення поняття цивільного захисту:

цивільний захист – система організаційних, інженерно-технічних, санітарно-гігієнічних, протиепідемічних та інших заходів, які здійснюються центральними і місцевими органами виконавчої влади, органами місцевого самоврядування, підпорядкованими їм силами і засобами, підприємствами, установами та організаціями незалежно від форми власності, добровільними рятувальними формуваннями, що забезпечують виконання цих заходів з метою запобігання та ліквідації надзвичайних ситуацій, які загрожують життю та здоров'ю людей, завдають матеріальних збитків у мирний час і в особливий період.

7.2.2. Пожежа

Пожежа — це стихійне поширення горіння, яке виявляється в нищівній дії вогню, що вийшов з-під контролю людини. Пожежі можуть виникнути як наслідок уражаючого фактора від світлового випромінювання ядерного вибуху, при застосуванні звичайних засобів ураження і спеціальних (піреогелю, терміту, електрону і білого фосфору). Стихійні пожежі можуть виникнути внаслідок розрядів блискавки, самозапалювання сіна й торфу, від залишеного багаття, непогашеного сірника, тліючого недопалка, іскор із транспортних засобів, неправильного користування електроприладами, несправності нагрівних приладів, механічного нагрівання та іскроутворення.

Горіння включає три необхідні інгредієнти: паливо, тепло і кисень.Пожежа виникає тільки тоді, коли вони всі наявні. Достатньо винести з системи один з них — пожежа припиняється. Цього можна досягти двома способами: охолодженням вогню (видалення тепла), як правило за допомогою води, позбавленням вогню палива, припинення доступу кисню.

Пожежі за природою походження можна поділити на два види:

- Природні
- Антропогенні

До *природних* належать пожежі, що виникають унаслідок прямих ударів блискавки (розрядів атмосферної електрики), виверження вулканів, самозаймання торфу, вугілля тощо. Кількість таких пожеж незначна - менше 1%.

Антропогенні пожежі - це пожежі, що виникають на об'єктах, створених руками людей (у синтетичному середовищі, прямо чи побічно пов'язані з людським чинником, тобто з пожежонебезпечною діяльністю людини або невтручанням людини для запобігання пожежонебезпечним ситуаціям). Такі пожежі виникають у більшості випадків. В предметах цивільного захисту та охорони праці виділяється визначення пожежної безпеки об’єкту.

Пожежна безпека об'єкта - стан об'єкта, за якого з регламентованою імовірністю виключається можливість виникнення і розвитку пожежі та впливу на людей її небезпечних факторів, а також забезпечується захист матеріальних цінностей. В приміщенні в котрому проводилось виконання розробка системи детектування попередньо були проведені міри, що до забезпечення пожежної безпеки.

Основними напрямками забезпечення пожежної безпеки було усунення умов виникнення пожежі та мінімізація її наслідків. В приміщенні знаходяться системи пожежної безпеки, спрямовані на запобігання пожежі, дії на людей та матеріальні цінності небезпечних факторів пожежі, в тому числі їх вторинних проявів. До таких факторів, згідно з ГОСТ 12.1.004-91, належать: полум'я та іскри, підвищена температура навколишнього середовища, токсичні продукти горіння й термічного розкладу матеріалів і речовин, дим, знижена концентрація кисню.

Вторинними проявами небезпечних факторів пожежі вважаються: уламки, частини зруйнованих апаратів, агрегатів, установок, конструкцій;

радіоактивні та токсичні речовини і матеріали, викинуті зі зруйнованих апаратів та установок; електричний струм, пов'язаний з переходом напруги на струмопровідні елементи будівельних конструкцій, апаратів, агрегатів внаслідок пошкодження ізоляції під дією високих температур; небезпечні фактори вибухів, пов'язаних з пожежами; вогнегасні речовини.

7.2.3. Висновки

В даному розділі було наведено основні законодавчі положення, щодо цивільної оборони в Україні. Приведені типи пожеж за причиною та джерелом появи. Як висновок можна сформувати основний тезис про те, що цивільна оборона України є державною системою органів управління, сил і засобів, що створюється для організації і забезпечення захисту населення від наслідків надзвичайних ситуацій техногенного, екологічного, природного характеру». Цивільна оборона України створюється як складова частина загальної оборони України і державної системи попередження надзвичайних ситуацій зокрема пожеж у разі та дій у разі їх виникнення.

РОЗДІЛ 8
ЕКОЛОГІЯ

8.1. Актуальність охорони навколишнього середовища

В Україні щороку продовжує зростати антропогенне та техногенне навантаження на навколишнє середовище. Наша держава має найвищий в Європі рівень розораності сільськогосподарських угідь, споживання водних ресурсів, вирубки лісів. Близько 15% території України з населенням понад 10 млн. осіб знаходиться в критичному екологічному стані. В Україні близько 70% поверхневих вод і значна частка запасів підземних, втратили своє значення як джерела питної води. Занепокоєність викликає стан басейну р. Дніпро, який забезпечує питною водою понад 75% (35 млн.) населення країни.

Якість води Дніпровського басейну поступово погіршується. Збільшуються середньорічні показники вмісту у воді амонійного азоту, нітратів, фосфатів та підвищення мінералізації води, забруднення водних об’єктів фенолами та нафтопродуктами. Погіршується і якість підземних вод внаслідок інтенсивної експлуатації продуктивних водоносних горизонтів. Виявлено понад 290 сформованих осередків забруднення підземних вод в основних водоносних горизонтах.

В Україні основним документом який нормує екологічну ситуацію є Закон України «Про охорону навколишнього природного середовища». Завданням законодавства про охорону навколишнього природного середовища є регулювання відносин у галузі охорони, використання і відтворення природних ресурсів, забезпечення екологічної безпеки, запобігання і ліквідації негативного впливу господарської та іншої діяльності на навколишнє природне середовище, збереження природних ресурсів, генетичного фонду живої природи, ландшафтів та інших природних комплексів, унікальних територій та природних об'єктів, пов'язаних з історико-культурною спадщиною.

8.2. Забруднення довкілля що виникають при роботі на ТОВ "Астон"

Підприємство «Астон» яке є базою проведення наукового дослідження засноване у грудні 1991 р. за умов становлення та розвитку ринкових відносин, одне з перших приватних підприємств у м. Тернополі.

Видавництво володіє великою кількістю матеріально-технічних засобів і високими об’ємами виробництва, як на цехах так і в офісах. Все устаткування та офісна техніка, затрачують папір використовують фарби та тонери.

8.3. Заходи щодо зниження забруднення довкілля

Великої шкоди довкіллю завдають не утилізовані агрегати друкарських апаратів. Найбільше це стосується пластику, який має занадто великий період розпаду. Ця проблема вимагає ефективних рішень щодо утилізації.

Поширена останнім часом тенденція обов'язкової переробки і використання відпрацьованих матеріалів торкнулася і використаних картриджів для принтерів. На видавництві, де використовується безліч копіювальних апаратів, знають, що картриджі мають властивість при постійному завданні механічних пошкоджень бруднити навколишні предмети. Це є небезпечним для здоров'я, оскільки тонер , який знаходиться у картриджі, надзвичайно токсичний. В оригінальних картриджах він надійно запечатаний, а після перезаправки картриджа корпус втрачає свою герметичність. Навіть якщо уникати попадання попадання залишків барвника на шкіру, - частинки тонера дуже дрібні (3-4 мікрона), потрапляючи в повітря, погано впливають на здоров'я оточуючих.

У багатьох цивілізованих країнах світу, де дбають про довкілля , вже давно заборонено викидати картриджі, як і іншу використану офісну техніку в сміттєві баки. Тому в офісі підприємства було організовано пункт збирання

відпрацьованих картриджів від принтерів для подальшої правильної та безпечної їх утилізації.

В 2002 році, європейським парламентом був прийнятий закон, що забороняє вставляти в картриджі спеціальні електронні чіпи, які унеможливлювали вторинне використання оригінальних картриджів .

Найбільш поширеним у всьому світі і найбезпечнішим способом утилізації картриджів сьогодні є їх переробка. Утилізовувати картриджі слід компаніям, що виробляють витратні матеріали для картриджів і компоненти старого картриджа використовують для виробництва нових. Використаний картридж чиститься, зношені деталі міняються, картридж заправляється тонером і продається заново в упаковці власного виробництва. Цей процес називається відновленням картриджа. Відновлені картриджі коштують набагато дешевше нових. При оптовій закупівлі картриджів в таких компаніях видається нададуть упаковку для зберігання і транспортування на переробку відпрацьованих картриджів.

Переробка паперу несе численні переваги для навколишнього середовища. папір має один з найвищого ступенів переробки

Економія паперу – один з варіантів покращення впливу на екологію на виробництві. Незважаючи на те що в великій кількості використовують цифрові засоби, все таки на видавництві використовується велика кількість паперу не тільки для виробництва продукції, але й використання в офісі. Одним зі шляхів покращення ситуації можна назвати повну заміну всіх паперових носіїв на цифрові. Також проблему використання паперу на виробництві можна вирішити за рахунок повторного використання паперу. Вторинне рослинне волокно, яке є одним з основних джерел целюлозовмісної сировини для виробництва паперу і картону. Використання макулатури забезпечує утилізацію використаних целюлозних виробів, економію деревини, а також скорочення витрат енергії на виробництво паперу й картону. Ці волокна, при їх вторинному застосуванні, відрізняються за властивостями, які були притаманні ними до

проходження всього циклу операцій паперового виробництва, через вплив сушінням, в результаті якого відбулись деякі незворотні зміни: втрата їх еластичності, прозорості, пониження величини сил зв'язку між волокнами (порівняно з папером виготовленим не з вторинної сировини).

Більша частина макулатури переробляється в картон для тари (гофрований картон, коробочний картон) та упаковок, для чого використовують так звані «коричневі» марки макулатури як сировину. Високоякісні марки, що мають вищу ступінь білості, містять, як правило, значну кількість вибіленої целюлози, а отже можуть бути використані для виробництва такого паперу як газетний, для друку та письма, паперу санітарно-побутового призначення (туалетного паперу).

Чинний стандарт на макулатуру в Україні (ГОСТ 10700-97) визначає лише 13 марок макулатури залежно від способу розпускання, і не відповідає сучасним технологічним вимогам.

Для економії паперу також можна застосувати результати дослідження, проведеного Пенсільванським університетом, котре виявило, що зміна встановлених Microsoft за замовчуванням полів в 1 дюйм на 0.75 в текстовому редакторі Word здатне щорічно давати економію в $120,000 у вартості паперу й видаленні відходів. Що стосується лісів, дослідження затверджує, що такий крок збереже 72 акра (28,8 га) лісів. Такі прості зміни були впроваджені і на ТОВ "Астон".

В даному розділі було показано важливість бережливого використання вичерпних ресурсів та утилізації друкарських розхідних матеріалів.

ВИСНОВКИ

Результатом роботи над даною кваліфікаційною роботою на рівень магістра є розроблений алгоритм системи детектування та аналізу емотиконів. Було проведено дослідження існуючих систем, на основі принципу роботи яких було сформовано оптимальні алгоритми роботи існуючої системи.

В розробленому алгоритмі використані прихована марківська модель та алгоритми індукції контекстно-вільної граматики.

Розробка і дослідження, що були проведені в процесі кваліфікаційної роботи, які були здійсненні для математичного обґрунтування та реалізації на практиці алгоритму детектування. Ефективність методу було доведено виконанням дослідницького розділу для визначення похибок детектування символів емоційного наповнення в вхідних нетерміналах.

Дане дослідження доводить, що задача визначення приналежності групи символів до емотиконів в повідомленні, або визначення можливості породження слова-емотикона граматикою алгоритмічно розв'язна. Під час розв'язання цієї задачі можна побудувати дерево синтаксичного аналізу, тому створену програму можна назвати синтаксичним аналізатором.

Розроблений алгоритм можна використати в системах лексичного аналізу для зменшення омонімії тексту вхідних даних. Система буде забезпечувати відділення графем лексичного типу від графем нелексичного (емоційного) наповнення, а також здійснювати додатковий аналіз емоційної складової повідомлення для уточнення емоційного виведення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Мэтт Зандстра* PHP: объекты, шаблоны и методики программирования, 3-е издание = PHP Objects, Patterns and Practice, Third Edition — М.: «Вильямс», 2010. — С. 560. — ISBN 978-5-8459-1689-1.
2. Условные марковские процессы и их применение к теории оптимальногоуправления / *Р.Л. Стратонович* – М.: Книга по Требованию, 2012. – 311 С.ISBN 978-5-458-29616-8
3. *Стив Суэринг, Тим Конверс, Джойс Парк* PHP и MySQL. Библия программиста, 2-е издание = PHP 6 and MySQL 6 Bible — М.: «Диалектика», 2010. — С. 912. — ISBN 978-5-8459-1640-2.
4. Теория вероятностей и марковские процессы / Дынкин Е.Б., Юшкевич А.А. – М.: Книга по Требованию, 2013. – 237 с. ISBN 978-5-458-59967-2
5. *Марковские процессы Портенко Н. И., Скороход А. В., Шуренков В. М. М: "ВИНИТИ", 1989— С. 248*—ISBN. 5-9221-0186-2.
6. *Марковские процессы Портенко Н. И., Скороход А. В., Шуренков В. М. М: "ВИНИТИ", 1989— С. 248*—ISBN. 5-9221-0186-2.
7. *Джон Хопкрофт, Раджив Мотвани, Джеффри Ульман* Введение в теорию автоматов, языков и вычислений = Introduction to Automata Theory, Languages, and Computation. — М.: «Вильямс», 2002. — С. 528. — ISBN 0-201-44124-1
8. *Ахо, Альфред, В., Хопкрофт, Джон, Ульман, Джеффри, Д.* Структуры данных и алгоритмы = Data Structures and Algorithms. — Издательский дом «Вильямс», 2000. — С. 384. — ISBN 5-8459-0122-7
9. Jelinek, F.; Bahl, L.; Mercer, R. (1975). "Design of a linguistic statistical decoder for the recognition of continuous speech". *IEEE Transactions on Information Theory* **21** (3): 250. doi:10.1109/TIT.1975.1055384. edit
10. Xuedong Huang, M. Jack, and Y. Ariki (1990). *Hidden Markov Models for Speech Recognition*. Edinburgh University Press. ISBN 0-7486-0162-7.

11. Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon (2001). *Spoken Language Processing*. Prentice Hall. ISBN 0-13-022616-5.
12. M. Bishop and E. Thompson (1986). "Maximum Likelihood Alignment of DNA Sequences". *Journal of Molecular Biology* **190** (2): 159–165. doi:10.1016/0022-2836(86)90289-5. PMID 3641921.
13. Richard Durbin, Sean R. Eddy, Anders Krogh, Graeme Mitchison (1999). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press. ISBN 0-521-62971-3.
14. Ghahramani, Zoubin; Jordan, Michael I. (1997). "Factorial Hidden Markov Models".*Machine Learning* 29 (2/3): 245–273. doi:10.1023/A:1007425814087.
15. CAO: A Fully Automatic Emoticon Analysis System Based on Theory of Kinesics Michal Ptaszynski, Student Member, IEEE, Jacek Maciejewski, Pawel Dybala, Rafal Rzepka, and Kenji Araki—5MB. —Kioto University — Систем. вимоги: Pentium ; 32 Mb RAM ; Windows 95, 98, 2000, XP ; MS Word 97-2013 – Режим доступу до реферату: www.computer.org/csdl/trans/ta/2010/01/tta2010010046-abs.html
16. Robust kaomoji detection in Twitter Steven Bedrick, Russell Beckley, Brian Roark, Richard Sproat—5MB. —Oregon University — Систем. вимоги: Pentium ; 32 Mb RAM ; Windows 95, 98, 2000, XP ; MS Word 97-2013 – Режим доступу до реферату: www.aclweb.org/anthology/W12-2107
17. Research on Emoticons: Review of the Field and Proposal of Research Framework Michal Ptaszynski Rafal Rzepka Kenji Araki Yoshio Momouchi —5MB. —Kioto University — Систем. вимоги: Pentium ; 32 Mb RAM ; Windows 95, 98, 2000, XP ; MS Word 97-2013 – Режим доступу до реферату: arakilab.media.eng.hokudai.ac.jp/Araki_Lab/papers.html
18. ШейкоВ.М., Кушнаренко Н.М. Організація та методика науково-дослідницької діяльності: Підручник. – 3-є вид., стер. – К.: Знання-Прес, 2003. – 295 с.
19. Жидецький В.Ц. “Практикум з охорони праці” – Львів:Афіша, 2000. – 204с.

АНОТАЦІЯ

Кміть П.О Розробка інформаційної системи детектування та аналізу емотиконів в текстових даних

Метою дослідження є розробка алгоритму аналізу та детектування емотиконів з використання математичного апарату прихованих марківських моделей.

Предметом дослідження є прихована марківська модель, індукція контекстно вільної граматики в системі детектування та аналізу емотиконів.

Об’єкт дослідження текстові дані з омонімією породженою емотиконами.

Ціллю дослідження є здійснення розробки алгоритму для детектування емотиконів в вхідних текстових даних.

Дана система може використовуватись в системах лексичного аналізу для зменшення омонімії даних, або для покращення аналізу повідомлення.

Рік виконання дипломної роботи 2013

Рік захисту роботи 2013

Ключові слова: ПРИХОВАНА МОДЕЛЬ МАРКОВА, ЕМОТИКОН, МЕРЕЖІ ДОВІРИ, СТОХАСТИЧНА КОНТЕКСТНО-ВІЛЬНА ГРАМАТИКА.

Дипломна містить с. 90 , рис. - 12 , табл.. - 8 , плакати. – 8, додат. – 2, використаних джерел – 20 найменувань, 1 додаток.

ABSTRACT

Kmit P.O. Development of information system for detection and analysis of emoticons in text data.

Purpose of the research is development of algorithm for analysis and detection of emoticons using apparatus of hidden markov model.

Subjects of the study are hidden markov model, induction of stochastic context-free grammar in emoticon detection and analysis system.

Object of the study is text data with high homonymy caused by emoticons.

The goal is to develop algorithm that will allow to detect emoticons in input data.

The system can be used in lexical analysis systems to reduce homonymy, or to enhance message analysis

Thesis release year 2013

Defense of the thesis year 2013

Key words: HIDDEN MARKOV MODEL, EMOTICON, TRUST NETWORKS, STOCHASTIC CONTEXT-FREE GRAMMAR.

Explanatory note consists of pages - 90, figures - 12 , tables - 8, placards. – 8, annexes - 1.

ДОДАТОК

```
select {
    width: 150px;
}
input
{
    color: black;
    width: 145px;
}
body {
    background-color: #999;
    background-image:url(bg.jpg);
}
.border {

    border-width:1px;
    border:#333;
    outline:solid;
    outline-width:1px;
    outline-color:#666;

    background:grey;
    margin:15px;
    padding:10px;

    box-shadow: 0.3em 0.3em #999;

    width: 50%;
```

```
}

.text {
    margin:15px;
    font-weight: bold;
}
.border:hover{
    outline:double;
    outline-color:#F90;
}
.border>*{
    width: content-box;
}
#name{
    width: 95%;
}
<?php
ini_set('default_charset', 'UTF-8');
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Детектування емотиконів</title>
        <link type="text/css" rel="stylesheet" href="css/style.css"  />
    </head>
    <body>
        <div id="enter_string" class="border sub1">
```

```
    Enter a string: </br>
<form id="form1" name="form1" method="post"  >
    <input name="input_string" type="text" id="name"
    <?php
    if (isset($_POST['input_string'])) {
        echo 'value="' . $_POST['input_string'] . '"';
    }
    ?> />
    </br>
    <input type="submit" name="submit" id="submit" value="Аналізувати" />
</form>
</div>
<?php
if (isset($_POST['input_string'])) {
    $emoticon = $_POST['input_string'];
    $emoticon = deleteSymetrical($emoticon, "(", ")");
    $mas = explode(" ", $emoticon);
    $emoticons = array();

    foreach ($mas as $string) {
        $temp = checkWholeEmoticon($string);
        foreach ($temp as $mas2) {
            $emoticons[] = $mas2;
        }
    }
    foreach ($emoticons as $mas){
        echo "<div class='border'>";
        echo "Emoticon: ".$mas[1]."</br>";
        switch ($mas[2]) {
```

```
            case "1":
                echo "Emotion: Anger</br>";

                break;
            case "2":
                echo "Emotion: Like/Joy</br>";

                break;
            case "3":
                echo "Emotion: Sorrow</br>";

                break;
            case "4":
                echo "Emotion: Dislike</br>";

                break;
            default:
                echo "NO EMOTION FOUND";
                break;
        }
        echo "</div>";

    }
    //var_dump($emoticons);
}

function checkWholeEmoticon($_emoticon) {
    connectToDb();
    $emoticon = mysql_real_escape_string($_emoticon);
```

```
        $result = mysql_query("SELECT * FROM `emoticns_full` WHERE
POSITION(emoticon IN BINARY '$emoticon')
                                        ORDER BY POSITION(emoticon IN
BINARY '$emoticon')");
        $emoticons_found = array();

        while (!is_null($result) && $row = mysql_fetch_row($result)) {
            do {
                $pos = strpos($_emoticon, $row[1]);
                if ($pos !== FALSE) {
                    $_emoticon = substr_replace($_emoticon, "", $pos, strlen($row[1]));

                    $emoticons_found[] = $row;
                } else {
                    break;
                }
            } while (true);
        }

        $temp = checkEmoticonbyParts($_emoticon);
        if (!is_null($temp)) {
            foreach ($temp as $value) {
                if (!is_null($value)) {
                    $emoticons_found[] = $value;
                }
            }
        }
        return $emoticons_found;
    }
```

```
    function checkEmoticonbyParts($_emoticon) {
        connectToDb();
        $emoticon = mysql_real_escape_string($_emoticon);
        $emoticons = array();

        do {
            $result_eye = mysql_query("SELECT * FROM `emoticon_eyes` WHERE
POSITION(part IN BINARY '$emoticon')
                                        ORDER BY POSITION(part IN BINARY
'$emoticon')");
            $emoticon_eyes = array();
            $emoticon_nose = array();
            $emoticon_mouth = array();

            if (!is_null($result_eye) && $row = mysql_fetch_row($result_eye)) {
                $pos = strpos($emoticon, $row[1]);
                if ($pos !== FALSE) {
                    $emoticon = substr_replace($emoticon, "", 0, $pos);
                    $pos = strpos($emoticon, $row[1]);
                    $emoticon = substr_replace($emoticon, "", $pos, strlen($row[1]));
                    $emoticon_eyes = $row;
                }
                $result_nose = mysql_query("SELECT * FROM `emoticon_noses`
WHERE POSITION(part IN BINARY '$emoticon')
                                        ORDER BY POSITION(part IN BINARY
'$emoticon')");

                if (!is_null($result_nose) && $row = mysql_fetch_row($result_nose)) {
```

```
                $pos = strpos($emoticon, $row[1]);

                if ($pos !== FALSE) {
                    $emoticon = substr_replace($emoticon, "", 0, $pos);
                    $pos = strpos($emoticon, $row[1]);
                    $emoticon = substr_replace($emoticon, "", $pos, strlen($row[1]));
                    $emoticon_nose = $row;
                }
            }
            $result_mouths = mysql_query("SELECT * FROM `emoticon_mouths`
WHERE POSITION(part IN BINARY '$emoticon')
                                        ORDER BY POSITION(part IN BINARY
'$emoticon')");

            if (!is_null($result_mouths) && $row =
mysql_fetch_row($result_mouths)) {
                $pos = strpos($emoticon, $row[1]);
                if ($pos !== FALSE) {
                    $emoticon = substr_replace($emoticon, "", 0, $pos);
                    $pos = strpos($emoticon, $row[1]);
                    $emoticon = substr_replace($emoticon, "", $pos, strlen($row[1]));
                    $emoticon_mouth = $row;
                }
            } else {
                continue;
            }
        } else {
            break;
        }
```

```
        $temp = (string) $emoticon_eyes[1];
        if (isset($emoticon_nose[1])) {
            $temp.= (string) $emoticon_nose[1];
        }
        $temp.= (string) $emoticon_mouth[1];
        $emoticons[] = array("1", $temp, $emoticon_mouth[2], null);

    } while ($emoticon);
   return ($emoticons);
}

function connectToDb() {
    $host = "localhost";
    $user = "root";
    $pass = "lalulilelo123";

    $databaseName = "emoticons";

    $con = mysql_connect($host, $user, $pass) or die(
                mysql_error()
    );
    mysql_query('SET CHARACTER SET utf8');

    $dbs = mysql_select_db($databaseName, $con) or die(mysql_error());
}

function CloseDb() {
    //mysql_close($con);
}
```

```
        function deleteSymetrical($emoticon, $left, $right) {

            while (true) {
                echo $pos_begin = strrpos($emoticon, $left);
                if ($pos_begin !== false) {
                    echo $pos_end = strrpos($emoticon, $right, -1);
                    if ($pos_end !== false) {
                        $emoticon = substr_replace($emoticon, "", $pos_begin, 1);
                        $emoticon = substr_replace($emoticon, "", $pos_end – 1, 1);
                    } else {
                        break;
                    }
                } else {
                    break;
                }
            }
            return $emoticon;
        }
        ?>
    </body>
</html>
```