НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«До захисту допущено»
В.о. завідувача кафедри

_____ М.В.Грайворонський
(підпис)

“_____”_____2019 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

на тему: Генерування стеганографічних текстів на основі рекурентних нейронних мереж

Виконав: студент 4 курсу, групи ФБ – 52
(шифр групи)

Кравченко Олександр Михайлович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник к.т.н., доц. каф. ІБ, Родіонов Андрій Миколайович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент к.т.н., доц. каф. ФТЗЗІ, Прогонов Дмитро Олександрович _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.170101 «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту

Кравченко Олександру Михайловичу
(прізвище, ім'я, по батькові)

1. Тема роботи:

«Генерування стеганографічних текстів на основі рекурентних нейронних мереж»,

науковий керівник роботи:

доц. каф. ІБ, к.т.н., Родіонов Андрій Миколайович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «27 травня» 2019 р. № 1414-с

2. Термін подання студентом роботи: 10 червня 2019 р.

3. Вихідні дані до роботи: методи машинного навчання для генерування текстових послідовностей та алгоритми текстової стеганографії

4. Зміст роботи:
- Огляд існуючих методів текстової стеганографії
- Аналіз створення архітектури нейронної мережі
- Побудова власного стеганографічного алгоритму
- Аналіз результатів і порівняння з існуючими методами

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)
- Схема нейронної мережі

- Презентація

6. Дата видачі завдання: 7 вересня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Вивчення літератури за тематикою проекту	07.09.18 – 15.11.18	
2	Аналіз існуючих алгоритмів	15.11.18 – 20.12.18	
3	Написання першого розділу дипломної роботи	20.12.18 – 25.12.18	
4	Розгляд методів машинного навчання	10.01.19 – 10.02.19	
5	Написання другого розділу дипломної роботи	10.02.19 – 28.02.19	
6	Розробка власного алгоритму і написання програмної частини	28.02.19 – 25.03.19	
7	Написання третього розділу дипломної роботи	25.03.19 – 20.04.19	
8	Проходження переддипломної практики	20.04.19 – 20.05.19	
9	Аналіз отриманих результатів	10.05.19 – 20.05.19	
10	Написання четвертого розділу дипломної роботи	20.05.19 – 30.05.19	
11	Передзахист проекту	30.05.19	
12	Підготовка графічної частини дипломної роботи	30.05.19 – 20.06.19	
13	Захист дипломної роботи	20.06.19	

Студент _____ Кравченко О. М.
(підпис) (ініціали, прізвище)

Науковий керівник роботи _____ Родіонов А. М.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Дипломна робота складається з 56 сторінок та містить 11 таблиць, 17 рисунків і 23 літературних джерела.

Об'єктом дослідження є алгоритми текстової стеганографії. Предметом дослідження є методи генерування текстових стегоконтейнерів за допомогою рекурентних нейронних мереж.

Основними методами досліджень в роботі є методи машинного навчання, а також для оцінки наведених алгоритмів використовувались статистичні методи аналізу отриманих результатів. Було проаналізовано існуючі методи приховання інформації в текстових послідовностях.

Результатом виконання даної роботи є побудований алгоритм автоматичної генерації стеганографічних текстів, який може використовуватись для безпечної передачі та зберігання приватної інформації. Подальшими дослідженнями для покращення наведеного алгоритму є дослідження в галузі використання штучного інтелекту для генерації текстів.

РЕКУРЕНТНА НЕЙРОННА МЕРЕЖА, ЛАНЦЮГ МАРКОВА, СТЕГАНОГРАФІЧНИЙ АЛГОРИТМ, МЕТОД ТЕКСТОВОЇ СТЕГАНОГРАФІЇ, НАБІР ДАНИХ, ТЕКСТОВА ПОСЛІДОВНІСТЬ

ABSTRACT

The thesis consists of 56 pages and contains 11 tables, 17 figures and 23 literary sources.

The object of research is the algorithm of text steganography. The subject of the study is the method for generating text stego containers using recurrent neural networks.

The main methods of research in this work are methods of machine learning, as well as to evaluate the algorithms used statistical methods for analyzing the results. Existing methods of concealing information in text sequences were also analyzed.

The result of this work is the algorithm for the automatic generation of steganographic texts, which can be used for the safe transfer and storage of private information. Further research to improve the above algorithm is the study of the use of artificial intelligence for text generation.

RECOVERY NEURAL NETWORK, LANCY MARKOV, STEGANOGRAPHIC ALGORITHM, MATHEW OF TEXT STEGANOGRAPHY, DATA COLLECTION, TEXT SEQUENCE

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів……...7
Вступ…….8
1 Огляд існуючих методів текстової стеганографії…..……...10
1.1 Основні поняття стеганографії……..10
1.2 Загальна характеристика текстової стеганографії……11
1.3 Алгоритми текстової стеганографії……...12
Висновки до розділу 1……..21
2 Використання нейронних мереж для генерування послідовностей……..22
2.1 Особливості роботи глибоких нейронних мереж прямого поширення..22
2.2 Рекурентні нейронні мережі……...26
2.3 Мережі довгострокової/короткострокової пам'яті……..28
2.4 Двоспрямовані рекурентні нейронні мережі……30
Висновки до розділу 2……..32
3 Побудова архітектури нейронної мережі та стеганографічного алгоритму.33
3.1 Опис наборів даних…….33
3.2 Представлення даних……...37
3.3 Архітектура нейронної мережі……..40
3.4 Стеганографічний алгоритм……...41
Висновки до розділу 3……..43
4 Дослідження отриманних результатів……..44
4.1 Аналіз результатів навчання word2vec моделі……...44
4.2 Аналіз результатів навчання рекурентної мережі……45
4.3 Аналіз стеганографічного алгоритму……47
Висновки до розділу 4……..54
Висновки……55
Перелік джерел посилань..……57
Додаток А Схема архітектури нейронної мережі……...60

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

OCR - optical character recognition(оптичне розпізнавання символів)

CFG – context free grammar(набір правил перезапису (або підстановок), що використовуються для створення шаблонів рядків)

RNN – recurrent neural network(рекурентна нейронна мережа)

LSTM - Long short-term memory(мережа довгострокової, короткострокової пам’яті)

CBOW - Continuous Bag of Words (текстова модель «мішку слів»)

ER - Embedding Rate(якість вбудовування)

ВСТУП

Актуальність роботи. В наш час у зв’язку з розвитком технологій передачі інформації проблема інформаційної безпеки стає актуальною як ніколи. Кожного дня за допомогою мережі інтернет передається і оброблюється величезна кількість інформації, що сприяє росту числа витоків особистої інформації. Згідно з законів кожна людина має право на таємницю листування і недоторканність приватного життя, проте методів забезпечення цих прав не так багато. Криптографія і стеганографія – засоби, за допомогою яких можна безпечно передавати дані. За допомогою криптографічних методів змінюють саме повідомлення таким чином, щоб неможливо було, не знаючи спеціального секрету розшифрувати зміст повідомлення. Обмеження криптографії полягає в тому, що третя сторона завжди знає про зв'язок через незрозумілу природу повідомлення. Стеганографія долає це обмеження, приховуючи повідомлення в нейтральному об'єкті, який не викликає підозр. Таким чином за допомогою стеганографії приховується сам факт передачі повідомлення.

Текст є найбільш широко використовуваним інформаційним носієм у повсякденному житті людей, тому використання тексту як носія для приховування інформації має велике наукове значення і практичне використання. Існує багато методів, які змінюють вхідні текстові дані, проте проблема генерації тексту схожого на людську мову, в якому приховано вхідні дані, ще не має прийнятного рішення.

Мета дослідження. Генерація тексту схожого на людську мову з прихованням вбудованих даних, за допомогою нейронної мережі.

Завдання роботи.

1. Вивчення існуючих алгоритмів текстової стеганографії.
2. Аналіз існуючих архітектур нейронних мереж та предметних областей, де вони застосовуються.
3. Вибір, аналіз, попередня обробка та представлення даних для тренування нейронної мережі.

4. Розробка архітектури нейронної мережі для генерування текстових послідовностей.

5. Розробка стеганографічного алгоритму.

6. Практична реалізація алгоритму моделі нейронної мережі і стеганографічного алгоритму

7. Аналіз і тести отриманих результатів

Об'єкт дослідження. Алгоритми текстової стеганографії.

Предмет дослідження. Генерування текстових стегоконтейнерів за допомогою нейронних мереж.

Методи дослідження. В роботі розглянуто методи машинного навчання, статистичні та емпіричні методи.

Наукова новизна. Використання рекурентної нейронної мережі для отримання текстового стегоконтейнера, текст якого копіює людську мову.

Практичне значення. Реалізація програми, яка автоматично генерує текстові послідовності з прихованням вбудованих даних.

Отримані результати можна використовувати для подальших досліджень і покращення методів стеганографії, які використовують в якості контейнера автоматично згенерований текст.

1 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ТЕКСТОВОЇ СТЕГАНОГРАФІЇ

1.1 Основні поняття стеганографії

Стеганографія - це мистецтво приховування інформації та приховування наявності вбудованої інформації. Вона часто служить кращим способом захисту повідомлення, ніж криптографія, яка лише приховує зміст повідомлення, а не існування повідомлення. Початкове повідомлення приховується всередині носія так, що зміни, які відбулися в носії, не можна помітити.

Користувачам мережі інтернет часто потрібно зберігати, відправляти або отримувати приватну інформацію. Найбільш поширений спосіб зробити це - перетворити дані в іншу форму. Отримані дані можуть бути зрозумілі лише тим, хто знає, як повернути їх в початкову форму. Цей метод захисту інформації називається шифруванням. Основним недоліком шифрування є те, що існування секретних даних не приховується. Дані, які були зашифровані, хоча і не читаються, все ще існують як дані, які за певний час можна розсекретити. Рішенням цієї проблеми є стеганографія - стародавнє мистецтво приховування повідомлень таким чином, щоб їх не можливо було виявити. Метою стеганографії є приховування існування повідомлення, в той час як криптографія перетворює повідомлення так, щоб його не можна було зрозуміти.

Перед винаходом цифрових засобів використовувалися традиційні методи надсилання або отримання повідомлень. Для повідомлень, де конфіденційність була першочерговою, шляхи реалізації безпеки були наступними:

1. Вибір посланника, здатного надійно передавати повідомлення.

2. Запис повідомлення, використовуючи такі позначення, щоб фактичне значення повідомлення не можливо було зрозуміти.

3. Приховання повідомлення так, щоб навіть його присутність неможливо було передбачити. [5, 6].

У сучасній стеганографії алгоритм є досить схожим, проте більшість понять стали стандартними і набули стану термінів:

- вбудовані дані - повідомлення, яке потрібно приховати;
- стегоконтейнер - середовище, яке використовується для приховування вбудованих даних, наприклад текст, зображення, аудіо чи відео файл;
- стегоключ – ключ, який використовувається для приховання вбудованих даних в стегоконтейнері;
- стегооб’єкт - це об'єкт, який ми отримуємо після приховування вбудованих даних в контейнері;
- стегоканал - канал, по якому передається стегооб’єкт [7, 8].

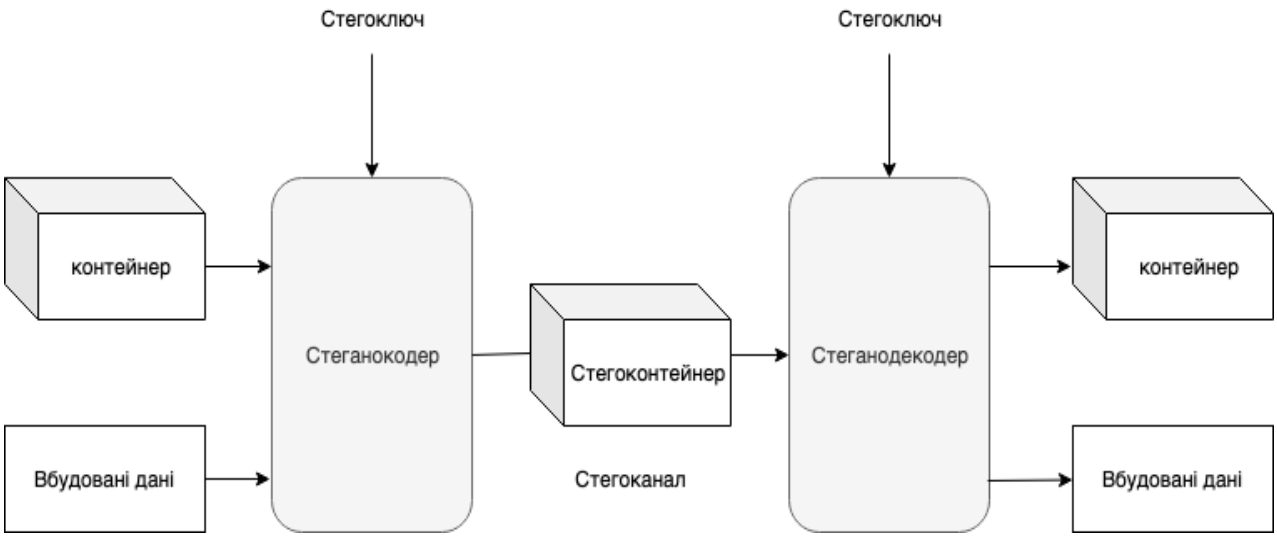


Рисунок 1.1 – Загальна модель стегосистеми

1.2 Загальна характеристика текстової стеганографії

Стеганографію можна поділити на текстову стеганографію, стеганографію зображень, аудіо та відео стеганографію, залежно від типу контейнеру, що використовується для вбудовування секретних даних. Текст дуже широко використовується при передачі даних, тому текстові стеганограми, якщо вони добре згенеровані, досить важко визначити. Однак, у порівнянні з зображенням і аудіо, тексти мають більш високий ступінь кодування

інформації, що призводить до дефіциту надлишкової інформації, та робить досить складним використання тексту як носія для приховування інформації.

Алгоритми текстової стеганографії загалом можна розділити на дві великі сім'ї: методи засновані на форматуванні, і методи засновані на контенті[3].

- *Методи засновані на форматуванні*

Включають зміну фізичного представлення тексту для приховування інформації. Цей метод має певні недоліки. Якщо стегофайл відкриється текстовим процесором, то будуть виявлені помилки та додаткові пробіли. Змінені розміри шрифтів можуть викликати підозру у читача. Крім того, якщо оригінальний відкритий текст доступний, порівняння цього відкритого тексту з підозрюваним стеганографічним текстом зробить змінені частини тексту досить видимими.

- *Методи засновані на контенті*

Щоб уникнути порівняння з відомим відкритим текстом, стеганографи часто вдаються до створення власних текстових контейнерів. Одним з методів є приховування інформації у випадковій послідовності символів. Інші методи використовують статистичні властивості мови, для того щоб, згенерований текстовий стегоконтейнер був максимально схожий на людську мову.

1.3 Алгоритми текстової стеганографії

- Метод лінійного зсуву

У цьому методі секретне повідомлення приховано вертикальним зміщенням рядків тексту. Позначена лінія має дві лінії управління по одній з обох сторін для виявлення напрямку руху позначеної лінії. Щоб приховати біт 0, рядок зсувається вгору, і щоб приховати біт 1, лінія зміщена вниз. Визначення зміщення лінії вгору або вниз здійснюється шляхом вимірювання відстані центроїда від позначеної лінії та її ліній керування. Якщо текст переписано або використовується програма розпізнавання символів (OCR),

прихована інформація буде знищена. Крім того, відстані можна спостерігати за допомогою спеціальних інструментів оцінки відстані[11].

- Метод зсуву слів

У цьому способі секретне повідомлення приховано шляхом зсуву слів горизонтально, тобто ліворуч або праворуч, щоб представляти біт 0 або 1 відповідно[3,11]. Зсув слів виявляється за допомогою методу кореляції. Цей метод можна ідентифікувати менше, оскільки зміна відстані між словами для заповнення рядка досить поширена. Але якщо хтось знає алгоритм відстаней, він може порівнювати стеготекст з алгоритмом і отримувати прихований вміст, використовуючи різницю. Крім того, переписування або використання програм OCR знищує приховану інформацію.

- Синтаксичний метод

Ця методика використовує знаки пунктуації, такі як крапка (.), кома (,) і т.п., щоб приховати біти 0 і 1. Але проблема з цим методом полягає в тому, що вона вимагає ідентифікації правильних місць для вставки знаків пунктуації. Тому слід дотримуватися обережності у використанні цього методу, оскільки читачі можуть помітити неправильне використання розділових знаків[11,3].

- White steg метод

Цей метод використовує пробіли для приховування секретного повідомлення. Існує три способи приховування даних за допомогою пробілів. В першому використовується інтервал між реченнями. Ми розміщуємо один пробіл, щоб приховати біт 0 і два пробіли, щоб приховати біт 1. В другому використовується кінець рядка, кодування така ж саме. Третя техніка використовує простір між словами. Але непослідовне використання пробілу не є прозорим[12].

- Метод спам-тексту

Для нього використовуються файли HTML і XML для приховування бітів. Якщо є різні початкові і закриваючі теги, інтерпретується біт 0, і якщо для відкриття і закриття використовується один тег, то інтерпретується біт 1. В

іншій техніці, біт 0 представлений відсутністю пробілу в тезі, а біт 1 - розміщенням пробілу всередині тега[13].

- Метод кодування особливостей

У методі кодуванні особливостей секретне повідомлення приховано, змінюючи одну або кілька особливостей тексту. Синтаксичний аналізатор вивчає документ і вибирає всі функції, які він може використовувати для приховування інформації. Наприклад, точку в літерах і і j можна зміщувати або змінювати висоту букв b, d, h[11,13]. Недолік цього методу полягає в тому, що якщо використовується програма розпізнавання, то прихований вміст буде знищено.

- Метод крикетного матчу

У цьому методі дані приховані в системі показників крикету, попередньо додаючи нульовий нуль перед числом, щоб представити біт 1 і залишивши число таким, як воно є, щоб представити біт 0[3].

- Метод змінної довжини слова

У цьому методі слова, які генерує людина або машина повинні відповідати довжині, яку визначить програма - зазвичай в одне слово кодується два біти інформації з вхідних даних[14]. Наприклад, слова довжиною в 4 і 8 символів можуть означати комбінацію біт "00", довжиною в 5 і 9 - "01", 6 і 10 - "10", 7 і 11 букв - "11". Слова, коротше 4 і довше 11 букв, можна вставляти будь-де для лексичного і граматичного зв'язку слів у реченні - програма-декодер буде просто ігнорувати їх.

- Метод першої літери

Цей метод зазвичай кодує досить велику кількість інформації в одному слові: зазвичай це 3 або 4 біта. Програма-помічник в цьому методі накладає обмеження вже не на довжину слова, а на першу (можна на другу) букву[14]. Зазвичай одну й ту ж саму комбінацію можуть кодувати кілька букв, наприклад, комбінацію "101" позначають слова, що починаються з "А", "Г" або "Т". Це дає досить велику свободу тому, хто генерує текст.

- Метод заснований на синонімах мови

Проведені досліди виявили, що для англійської мови середня кількість синонімів в одній підмножині дорівнює 2.56. Мінімальна кількість синонімів в множині дорівнює 2, а максимальна 13. В якості прикладу розглянемо підмножину S0:{ “думка”, “погляд”, “уявлення”, “міркування”}. В наведеній послідовності кожне слово має єдине смислове значення, що дозволяє закодувати кожне слово своїм унікальним кодом, наприклад думка = «00», погляд = «01», уявлення = «10», міркування = «11». Подібне кодування дозволяє обирати одне з чотирьох слів, в залежності від 2х біт секретної послідовності. Треба зауважити, що в такому разі семантика(змістове наповнення) повідомлення не зміниться, не залежно від того, яке слово буде обране.

Для того щоб приховати і потім відновити секретну послідовність, відправник і отримувач повинні мати однаковий словник синонімів.

Процедура приховання:

1) відправник обирає текстовий файл, який буде використовуватись як контейнер;
2) послідовне аналізує текст цього файлу, поки не знайде слово з словнику, яка має N синонімів;
3) обраховує цілу частину від log2(N). Це кількість біт, які можна закодувати вибравши одне із слів з словнику синонімів;
4) повторює дії 2-3 доки не буде приховану усю послідовність.

Аналогічні дії виконує той, хто хоче відновити секретну послідовність.

1) Так само послідовно аналізує текст файлу, поки не знайде слово з словнику.
2) З словнику отримує відображення слова в бітову послідовність.
3) Повторює дії 1-2 доки не дійде до кінця текстового файлу

Проте якщо слово має декілька значень подібне кодування виявляється неможливим. Також неможливе подібне кодування якщо один з синонімів складається з двох(або більше) слів, які відокремлюються пробілом.

Також далеко не завжди кількість синонімів дорівнює 2^k. Наприклад, розглянемо 3 множини, довжиною 3, 3, 5. Використання таких трьох множин, згідно з обрахуванням цілої частину від log2(N), загалом дозволяє закодувати 4 біта інформацій. В той час як потужність цих трьох множин дорівнює 3*3*5 = 45 станів, або трохи більше ніж 5 біт інформації.

Для того щоб збільшити об'єм інформації, яку буде приховано в контейнері використовують систему обчислення зі зміщеною основою. В цьому випадку працюють не з однією підмножиною, а з групою підмножин синонімів. Наприклад для трьох множин, які було розглянуто вище в такому разу можна буде закодувати [log2(45)] = 5 біт інформації. Для цього використовують код зі зміщеною основою з трьох цифр, в якому перша і третя цифри можуть приймати значення від 0 до 2x, а 2 цифра приймає значення від 0 до 4x. Нижче наведено таблицю для переводу з двійкового коду до коду зі зміщеною основою[14].

Таблиця 1.1 – Відповідність між двійковим кодом і кодом зі зміщеною основою

Двійковий код	00000	00001	00010	…	01110	01111	…	11111
Код зі зміщеною основою	000	001	002	…	042	100	…	201

- Метод мімікрії

Метод мімікрії генерує змістовний текст, використовуючи синтаксис, який описано в CFG і приховує інформацію, обираючи із CFG певні фрази і слова. CFG – один із способів опису мови, який складається з статичних слів, фраз та вузлів, де може бути прийняте рішення, яке слово чи фразу вставляти наступним. Мімікрія створює бінарне дерево, яке засноване на можливостях CFG, і обираючи ті з листків дерева, які кодують потрібний біт[14]. Наприклад потрібно приховати наступні біти – «0101», використовуючи наступне бінарне дерево:

Старт -> іменник | дієслово

Іменник -> Олег | Тарас

Дієслово -> поїхав у відрядження **куди** | поїхав подорожувати **куди**

Куди -> в **напрям** Корею | в **напрям** Мінесоту

Напрям -> північну | південну

Таблиця 1.2 – Формування тексту методом мімікрії

Шаг	Відповіді, отримані в процесі	Прихований біт	Вибір процедури
1	**Старт**	-	**іменник	дієслово**
2	**іменник	дієслово**	1	**Іменник** -> Тарас
3	Тарас **дієслово**	0	**Дієслово** -> поїхав у відрядження **куди**
4	Тарас поїхав у відрядження **куди**	1	Куди -> в **напрям** Мінесоту
5	Тарас поїхав у відрядження в **напрям** Мінесоту	0	**Напрям** -> північну

Таким чином отримуємо речення: Тарас поїхав у відрядження в північну Мінесоту.

Недоліками цього методу є: неможливість шифрування великої кількості даних, низька продуктивність і невисока скритність.

- Метод заснований на ланцюгу Маркова

У галузі обробки природної мови зазвичай використовують статистичну модель мови для моделювання текстів. Мовна модель є розподілом ймовірностей над послідовностями слів, вона може бути виражена наступною формулою:

$$p(S) = p(w_1, w_2, w_3, \ldots, w_n) = p(w_1) * p(w_2 | w_1) * \ldots * p(w_n | w_1, w_2, w_3, \ldots, w_{n-1}), \quad (1.1)$$

де S позначає ціле речення довжиною n, а w_i позначає і-те слово в ньому. p(S) позначає ймовірність всього речення. Воно фактично складається з

добутку n-умовних ймовірностей, кожна з умовних ймовірностей обчислює розподіл ймовірності n-го слова, коли наведено перші n-1 слова, тобто $p(w_n| w_1, w_2, w_3, \ldots, w_{n-1})$. Тому для автоматичної генерації високоякісних текстів, необхідно отримати гарну оцінку статистичної мовної моделі набору навчальних вибірок[4].

В теорії ймовірностей ланцюг Маркова є стохастичною моделлю, що описує послідовність можливих подій, в яких ймовірність кожної події залежить тільки від стану, в попередній події. Модель ланцюга Маркова підходить для моделювання сигналів часових рядів. Наприклад, припустимо, що існує простір величин $X = \{x_1, x_2, \ldots, x_m\}$, та $Q = \{q_1, q_2, \ldots, q_n\}$, стохастична змінна послідовність, значення якої відбираються з X. Для зручності наведеного опису позначимо значення t-го стану як x^t, такого що $q_t = x^t$, $x^t \in X$.

Якщо ми вважаємо, що значення стану в кожному моменті послідовності пов'язане зі станом усіх попередніх моментів, тобто $p(q_t | q_1, q_2, q_3, \ldots, q_{t-1})$ то ланцюгова модель Маркова виглядає наступним чином:

$$P(q_t = x^t) = f(P(q_{t-1} = x^{t-1}), P(q_{t-2} = x^{t-2}), \ldots, P(q_1 = x^1)), \quad (1.2)$$

$$\sum_{i=1}^{m} P(q_t = x_t) = 1, \forall x_i \in X$$

Де f - функція передачі ймовірностей. Тоді ймовірність всієї послідовності може бути виражена наступним чином:

$$p(Q) = p(q_1, q_2, q_3, \ldots, q_n) = p(q_1 = x^1)\, p(q_2 = x^2) \ldots p(q_n = x^n) = p(q_1) * p(q_2|q_1) *\ldots* p(q_n| q_1, q_2, q_3, \ldots, q_{n-1}) \quad (1.3)$$

Порівнюючи формулу (1.1) з формулою (1.3), можна побачити, що якщо розглядати сигнал x_i в кожний момент часу у формулі (3), як і-е слово в реченні, то він може точно представляти умовний розподіл ймовірності кожного слова в текст, який представляється як $p(w_n| w_1, w_2, w_3, \ldots, w_{n-1})$, а потім він може цілком моделювати статистичну модель мови тексту. Саме через цю спільність модель ланцюга Маркова дуже підходить для моделювання тексту і широко використовується в галузі обробки природної мови, особливо в області автоматичного формування тексту.

Як правило, в реальних ситуаціях вплив сигналу в кожний момент часу в послідовності на наступний сигнал обмежений, тобто існує «довжина» впливу, і за її межами не буде продовжуватись вплив на наступний сигнал. В такому випадку припускають, що для сигналу часового ряду значення кожного сигналу часу впливає тільки на кілька перших наступних моментів. Якщо на величину сигналу в кожний момент впливають тільки сигнали попередніх m моментів, марківську модель називають m-порядковою і вона може бути виражено наступним чином:

$P(Q_t=x_t| Q_{t-1}=x_{t-1}, Q_{t-2}=x_{t-2}, \ldots, Q_1=x_1) = P(Q_t=x_t| Q_{t-1}=x_{t-1}, Q_{t-2}=x_{t-2}, \ldots, Q_{t-m}=x_{t-m})$, де $n > t > m$ (1.4)

При використанні моделі Маркова для автоматичного генерування тексту, для великого навчального набору даних, що містить багато речень, спочатку будується великий словник D, який містить всі слова, що з'являлися в навчальному наборі, тобто $D = \{word_{D1}, word_{D2}, word_{D3}, \ldots, word_{Dn}\}$, де $word_{Di}$ позначає i-те слово в словнику і N- загальна кількість слів в словнику. Словник D відповідає значенням простору X, описаному вище. Як згадувалось раніше, кожне речення S можна розглядати як послідовний сигнал i-те слово в S може бути представлено як сигнал в момент часу i, тобто:

$$S = \{word_{S1}, word_{S2}, word_{S3}, \ldots, word_{SL}\}, \quad (1.5)$$

де $word_{si} \in D$, $word_{si}$ вказує i-те слово у реченні S і L його довжина. У процесі автоматичного формування тексту необхідно обчислити ймовірність переходу кожного слова. Для марківської ланцюгової моделі за теоремою великих чисел зазвичай використовується частота кожної фрази в наборі даних для наближення ймовірності. Наприклад, для моделі ланцюга Маркова другого порядку формула розрахунку виглядає так:

$$P(word_{s_n} = word_{D_i} | word_{s_{n-2}}, word_{s_{n-1}}) \approx \frac{count(word_{s_{n-2}}, word_{s_{n-1}}, word_{D_i})}{count(word_{s_{n-2}}, word_{s_{n-1}})}, \quad (1.6)$$

де $\sum_{i=1}^{N} p(word_{s_n} = word_{D_i} | word_{s_{n-2}}, word_{s_{n-1}}) = 1$

$count(word_{s_{n-2}}, word_{s_{n-1}}, word_{D_i})$ - кількість появ цієї фрази в наборі даних.

Якщо не потрібно вбудовувати інформацію, а задачею є просто генерування природного тексту, зазвичай вибирають слово з найбільшою ймовірністю як вихідне на кожній ітерації.

Для приховування інформації існує декілька алгоритмів, один з них заснований на кодуванні Гаффмана. Існує дві основні частини цього алгоритму кодування:

1) Створення дерева Гаффмана з символів вводу.

2) Ставлення у відповідність символам дерева Гаффмана кодових значень.

Кроки для створення дерева Гаффмана:

Вхідні дані: масив унікальних символів разом з частотою їх виникнення

Вихід: Дерево Гаффмана.

1. Створення вузла(листа) для кожного унікального символу та побудова мінімальної «купи» всіх вузлів(листів).

2. Витягання двох вузлів з мінімальною частотою з «купи».

3. Створення нового внутрішнього вузла з частотою, що дорівнює сумі частот двох вузлів. Призначення першого витягнутого вузла – лівою дитиною, а іншого витягнутого вузла - правою дитиною. Додавання цього вузла до «купи».

4. Повторення кроків 2 і 3 до тих пір, поки в «купі» не буде лише одного вузла. Залишившийся вузел - кореневий вузел. Дерево завершено[4].

Нижче на рисунку 1.2 можна побачити як в бітову послідовність за допомогою кодування Гаффмана ставиться у відповідність послідовність слів, яка поетапно генерується марківською ланцюговою моделлю.

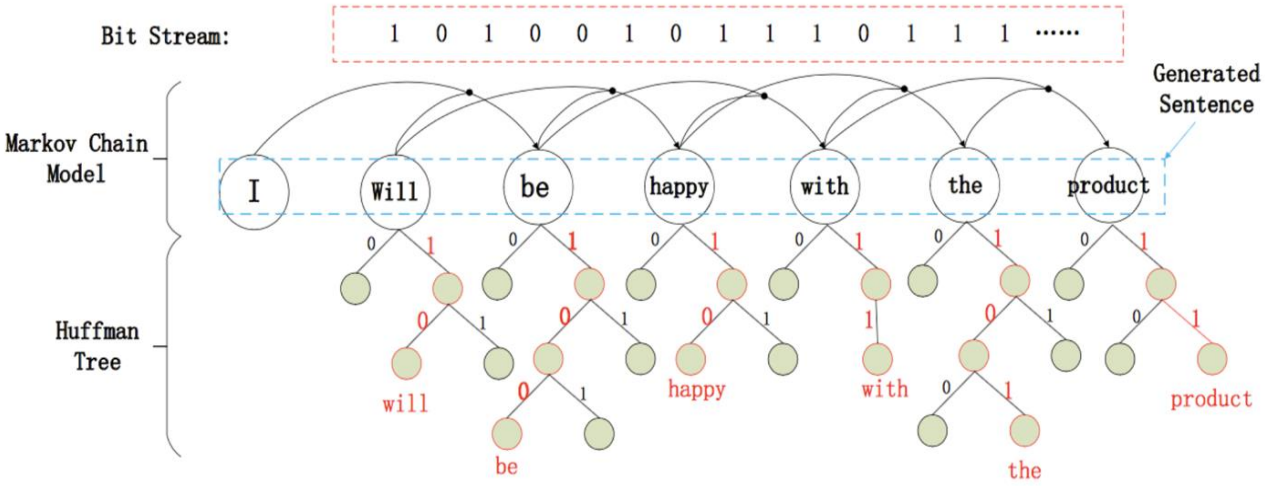


Рисунок 1.2 – Загальна схема алгоритму, заснованому на Марківському ланцюгу і кодуванні Гаффмана

Висновки до розділу 1

В даному розділі було розглянуто загальну схему того, як працює стеганосистема, а також введено терміни, якими оперує галузь стеганографії. Подана загальна характеристика стеганографії текстів, а також введено класифікацію.

Досліджено існуючи алгоритми текстової стеганографії. Описано загальні засади на яких вони засновані, а також переваги та недоліки кожного з алгоритмів. Схему роботу алгоритмів, які є найбільш актуальними і слугували основою для створення власного методу розглянуто більш детально.

2 ВИКОРИСТАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ ГЕНЕРУВАННЯ ПОСЛІДОВНОСТЕЙ

2.1 Особливості роботи глибоких нейронних мереж прямого поширення

Сучасне глибоке навчання пропонує розвинену інфраструктуру навчання з вчителем. Завдяки додаванню додаткових шарів і блоків в межах одного шару, глибока мережа може представляти все більш і більш складні функції. Більшість завдань, що зводяться до відображення вхідного вектора на вихідний, з котрими людина справляється легко і невимушено, може бути вирішено методами глибокого навчання при наявності досить великих моделей і наборів позначених прикладів. Інші завдання, які не можна описати як асоціювання одного вектора з іншим, настільки важкі, що людині потрібен час для їх вирішення, і поки не піддаються глибокому навчанню.

Глибокі мережі прямого поширення, які називають також нейронними мережами прямого поширення, або багатошаровими перцептронома - найтиповіші приклади моделей глибокого навчання. Мета мережі прямого поширення - апроксимувати деяку функцію f*. Наприклад, в разі класифікатора y = f * (x) – відображення входу x в категорію y. Мережа прямого поширення визначає відображення y = f (x; θ) і шляхом навчання знаходить значення параметрів θ, що дають найкращу апроксимацію

Слова «пряме поширення» означають, що поширення інформації починається з x, проходить через проміжні обчислення, необхідні для визначення f, і закінчується виходом y. Не існує зворотніх зв'язків, за якими виходи моделі подаються на її вхід. Узагальнені нейронні мережі, що включають такі зворотні зв'язки, називаються рекурентними[15].

Для початку розглянемо схему, як влаштовано перцептрон – найпростішу схему одного штучного нейрону яку було запропоновано ще в 50х -60х роках минулого сторіччя. На рисунку 2.1 x1,x2, x3 – позначають вхідні дані, а на виході ми отримуємо єдине значення output.

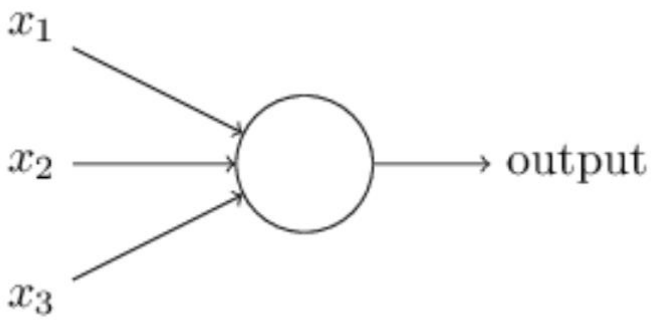


Рисунок 2.1 – Схема одного штучного нейрону

В наведеному вище рисунку перцептрон має три входи, x1,x2, x3. В загальному випадку кількість входів може бути будь-якою. Обчислення результатів вводиться за допомогою ваг, w1, w2 ,. . ., реальних чисел, що можуть інтерпретуватися як значимість вхідних даних. Вихід нейронів, «0» чи «1», залежить вид того, чи значення виразу $\sum_j w_j x_j$ перевищує порогове. В формульному вигляді:

$$output = \begin{cases} 0 \; if \; \sum_j w_j x_j \; \leq threshold \\ 1 \; if \; \sum_j w_j x_j \; \geq \; threshold \end{cases} \tag{2.1}$$

Щоб спростити спосіб опису перцептронів у виразі $\sum_j w_j x_j \; \geq \; threshold$ зазвичай замінюють запис $\sum_j w_j x_j$ таким чином: $w \cdot x = \sum_j w_j x_j$, де w і x - вектори, ваг і входів відповідно. Також зазвичай значення порогу переносять в іншу сторону нерівності і замінюють її на зміщення перцептрону, b ≡ - $threshold$. З використанням зміщення, а не порогу, формула (2.1) набуває виду:

$$output = \begin{cases} 0 \; if \; w \cdot x + b \; \leq 0 \\ 1 \; if \; w \cdot x + b \; \leq 0 \end{cases} \tag{2.2}$$

Також сучасні нейрони замість того, щоб набувати на виході лише значення 0 або 1, замінюють порогову функцію на інші так звані функції активації. Одним з прикладів є функція сигмоїду. Розрахунок виходу отримується за такою формулою σ(wx + b), де σ називається сигмоподібною функцією, і визначається:

$$\sigma(z) \equiv \frac{1}{1+e^{-z}} \tag{2.3}$$

Якщо замість Z підставити отриманий вираз для входів x1, x2, x3, … і значень ваги w1, w2 , ... отримаємо:

$$\sigma(z) \equiv \frac{1}{1+e^{-wx-b)}} \quad (2.4)$$

Використання на виході функції по типу сигмоїда дає можливість згладити значення виходу, для того щоб при навчанні нейронної мережі невелика зміна вектору зміщення або ваг не так сильно впливала на вихід. Завдяки цьому тренування нейронної мережі дає кращі результати[15]. Для порівняння графіки сигмоїди і функції кроку представлено нижче на рисунках 2.2 і 2.3.

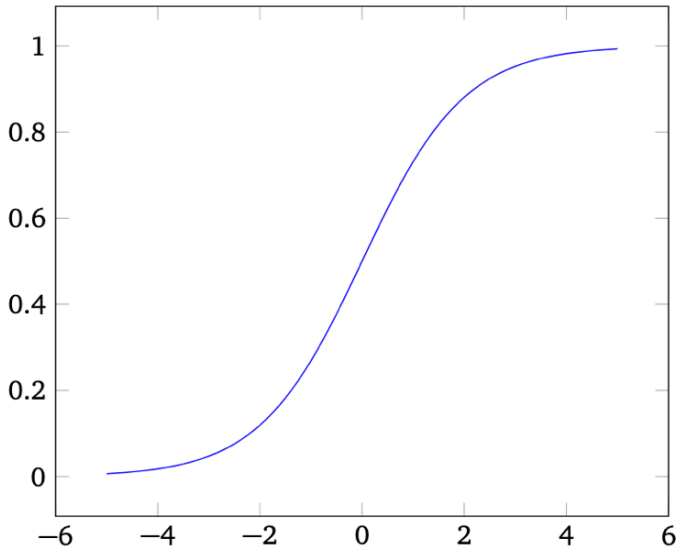


Рисунок 2.2 – Сигмоїдальна функція

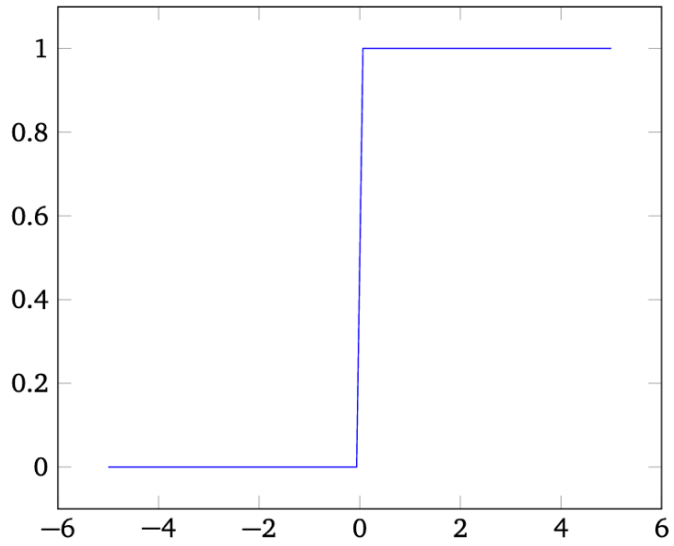


Рисунок 2.3 – Функція кроку

Для побудови нейронної мережі, окремі нейрони об’єднують в шари нейронів, ці шари розміщують послідовно один за одним. Крайній лівий шар у цій мережі називається вхідним шаром, а нейрони всередині шару називаються вхідними нейронами. Крайній правий вихідний шар містить вихідні нейрони. Середній шар називається прихованим шаром, оскільки нейрони в цьому шарі не є ні входами, ні виходами мережі. Схему нейронної мережі ілюструє рисунок 2.4. На цьому прикладі мережа має 2 приховані шари і лише один нейрон на виході

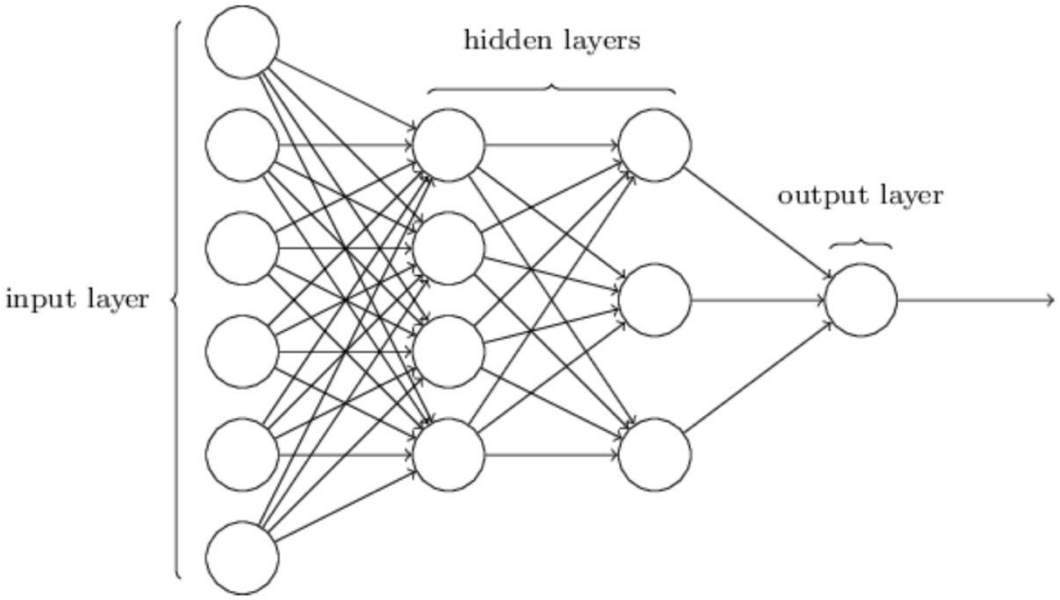


Рисунок 2.4 – Загальна схема нейронної мережі

Глибокими нейронними мережами називають ті, які мають 2 або більше прихованих шарів. Кожен з цих шарів може мати довільну кількість нейронів і на виході використовувати різні функції активації. До того ж крім звичайних цільнозв’язаних шарів існує велика кількість інших, таких як згорткові, рекурентні, відсіювання тощо. Будування архітектури нейронної мережі досить складна задача, бо треба розуміти за допомогою яких шарів буде краще представити вхідні дані, щоб результат отриманий на виході був задовільним.

Іншим важливим аспектом нейронних мереж є те, яким чином відбувається їх тренування, а саме зміна векторів ваги таким чином, щоб вихід мережі співпадав з необхідним результатом. Для цього вводять поняття функції помилки, яка показує як відрізняється отриманий результат від еталонного. Для навчання нейронної мережі використовують алгоритм зворотнього поширення, який складається з таких кроків[16]:

1.На вхідний шар подаються дані для навчання.

2.Послідовно для кожного шару обраховується значення його виходу і застосовується функція активації.

3. Обраховується значення помилки на виході,

4.В зворотній послідовності для кожного шару обчислюється значення помилки і градієнт на який змінюється вектор ваги.

2.2 Рекурентні нейронні мережі

Рекурентні нейронні мережі, або РНМ - це сімейство нейронних мереж для обробки послідовних даних x (1), ..., x (τ). Рекурентна мережа масштабується на набагато довші послідовності, ніж неспеціалізовані нейронні мережі. Більшість рекурентних мереж здатна також обробляти послідовності змінної довжини.

Для переходу від багатошарових мереж до рекурентних скористаємося однією з ранніх ідей машинного навчання і статистичного моделювання: поділ параметрів між різними частинами моделі. Поділ параметрів дозволяє застосувати модель до прикладів різної форми (в даному випадку - довжини) і виконати для них узагальнення. Якби для кожного тимчасового індексу були окремі параметри, то ми не змогли б ні узагальнити модель на довжини послідовностей, що не зустрічалися на етапі навчання, ні розповсюдити статистичну силу на послідовності різної довжини і на різні моменті часу. Такий поділ особливо важливо, якщо деяка частина інформації може зустрічатися в декількох місцях послідовності, що характерно для тестових послідовностей.

В рекурентних нейронних мережах на відміну від звичайних інформація передається не тільки послідовно від одного шару до іншого, а ще й поширюється між нейронами одного шару. Таким чином в мережі реалізується «пам'ять». Це змінює характер її роботи таким чином, що з’являється можливість аналізувати послідовності будь-яких даних, для яких є важливим порядок[2]. На рисунку 2.5 можна бачити, що на кожному циклі роботи внутрішній шар нейронів отримує набір вхідних даних X і інформацію по попередньому стані внутрішнього шару A, на підставі чого генерує відповідь h.

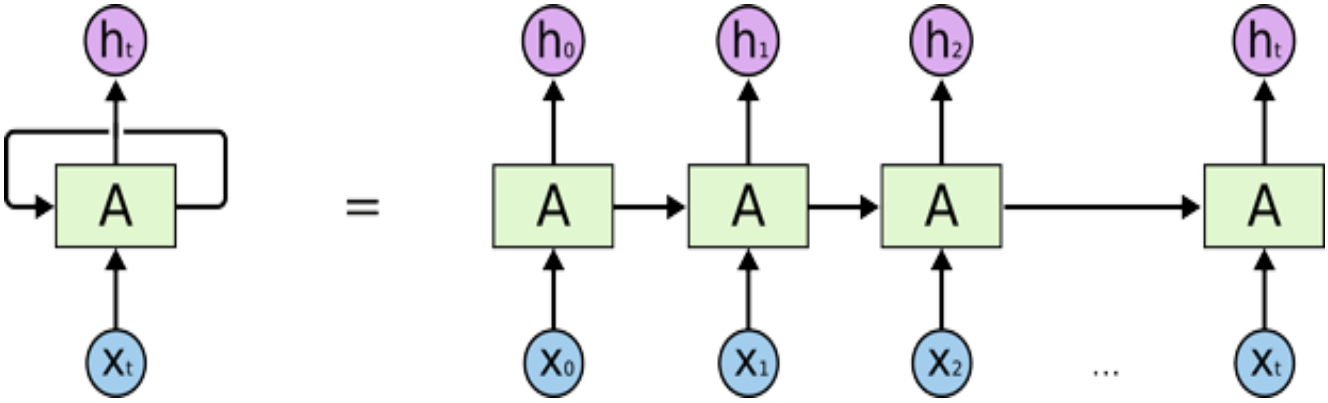


Рисунок 2.5 – Схема одношарової рекурентної нейронної мережі

Обчислення в більшості РНМ можна розкласти на три блоки параметрів і асоційовані з ними перетвореннями:

1) із входу в приховане стан;

2) з попереднього прихованого стану в наступне;

3) з прихованого стану в вихід.

З огляду на ці перетворення глибоку рекурентну нейронну мережу можна зробити глибокої різними способами. На рисунку 2.6 приведено схеми різних варіацій глибоких нейронних мереж. На рисунку 2.6(А) показано, як прихований рекурентний стан можна розділити на ієрархічно організовані групи. На рисунку 2.6(В) між входом і прихованим станом, між двома прихованими рівнями прихованого стану і між прихованим станом і виходом можна помістити глибше обчислення. Це може подовжити найкоротший шлях, що з'єднує різні часові кроки. На рисунку 2.6(С) показано як ефект подовження шляху можна згладити шляхом додавання прямих зв'язків в мережі.

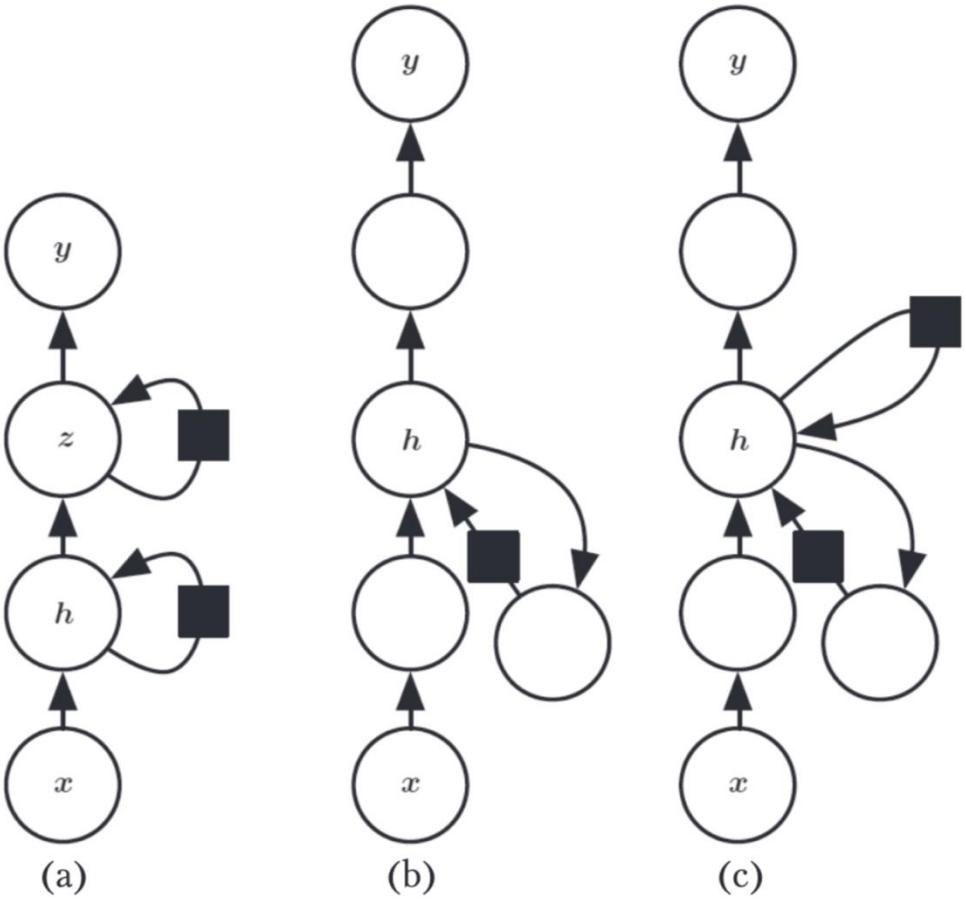


Рисунок 2.6 – Приклади глибоких RNN

Проте в класичних рекурентних мережах виникає проблема навчання довгострокових залежностей. Основна складність полягає в тому, що градієнти, що поширюються через багато шарів, або зникають (в більшості випадків), або починають вибухово рости (рідко, але з великою втратою для оптимізації). Навіть якщо припустити, що при заданих параметрах рекурентна мережа стійка (може зберігати спогади без вибухового зростання градієнтів), все одно залишається проблема призначення довгостроковим залежностям експоненціально менших ваг, ніж короткостроковим[15].

2.3 Мережі довгострокової/короткострокової пам'яті

Проблема зникаючого і вибухового градієнта для РНМ була незалежно виявлена декількома дослідниками. Можна було б сподіватися уникнути її, просто залишаючись в області простору параметрів, де градієнти не зникають і не ростуть вибухово. На жаль, для зберігання «спогадів» способом, стійким

до малих збурень, РНМ повинна увійти в область простору параметрів, де градієнти зникають. Точніше кажучи, якщо модель здатна уявити довгострокові залежності, то абсолютна величина градієнта довгострокової взаємодії експоненціально менше, ніж короткострокового. Це не означає, що мережу взагалі неможливо навчити, просто навчання довгострокових залежностей може зайняти дуже багато часу, тому що сигнал про ці залежності буде замаскований найдрібнішими флуктуаціями, що виникають через короткострокові залежності. Експерименти, описані в роботі Bengio et al. (1994), показують, що на практиці при збільшені протяжності залежностей, які потрібно запам'ятовувати, градієнтна оптимізація стає все важчою, і ймовірність успішно навчити традиційну РНМ методом стохастичного градієнтного спуску швидко спадає до 0, коли довжина послідовностей дорівнює всього 10 або 20.

Для вирішення цієї проблеми використовуються вентильні РНМ, які засновані на ідеї прокладання таких шляхів крізь час, на яких похідні не обнуляються і не спрямовуються різко вгору. Вентильні РНМ узагальнюють ідею блоків з витоком на ваги зв'язків, які можуть змінюватися на кожному часовому кроці[15].

Блоки з витоком дозволяють мережі накопичувати інформацію (наприклад, свідоцтва на користь конкретної ознаки або категорії) протягом тривалого часу. Але іноді корисно, щоб після використання цієї інформації нейронна мережа забула старий стан. Наприклад, якщо послідовність складається з під-послідовностей і ми хочемо, щоб блок з витоком накопичував свідоцтва всередині кожної підпослідовності, то необхідний механізм забування старого стану - скидання його в нуль. І добре б, щоб нейронна мережа навчилася, коли це потрібно робити, не обтяжуючи нас прийняттям рішення. Саме для цього і призначені вентильні РНМ.

Однією з реалізацій вентильної РНМ LSTM-мережа. У LSTM-мережах внутрішні нейрони мають в своєму розпорядженні складну систему воріт, а також концепцією клітинного стану, що і являє собою вид довгострокової

пам'яті. Ворота визначають інформацію, що потрапить в клітинний стан, та інформацію, яка зітреться з нього, таким чином обираючи інформацію яка вплине на результат, який згенерує RNN на даному етапі. На рисунку 2.7 зображено схему шару LSTM, який працює за таким принципом: нейрони у LSTM шарах можуть зчитувати і змінювати стан клітини[10].

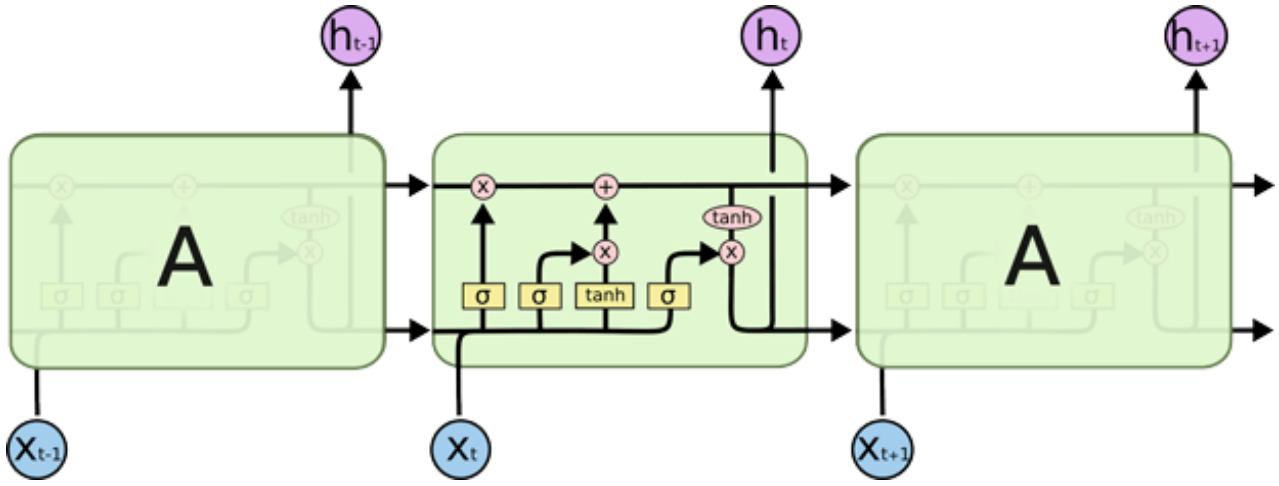


Рисунок 2.7 – Схема LSTM мережі

2.4 Двоспрямовані рекурентні нейронні мережі

Всі розглянуті до сих пір рекурентні мережі мали «казуальну» структуру, тобто на стан в момент t впливає тільки інформація з попередніх моменті часу x_1, ..., x_{t-1} і поточний вхід x_t. Але в багатьох застосунках ми хочемо отримувати передбачення y_t , яке може залежати від всієї вхідної послідовності. Наприклад, в задачі розпізнавання мови правильна інтерпретація поточного звуку як фонеми може залежати від кількох наступних фонем через коартикуляцію і навіть від декількох наступних слів через лінгвістичні залежності між сусідніми словами: якщо акустично допустимі дві інтерпретації поточного слова, то , щоб розрізнити їх, можливо, знадобиться заглянути далеко в майбутнє (або в минуле). Це відноситься також до розпізнавання рукописних текстів і багатьох інших завдань навчання однієї послідовності на основі іншої.

Двоспрямовані рекурентні нейронні мережі були придумані для рішення саме цієї потреби. Як випливає із самої назви, двоспрямовані РНМ є комбінацією РНМ, що рухається вперед у часі (від початку послідовності до її кінця), і РНМ, що рухається в зворотному напрямку. На рис. 2.8 показана типова двоспрямована РНМ; $h^{(t)}$ позначає стан тієї РНМ, що рухається вперед, а $g^{(t)}$ - тієї, що рухається назад. Це дозволяє вихідним блокам $o^{(t)}$ обчислювати представлення, яке залежить як від минулого, так і від майбутнього, але найбільш чутливо до вхідних значень поблизу моменту t; при цьому задавати вікно фіксованого розміру навколо t необов'язково (на відміну від мереж прямого поширення, згорткових мереж і звичайних РНМ з буфером фіксованого розміру)[15].

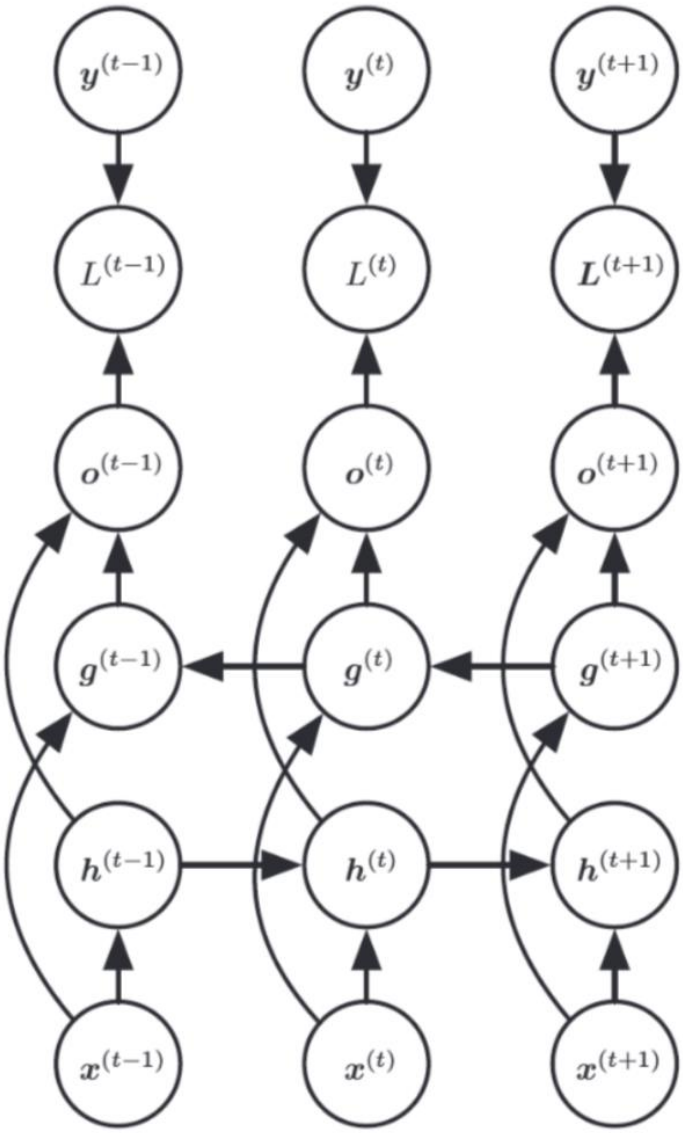


Рисунок 2.8 – Двоспрямована РНМ

Висновки до розділу 2

В даному розділі було розглянуто загальне влаштування нейронної мережі. Яким чином вона працює і що використовує для апроксимації певних функцій.

Також було досліджено один із видів нейронних мереж - рекурентні нейронні мережі, які оброблюють послідовності. Загальний принцип їх роботи і галузі, де їх доцільно використовувати. Проблеми які існують в рекурентних нейронних мережах та методи їх вирішення, а також оптимізацію РНМ для отримання кращих результатів.

3 ПОБУДОВА АРХІТЕКТУРИ НЕЙРОННОЇ МЕРЕЖІ ТА СТЕГАНОГРАФІЧНОГО АЛГОРИТМУ

3.1 Опис наборів даних

Для розгляду було обрано 3 набори даних:

- IMDB Dataset – набір даних містить 50 000 відгуків користувачів на фільми, опис яких є на сайті www.imdb.com. У всій колекції не більше 30 відгуків для будь-якого фільму, тому що відгуки про один фільм часто мають корельовані рейтинги.
- Twitter sentiment140 – нараховує 16000000 текстових «твітів», користувачів мережі twitter. Цей набір даних було створено автоматично з використанням twitter API [1].
- News dataset - містить 143 000 статей з 15 американських публікацій, включаючи New York Times, Breitbart, CNN і та інших.

Попередня обробка даних:

- видалено усі посилання;
- видалено html-теги;
- видалено спеціальні позначення посилань на інших користувачів (Наприклад @user);
- видалено розділові знаки і символи які не входять до латинського алфавіту чи не є цифрами;
- всі тексти було приведено до нижнього регістру;
- за допомогою токенизаторів мови поділено на речення, а кожне речення поділено на окремі слова. Кожне речення, було відформатоване таким чином, щоб його довжина не перевищувала 40 слів;
- Для слів, які зустрічались в тексті 1 раз було застосовано функцію автоматичної орфографічної корекції, якщо ця функція не допомагала усе речення видалялось з набору даних;

На рисунку 3.1 можна бачити, як змінилась кількість символів після обробки для кожного набору.

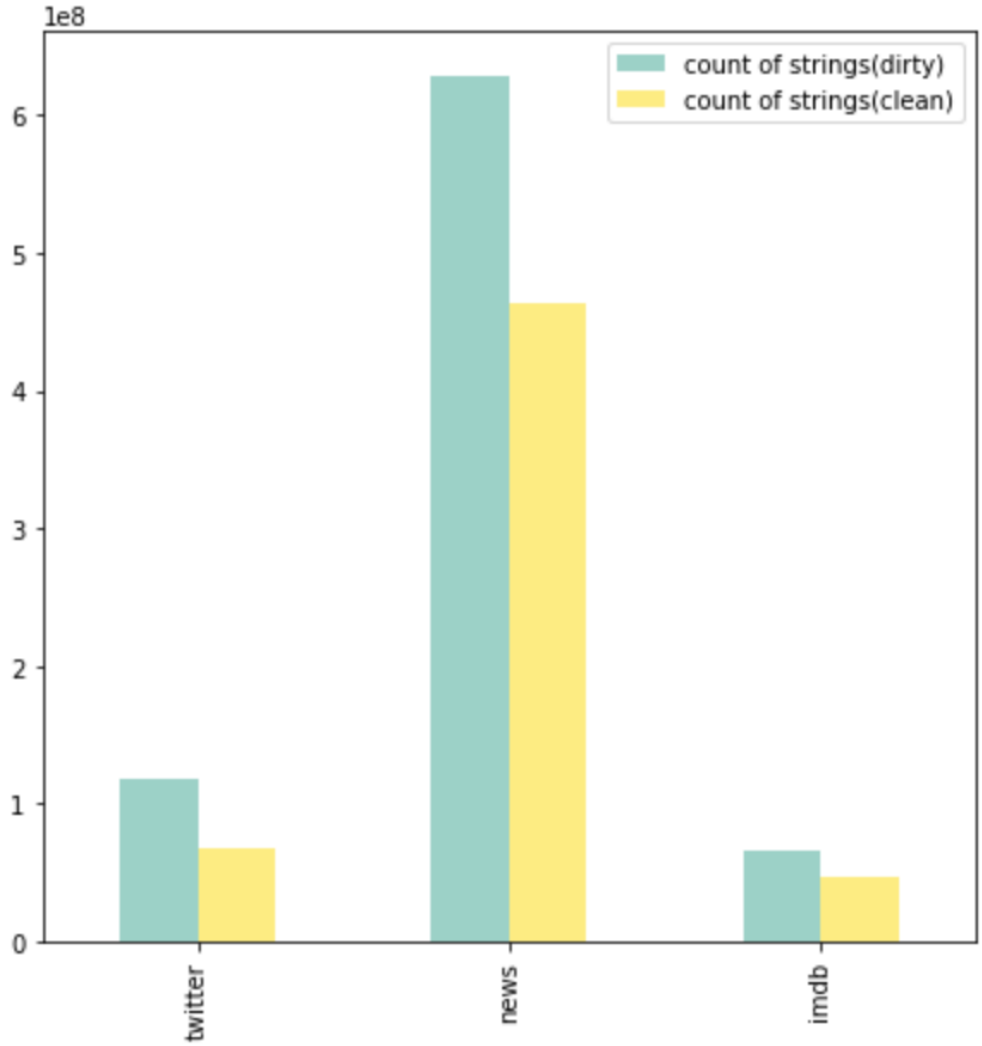
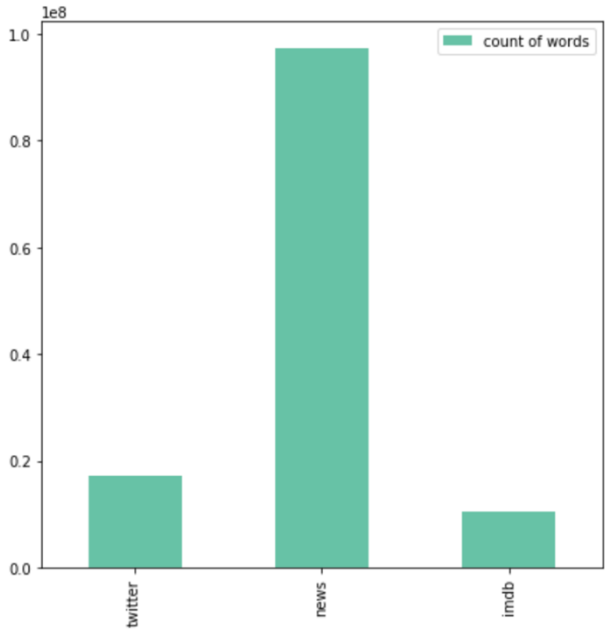
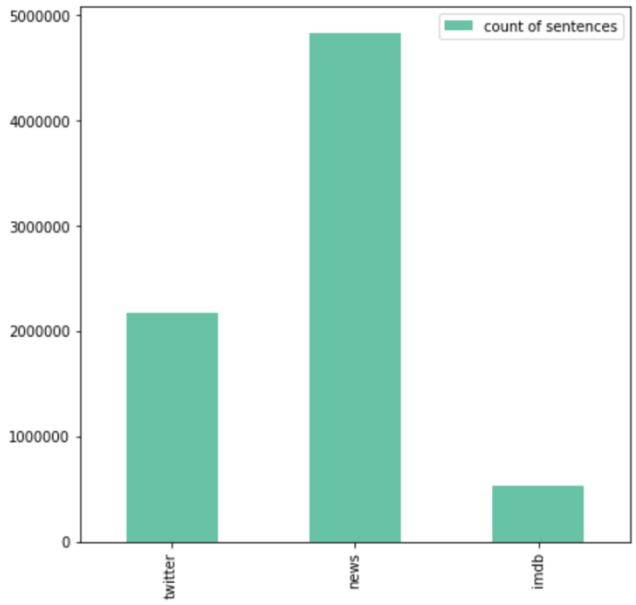


Рисунок 3.1 – Кількість символів на наборах даних.

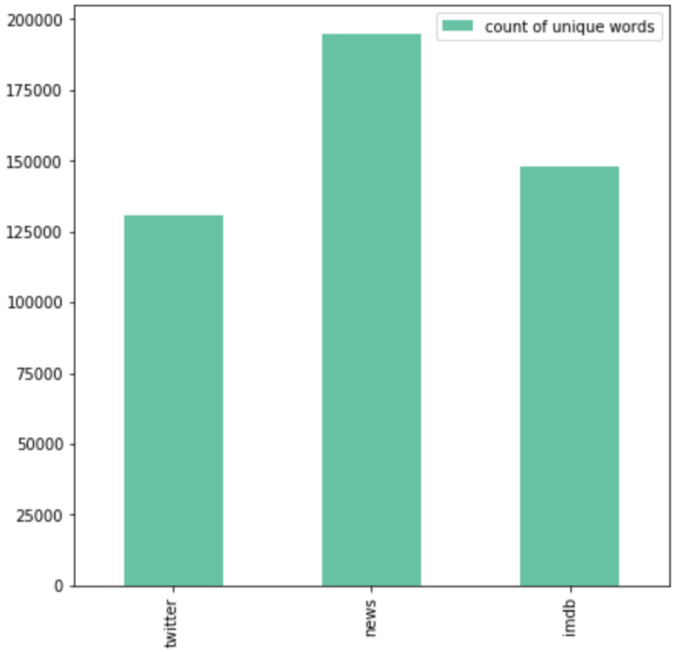
Також для кожного з наборів даних було пораховано загальну кількість слів, кількість унікальних слів і речень. Як можна бачити з рисунку 3.2 кількість слів і речень найбільша для набору даних news", адже він містить найбільшу кількість даних загалом. Кількість унікальних слів набувала найбільшого значення для набору "twitter", проте після орфографічної корекції і видалення незначної кількості речення кількість унікальних слів зменшились більше ніж в 3 рази. Це пояснюється тим, що користувачі мережі часто допускають орфографічні помилки, а також сам набір даних збирався автоматично, тож облікові записи, з яких збирались дані могли бути обліковими записами ботів.



(a) кількість слів



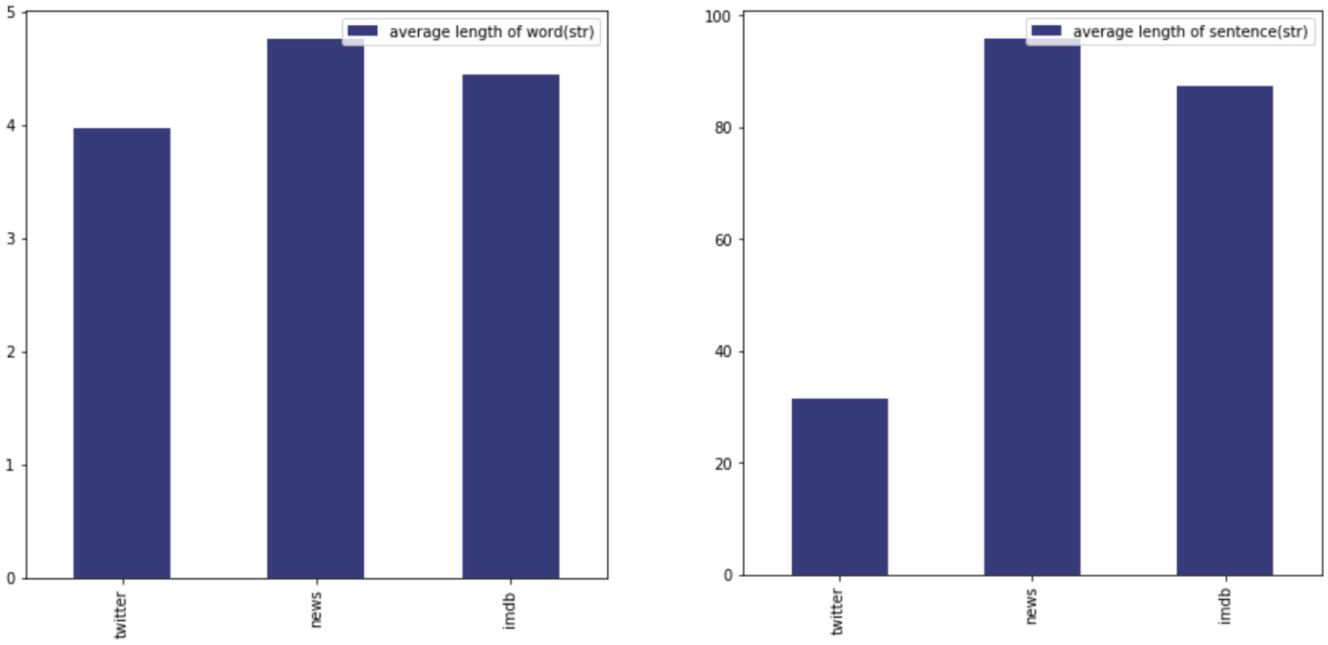
(b) кількість речень



(c) кількість унікальних слів

Рисунок 3.2 – Кількісні характеристики наборів даних

З рисунку 3.3, де представлено усереднені значення довжини речення і слова виміряні в символах можна зробити висновок, що в наборі даних “twitter”, речення майже втричі коротші аніж в інших наборах даних, а також середня довжина одного слова менша приблизно на половину символу. Це пояснюється тим, що у користувачів соціальної мережі обмежена довжина одного «твіта» - 280 символів і їм доводиться стисло висловлювати свої думки.



(a) середня довжина слова (b) середня довжина речення

Рисунок 3.3 – Усереднені характеристики наборів даних

Точні чисельні значення усіх величин з наведених вище графіків представлено в таблиці 4.1.

Таблиця 4.1 – Характеристики наборів даних

	count of strings (dirty)	count of strings (clean)	count of words	count of unique words	count of sentences	average length of sentence (str)	average length of word (str)
Twitter	118544178	68273863	17186326	130799	2173882	31.40	3.97
News	629349137	464359891	97451830	195132	4839065	95.96	4.76
Imdb	65471551	46044728	10352835	148210	526909	87.38	4.44

3.2 Представлення даних

Для того щоб дані можна було використовувати для навчання моделі їх треба представити в векторному вигляді. Найбільш простим представленням є співставлення кожному слову вектору розмірності (1, d), де d - довжина словнику унікальних слів. Для кожного слова вектор матиме вид: (0, 0, ... 0, 1, 0, …, 0), усі позиції крім і-тої заповнені нулями, а на і-тій позиції стоїть одиниця, де і – це індекс порядку цього слова в словнику унікальних слів. Таке представлення називається “one hot encoding”, воно дає можливість генеруючій моделі працювати з текстовим набором даних, проте в той же час воно дає дуже мало інформації про саме слово.

Більш вдалим представленням є “word2vec”, яке кожному слову ставить у відповідність вектор розмірності (1, n), де n – кількість ознак слова. Для такого представлення використовується спеціальна word2vec текстова модель, яка на вхід приймає окремі речення, а на виході строїть матрицю розмірності (d, n), таким чином кожне з d унікальних слів представляється вектором ознак довжини n. Для розрахунку векторних представлень слів, word2vec модель реалізує дві основні архітектури - Continuous Bag of Words (CBOW) і Skip-gram[9].

Схема роботи Continuous Bag of Words (CBOW)

На вході: словник V, слово *i* представляється як w_i

1) Створюємо векторне one hot представлення для вхідного контексту довжиною m: ($x_{(c-m)}$, ..., $x_{(c-1)}$, $x_{(c+1)}$, ..., $x_{(c+m)}$).
2) Для контексту отримуємо вбудовані вектори слів:

   $(v_{c-m} = \nu x^{(c-m)}, v_{c-m+1} = \nu x^{(c-m+1)}, \ldots, v_{c+m} = \nu x^{(c+m)})$, (3.1)

   де ν є вхідною матрицею слів такою, що і-й стовпець ν є тимчасовим вбудованим вектором для слова w_i, коли воно є входом для цієї моделі. Позначають цей вектор як v_i.
3) Усереднюємо ці вектори:

$$\hat{v} = \frac{v_{c-m}+v_{c-m+1}+\cdots+v_{c+m}}{2m} \quad (3.2)$$

4) Генеруємо оціночний вектор $Z = U\hat{v}$

   де U - матриця вихідного слова. J-й рядок U є n-розмірним вбудованим вектором слова w_j, коли воно є виходом моделі. Позначають цей ряд U як u_j.
5) Перетворюємо оціночний вектор на ймовірності за допомогою функції софтмакс: $\hat{y} = \text{softmax}(Z)$.
6) Оптимізуємо так, щоб наші ймовірності $\hat{y}$, які генеруються відповідали справжнім ймовірностям y, що є one hot представленням фактичного слова[17].

Схема роботи Skip-gram

На вході: словник V, слово *i* представляється як w_i

Матриці ν і U такі самі як і в моделі Continuous Bag of Words.

1) Генеруємо вхідний one hot вектор x.
2) Отримуємо вбудовані вектори для контексту $v_c = \nu x$.
3) Оскільки немає усереднення, просто присвоюємо $\hat{v} = v_c$.
4) Генеруємо 2m векторів оцінки ($u_{(c-m)}$, ..., $u_{(c-1)}$, $u_{(c+1)}$, ..., $u_{(c+m)}$) з використанням $u = U v_c$.
5) Перетворити кожний вектор оцінки на ймовірності за допомогою функції софтмакс: y = softmax (u).
6) Оптимізуємо так, щоб наш вектор ймовірностей генерувався і відповідав істинним ймовірностям $y_{(c-m)}$, ..., $y_{(c-1)}$, $y_{(c+1)}$, ..., $y_{(c+m)}$), які є одним з one hot векторів реального виходу[17].

Загальну схему обох алгоритмів демонструє рисунок 3.4. Архітектура CBOW передбачає поточне слово на основі контексту, а Skip-gram прогнозує «навколишні» слова, що відповідають поточному слову[18].

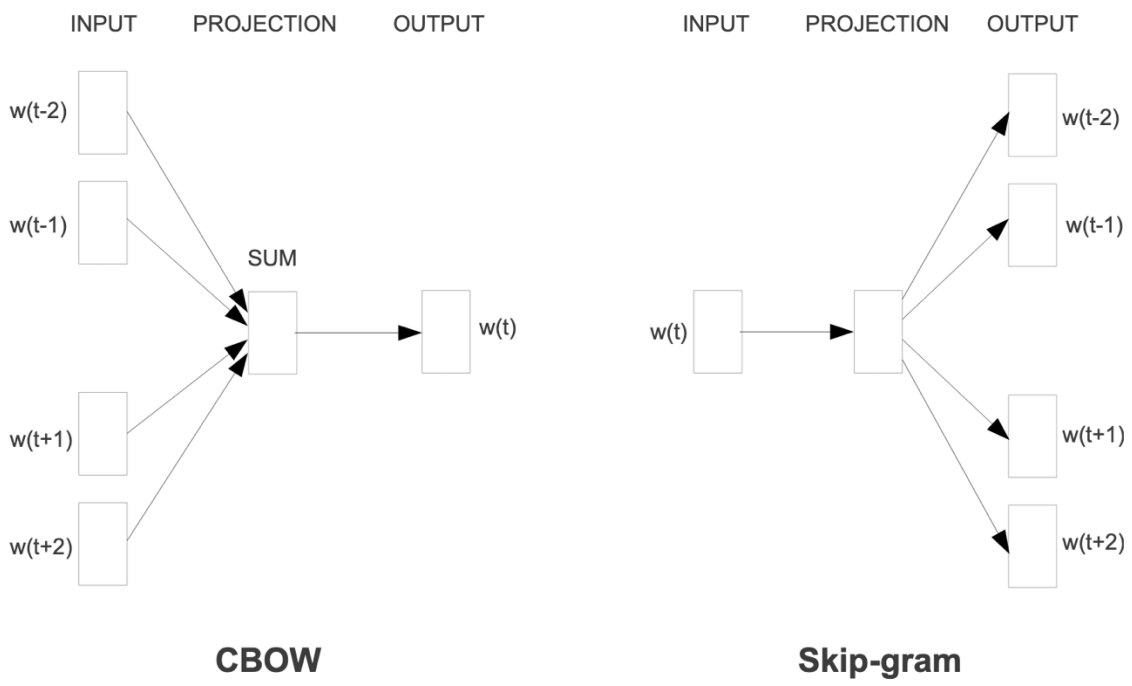


Рисунок 3.4 – Схема роботи CBOW і Skip-gram моделей

Якщо описувати принцип роботи word2vec словами - то це знаходження зв'язків між контекстами слів згідно з припущенням, що слова, що знаходяться в схожих контекстах, мають тенденцію означати схожі речі, тобто бути семантично близькими. Якщо формалізувати, то завдання стоїть таким чином: максимізація косинусної близькості між векторами слів, які знаходяться поруч один з одним, і мінімізація косинусної близькості між векторами слів, що не знаходяться далеко один від одного. Близькість в даному випадку розглядається, як близькість в контекстах.

Наприклад, слова «університет» і «інститут» часто зустрічаються в схожих контекстах, на кшталт «Студенти з університету дослідили» або «Студенти з інституту дослідили». Word2vec проаналізувавши ці контексти робить висновок, що слова «університет» і «інститут» є близькими за змістом. Тому векторні представлення цих двох слів будуть дуже схожими.

3.3 Архітектура нейронної мережі

На основі аналізу пункту 2 дипломної роботи, а також зважаючи на розмір пам’яті пристрою, на якому потім тренувалася нейронна мережа було побудовано таку архітектуру(схему наведено в додатку А):

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, None, 100)	14821000
bidirectional_1 (Bidirection	(None, None, 200)	160800
bidirectional_2 (Bidirection	(None, 800)	1923200
dense_1 (Dense)	(None, d)	1923200*d
activation_1 (Activation)	(None, d)	0

Total params: 16,905,000 + 1923200*d

- d – кількість слів в словнику унікальних слів;
- embedding_1 – шар, який представляє слово на основі навченої word2vec моделі;
- bidirectional_1 – перший двонапрямлений lstm шар;
- bidirectional_2 – другий двонапрямлений lstm шар;
- dense_1 – повнозв’язний шар;
- activation_1 – функція активації.

Загальна кількість параметрів, які треба підібрати складає 16,905,000 + 1923200*d, вона залежить від словнику унікальних слів. В якості функції активації було обрано softmax. Таким чином на виході ми отримуємо вектор, який має довжину d і для кожного слова з словника зазначена імовірність від 0 до 1 того, що воно буде наступним. В якості функції витрат обрано sparse_categorical_crossentropy - це функція втрат, яка використовується для вимірювання несхожості між розподілом спостережуваних класових міток і передбаченими ймовірностями класової приналежності.

Використання більшої кількості прихованих LSTM або повнозв'язних шарів в мережі, або збільшення кількості нейронів призводило до помилок, пов'язаних з пам'яттю на пристрої, де ця модель потім проходила тренування параметрів.

3.4 Стеганографічний алгоритм

Перш ніж реалізувати алгоритм приховування інформації в тексті і здобуття інформації із тексту, треба побудувати словник, який містив би слова, які найчастіше зустрічаються на першому місці в речення. Позначимо такий словник, як first_popular_words.

Функція приховання інформації

На вході:

послідовність секретних бітів,

словник first_popular_words,

генеративна модель,

word2vec модель,

n - кількість бітів, яке шифрує одне слово.

Принцип роботи:

1) Якщо довжина секретних біт не є кратною 2**n, вона доповнюється нулями у кінці;

2) Випадково обирається одне з слів з словнику first_popular_words, за допомогою word2vec моделі представляється у вигляді індексу і заноситься в масив word_idxs;
3) На основі масиву word_idxs модель робить передбачення і дає на виході вектор у довжиною d.

   З вектору у обираються 2**n індексів, які мають найбільші значення ймовірності і заносяться в candidate_pool.

   З послідовності бітів обираються перші n біт і представляються в десятковому вигляді, позначимо як dec_b.

   З candidate_pool обирається той індекс, послідовний номер якого співпадає з значенням dec_b. Індекс заноситься в масив word_idxs.

   Перші n біт секретної послідовності видаляються.
4) Шаг 3 повторюється доки послідовність секретних біт не буде порожньою.
5) Отриманий масив індексів word_idxs за допомогою word2vec моделі переводиться в текстове представлення.

На виході: автоматично згенерована текстова послідовність.

Функція здобуття інформації

На вході:

phrase - автоматично згенерована текстова послідовність,

генеративна модель,

word2vec модель,

n - кількість бітів, яке шифрує одне слово.

Принцип роботи:

1) за допомогою word2vec моделі представляється перше слово з phrase у вигляді індексу і заноситься в масив word_idxs, перше слово з послідовності text видаляється.

2) На основі word_idxs модель робить передбачення і дає на виході вектор у довжиною d.

   З цього вектору обираються 2**n індексів, які мають найбільший значення ймовірності. Ці індекси представляються у вигляді слів і заносяться в масив candidate_pool.

   З candidate_pool обирається той елемент, значення якого співпадає з значенням першого слова в phrase.

   Послідовний номер цього елементу переводиться у двійковий вигляд і заноситься в послідовність секретних біт.

   Індекс цього елементу заноситься в word_idxs.

   Перше слово з послідовності text видаляється.

3) Шаг 2 повторюється доки уся послідовність слів text не буде порожньою.

На виході: послідовність секретних бітів.

Висновки до розділу 3

В цьому розділі було проаналізовано дані, які використовувались для тренування нейронної мережі і на основі яких будуть генеруватися текстові послідовності. Також було досліджено методи представлення цих даних для тренування нейронної мережі.

Описано архітектуру нейронної мережі, які використовувалась для генерації текстових послідовностей, а також описано алгоритми приховання і відновлення інформації.

4 ДОСЛІДЖЕННЯ ОТРИМАННИХ РЕЗУЛЬТАТІВ

4.1 Аналіз результатів навчання word2vec моделі

Для представлення слова за допомогою цієї моделі, число ознак на виході було обрано: n=100, таким чином для кожного з наборів даних отримуємо матрицю розміром (100, d), де d – кількість унікальних слів для кожної з моделей, для кожного набору розмірність матриці представлено в таблиці 4.1. Можна було побудувати одну матрицю для всіх 3х наборів даних, проте кожен з наборів має свої специфічні слова, характерні для його тематики і в такому разі словник виявився б занадто великим, що погано вплинуло б на тренування генеративної моделі.

Таблиця 4.1 – Розмірності вихідних матриць

Набір даних	Розмірність
twitter	(130799, 100)
news	(195132, 100)
imdb	(148210, 100)

Для кожної з матриць, отриманих в результаті навчання моделі наведемо приклади пошуку семантично близьких слів на основі знаходження косинусної відстані між векторами(5 найближчих):

Для набору даних «twitter»:

hi: hello (0.83), hey (0.71), hii (0.71), heyy (0.66), quothiquot (0.66);

today: tonight (0.67), yesterday (0.62), todays (0.60), tomorrow (0.59), 2day (0.54);

try: attempt (0.63), continue (0.57), manage (0.56), stick (0.55), maybe (0.54).

Для набору даних «news»:

security: securitys (0.73), counterterrorism (0.64), guardsmen (0.62), cybersecurit y (0.60), reciprocity (0.57);

technology: software (0.79), ai (0.76), computing (0.76), technologies (0.75), bloc kchain (0.75);

institute: center (0.81), institutes (0.78), thinktank (0.70), university (0.68), univer sitys (0.64).

Для набору даних “imdb”:

film: movie (0.95), documentary (0.75), picture (0.74), flick (0.69), films (0.62);

good: decent (0.80), great (0.77), bad (0.75), cool (0.68), nice (0.68);

review: comment (0.82), reviews (0.70), summary (0.68), comments (0.68), imdb (0.67).

4.2 Аналіз результатів навчання рекурентної мережі

Кожна модель навчалась 300 епох на одній відеокарті Nvidia GeForce GTX 1060 3GB, проте зважаючи на те, що кожен з наборів даних має різний розмір словнику і різну кількість речень, тривалість однієї епохи для кожному з наборів була відмінною. Також з приводу досить незначної кількості пам’яті однієї відеокарти, порівняно з розміром словнику, міні-партії після яких змінювалось значення градієнта для кожного з наборів були невеликі і відрізнялись в залежності від набору. Протягом навчання для оцінки використовувалась метрика “Accuracy”, яка розраховується таким чином:

$$Accuracy = P/N, \tag{4.1}$$

де P – значення прикладів, для яких модель дає правильне передбачення наступного слова.

T – загальна кількість прикладів, на яких навчалась модель для кожного з наборів

Значення всіх параметрів для навчання наведено в таблиці нижче.

Таблиця 4.2 – Параметри навчання моделі

Набір даних	Розмір Міні-партії	Загальний час навчання	Accuracy
twitter	200	150 годин	82.4%
news	150	400 годин	73.8%
imdb	200	50 годин	91.5%

Приклади текстів, згенерованих моделями після навчання:

- Генерування тексту довжиною 5 слів:

Imdb:

“maybe professor entirely change graphic”

“she is agreeable housewife and girlfriend”

“all soft and solid materials”

News:

“come under pressure from advocates”

“see crime scenes gone cold”

“according to a new york”

Twitter:

“starting report time totally change”

“give improved subject for alert”

“worst bonus sweat result again”

- Генерування тексту довжиною 10 слів:

Imdb:

“she is professional killer and really knows how to kill”

“Mike completed programming only when he wanted really to sleep”

“his delivery was appreciated intensely he succeeds working in darkness”

News:

“if in washington tomorrow again tune revolt it wake president”

“upon taking office the trump administration will evaluate this case”

“white house said that the spending was a permanent part”

Twitter:

“usually it give improved subject force for some specific stuff”

“beautiful priceless flowers oh watching on it all the time”

“funny scenes from this video rly love it so funny”

4.3 Аналіз стеганографічного алгоритму

4.3.1 Embedding Rate

Однією з оцінок є метрика Embedding Rate(ER), яка показує скільки інформації може бути вбудовано в тексти, що є важливим показником для оцінки продуктивності стенографічного алгоритму. Метод розрахунку ER полягає в тому, щоб розділити фактичне число вбудованих бітів на кількість бітів всього генерованого тексту[4].

Математичний вираз для ER має такий вигляд:

$$ER = \frac{1}{N}\sum_{i=1}^{N}\frac{(L_i-1)k}{B(S_i)} = \frac{1}{N}\sum_{i=1}^{N}\frac{(L_i-1)k}{8\times\sum_{j}^{L_j} m_{i,j}} = \frac{\widehat{(L-1)}\times k}{8\times\hat{L}\times\hat{m}} \qquad (4.2)$$

де N - кількість згенерованих речень, а L_i - довжина i-го речення. k вказує число бітів, вбудованих у кожне слово, а B (S_i) - число бітів i-го речення. Оскільки кожна англійська буква фактично займає один байт в комп'ютері, тобто 8 біт, число біт, зайнятих кожним англійським реченням, є B (S_i) = 8 × $\sum_{j}^{L_j} m_{i,j}$, де $m_{i,j}$ являє собою кількість букв, що містяться в j-му слові i-го речення. $\hat{L}$ і $\hat{m}$ позначають середню довжину кожного речення у сформованому тексті і середню кількість букв, що містяться в кожному слові[4].

В таблиці 4.3 наведено результати для різних наборів даних і різного параметру k, який вказує скільки біт шифрує одне слово. У таблиці 4.4 вказано значення значення ER для деяких інших алгоритмів.

Таблиця 4.3 – Embedding Rate для різних наборів даних.

	ER for twitter	ER for imdb	ER for news	$\widehat{ER}$
k = 1	2.75%	2.67%	2.49%	2.63%
k = 2	5.50%	5.34%	4.99%	5.27%
k = 3	8.25%	8.01%	7.48%	7.91%
k = 4	11.00%	10.68%	9.98%	10.55%
k = 5	13.75%	13.36%	12.47%	13.19%

Таблиця 4.4 – ER інших алгоритмів текстової стеганографії

Методи	ER
Метод запропонований в [19]	0.3%
Метод запропонований в [20]	0.35%
Метод запропонований в [21]	1.0%
Метод запропонований в [22]	1.57%

Порівнюючи значення у таблицях 4.3 і 4.4 можна зазначити, що запропонований алгоритм має досить високі значення ER на відміну від попередніх алгоритмів текстової стеганографії.

4.3.2 Швидкість роботи

Важливим параметром є швидкість з якою алгоритм генерує текст з приховання в ньому секретної інформації, а потім з цього тексту та відновлює секрет. Вимірюється вона у байтах на секунду, тобто скільки байт інформації можна приховати/відновити за 1 секунду.

В таблиці 4.5 наведено порівняння швидкості роботи запропонованого алгоритму з існуючими аналогами.

Таблиця 4.5 – Швидкість роботи

метод	Encoding time	Decoding time
Запропонований у цій роботі	1.2 kB/s	1.15 kB/s
Метод запропонований у [23]	0.2 kB/s	0.2 kB/s
Метод запропонований у [4]	0.1 kB/s	0.1 kB/s

4.3.3 Перплексія

Метою, створеної системи є приховування наявності інформації в контейнері для забезпечення безпеки важливої інформації. Таким чином, маскування інформації є найважливішим фактором оцінки продуктивності системи приховування. Тобто стеганографічна операція не повинна призводити до відмінностей у розподілі носіїв у семантичному просторі. Виходячи з цього ми повинні перевірити, що речення, створені нашою моделлю, достатньо близькими до нестеганографічного носія (тобто, тренувальних текстів) на статистичній моделі мови. У теорії інформації, перплексія є вимірюванням того, наскільки добре розподіл ймовірності або ймовірнісна модель передбачає зразок. В галузі обробки природніх мов, перплексія є стандартною метрикою для перевірки якості речень. Вона визначається, як середня лог-ймовірність за словом в тестових текстах:

$$perplexity = 2^{-\frac{1}{n}\log p(s)} = 2^{-\frac{1}{n}\log p(w_1,w_2,w_3,..w_n)} = 2^{-\frac{1}{n}\log \sum_{i=1}^{n} logp(w_i|\, w_1,w_2,..w_{i-1})} \quad (4.3)$$

де S={w_1, w_2, w_3,…, w_n} – згенероване речення, p(S) вказує на розподіл ймовірностей у словах в реченні, ймовірність розраховується з мовної моделі навчальних текстів. n - кількість слів у генерованих реченнях.

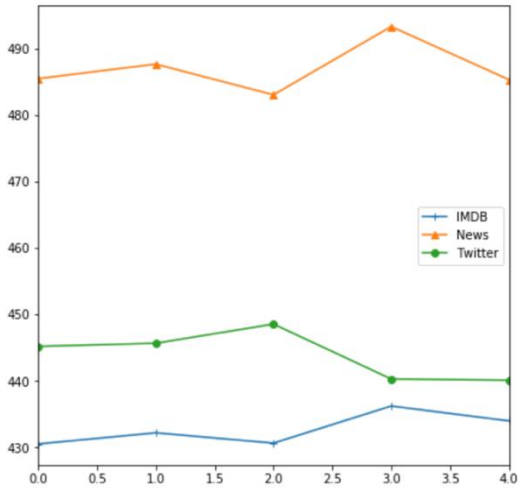
Таким чином перплексія обчислює різницю у статистичному розподілі мовної моделі між генерованими текстами та навчальними текстами. Чим меншим є його значення, тим більш подібним буде згенерований текст до статистичного розподілу навчального тексту. Для того, щоб оцінити перплексію було згенеровано по 1000 речень для різних значень bpw – кількості біт, які шифруються в одному повідомлені, для кожного з наборів даних. Спираючись на отримані результати було побудовано порівняльну таблицю 4.6, де значення перплексії даного алгоритму порівнюються із значеннями перплексії двох методів, заснованих на марковській ланцюговій моделі. Також на рисунку 4.1 побудовано графіки порівняння перплексії трьох алгоритмів для кожного з наборів.

Таблиця 4.6 – Значення перплексії для різних алгоритмів

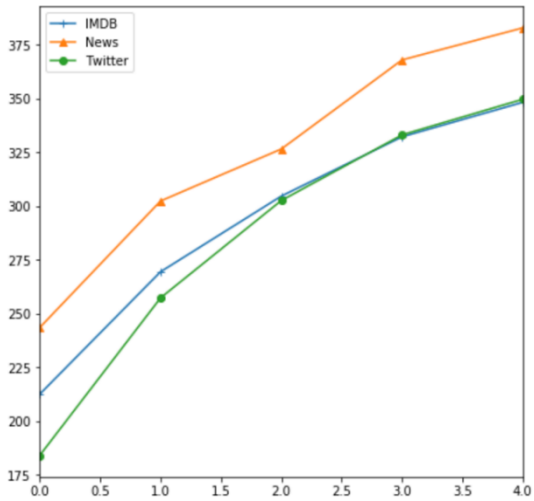
Вхідні дані		Метод запропонований у [23]	Метод запропонований у [22]	Метод, запропонований у цій роботі
Набір даних	bpw			
IMDB	1	430.38	212.58	23.10
	2	432.16	269.51	28.64
	3	430.61	304.71	32.44
	4	436.19	332.32	35.26
	5	433.95	348.36	36.25
News	1	485.47	243.57	17.63
	2	487.65	302.43	30.27
	3	483.03	326.62	44.32
	4	493.30	368.07	72.00
	5	485.31	382.99	103.25

Кінець таблиці 4.6

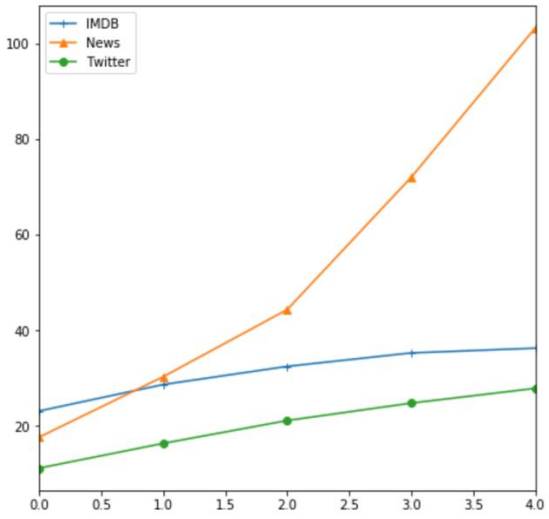
Twitter	1	445.16	184.09	11.15
	2	445.64	257.36	16.33
	3	448.52	302.66	21.11
	4	440.26	333.20	24.76
	5	440.08	349.78	27.85



(a) Метод запропонований у [23]



(b) Метод запропонований у [22]



(c) Метод, запропонований у цій роботі

Рисунок 4.1 – Порівняння значень перплексії трьох алгоритмів

Як можна бачити з таблиці 4.6, запропонована модель досить добре пристосовується до розподілу навчального тексту у порівнянні з попередніми алгоритмами.

Також цікаво подивитись значення перплексії кожної з моделей не на тренувальному наборі, а на іншому, який не використовувався для навчання. Для цього в формулі 4.3 значення ймовірністей p(S) розраховувались з мовної моделі двох інших наборів, відмінних від того, на якому навчалася нейронна мережа, а речення генерувались на кожній з трьох моделей. Значення, які було отримано, наведено в таблиці 4.7, а також на рисунку 4.2.

Таблиця 4.7 – Значення перплексії на різних моделях

bpw	IMDB model	News model	Twitter model
1	94.35	87.42	80.44
2	143.01	144.00	144.00
3	159.78	195.36	221.32
4	176.06	240.51	168.89
5	186.10	284.049	243.87

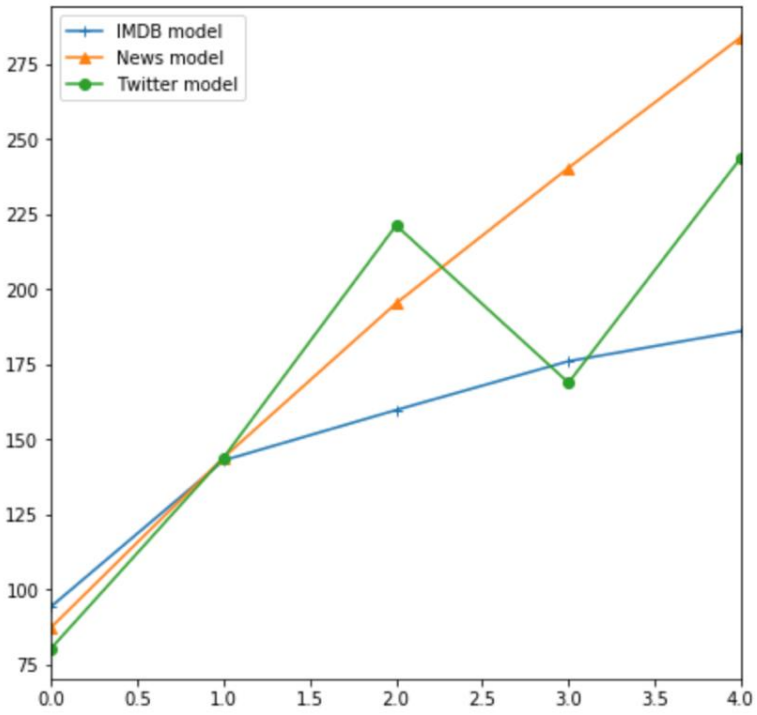


Рисунок 4.2 – Порівняння значень перплексії на різних моделях

Як можна бачити отримані результати відрізняються від результатів у таблиці 4.6, що свідчить про те, що алгоритм є досить чутливим до даних, які використовувались для навчання. Тому для того щоб речення, які отримуються на виході були добре згенеровані, набір вхідних даних повинен бути достатньо великим, а також для того, щоб згенеровані тексти були актуальними через деякий час, модель треба дотреновувати на нових даних, щоб розширювати словник і оновлювати ваги.

4.3.4 Експертна оцінка

Іншим методом оцінити якості машинно-генерованих текстів є експертна людська оцінка. Запропонована система система орієнтована на стеганографію, і її мета полягає в тому, щоб генерувати тексти, які не будуть залучати підозри з боку третьої сторони. Таким чином, ми використовуємо знаменитий тест Тьюрінга для оцінки нашої моделі. Експерту було запропоновано судити, чи були ці тексти створені машиною або написані людиною. Відповідно до того, як багато машинно-генерованих текстів ідентифікуються, як створені людиною, ми можемо оцінити природність текстів, сформованих нашим методом стеганорафії.

Для оцінки було використано 18 тексті довжиною 5 слів, дев’ять текстів, згенерованих даною моделлю і дев’ять текстів такої ж довжини написані людиною. В якості експертів виступили студенти, а також люди, які закінчили ВНЗ. Результати представлено в таблиці 4.8.

Таблиця 4.8 – Експертна оцінка результатів

	Ідентифіковані як людські	Ідентифіковані як згене ровані машиною
Тексти згенеровані моделлю	33%	67%
Людські тексти	79%	21%

Бачимо, що метод слабко пройшов тест Тьюрінга, критерієм якого є обдурити людей не менше ніж на 30% запитань. Звісно якщо генерувати тексти більшої довжини, вони хоч і будуть виглядати схожими на людські,

проте змістове наповнення цього тексту не матиме ніякого сенсу і в такому разі виокремити автоматично згенеровані тексти буде досить просто.

Висновки до розділу 4

В даному розділі було розглянуто результати тренування текстової word2vec моделі. А також проаналізовано тренування нейронною моделі і наведено приклади згенерованих текстів.

Досліджено стеганографічний алгоритм, швидкість та якість його роботи у порівнянні з іншими, а також наведено експертну оцінку результатів текстової генерації з прихованням в цьому тексті секретної інформації.

ВИСНОВКИ

Лінгвістична стеганографія заснована на технології автогенерації тексту, є досить перспективною, а також складною. Через високий ступінь кодування та меншу інформаційну надлишковість у тексті, дуже складною проблемою є приховання інформації в ньому на довгий час.

У даній бакалаврській дипломній роботі було проведено огляд основних методів текстової стеганографії. Розглянуто алгоритми, на основі яких вони працюють, проаналізовано переваги і недоліки кожного з них.

Наведено описання того, як працюють нейронні мережі, розглянуто основні методи, які використовуються для вирішення задач оптимізації і апроксимації певних функцій.

Детально досліджено рекурентні нейронні мережі, які оброблюють послідовності і обумовлено чому саме цей тип мереж використовують для текстових даних. Розглянуто проблеми які існують в РНМ та методи вирішення цих проблем, останні архітектури РНМ та засоби, які дозволяють покращити вихідні послідовності.

Проаналізовано способи представлення текстових послідовностей, як їх можна інтерпретувати в векторному вигляді та яке векторне представлення найкраще використовувати для нейронних мереж.

У даній роботі було запропоновано метод стеганографії, який може автоматично генерувати стеганографічний текст на основі рекурентних нейронних мереж. Він може автоматично генерувати текстовий носій з точки зору секретної інформації, яку необхідно вбудувати.

Розроблено програмне забезпечення, яке є комплексним вирішенням задачі приховування даних.

Для реалізації поставлених задач було використано мову програмування Python та бібліотеки для розробки глибоких нейронних мереж Keras, TensorFlow, Scikit-learn та ін.

Експериментальні результати показують, що продуктивність запропонованої моделі перевершує всі попередні споріднені методи з точки

зору інформаційної прихованої ємності і швидкості роботи алгоритмів приховання/відновлення секретних даних, а також тексти, які генерується за допомогою моделі в приблизно третині випадків розпізнаються, як створені людиною.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Go A. Twitter sentiment classification using distant supervision[J] / A. Go, R. Bhayani, L. Huang. // CS224N. – 2009.
2. Karpathy A. The Unreasonable Effectiveness of Recurrent Neural Networks [Електронний ресурс] / A. Karpathy. – 2015. – Режим доступу до ресурсу: http://karpathy.github.io/2015/05/21/rnn-effectiveness/.
3. Єлісєєв А. Исследование и разработка методов и алгоритмов стеганографического анализа отдельных контейнеров и их связанных наборов : дис. канд. техн. наук : ВАК / Єлісєєв А. – Ростов-на-Дону, 2013. – 173 с.
4. Automatically Generate Steganographic Text Based on Markov Model and Huffman Coding [Електронний ресурс] / [Y. Zhongliang, J. Shuyu, H. Yongfeng та ін.]. – 2018. – Режим доступу до ресурсу: https://arxiv.org/pdf/1811.04720.pdf.
5. Agarwal M. TEXT STEGANOGRAPHIC APPROACHES: A COMPARISON [Електронний ресурс] / Monika Agarwal. – 2013. – Режим доступу до ресурсу: https://arxiv.org/pdf/1302.2718.pdf.
6. Fabien A. Information Hiding|A Survey [Електронний ресурс] / A. Fabien, P. Petitcolas, J. Ross // Proceedings of the IEEE. – 1999. – Режим доступу до ресурсу: https://www.petitcolas.net/fabien/publications/ieee99-infohiding.pdf.
7. Kumar A. Steganography-A Data HidingTechnique [Електронний ресурс] / A. Kumar, K. Pooja // International Journal of Computer Applications. – 2010. – Режим доступу до ресурсу: chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/https://pdfs.semanticscholar.org/48f3/3b8cc8cb733b70c584825f3170a755ec30e6.pdf.
8. Rabah K. Steganography-The Art of Hiding Data [Електронний ресурс] / Kefa Rabah. – 2004. – Режим доступу до ресурсу: https://www.researchgate.net/publication/45949372_Steganography-The_Art_of_Hiding_Data.

9. Huang S. Word2Vec and FastText Word Embedding with Gensim [Електронний ресурс] / Steeve Huang. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c.
10. MOAWAD A. The magic of LSTM neural networks [Електронний ресурс] / Assaad MOAWAD. – 2018. – Режим доступу до ресурсу: https://medium.com/datathings/the-magic-of-lstm-neural-networks-6775e8b540cd.
11. Hassan Shirali-Shahreza M. A New Synonym Text Steganography [Електронний ресурс] / M. Hassan Shirali-Shahreza // IEEE. – 2008. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/4604331.
12. Bender W. Techniques for datahiding [Електронний ресурс] / W. Bender, D. Gruhl, N. Morimoto // IBM SYSTEMS JOURNAL – Режим доступу до ресурсу: https://pdfs.semanticscholar.org/8c82/c93dfc7d3672e58efd982a23791a8a419053.pdf.
13. Cummins J. Steganography And Digital Watermarking [Електронний ресурс] / J. Cummins, P. Diskin, S. Lau // Arizona State University. – 2004. – Режим доступу до ресурсу https://www.cs.bham.ac.uk/~mdr/teaching/modules03/security/students/SS5/Steganography.pdf.
14. Ярмолик С. В. Защита информации [Електронний ресурс] / С. В. Ярмолик, Ю. Н. Листопад // Информатизация образования. – 2005. – Режим доступу до ресурсу: http://www.giac.unibel.by/sm_full.aspx?guid=7933.
15. Goodfellow I. Deep Learning / I. Goodfellow., 2018. – 652 с. – (Massachusetts Institute of Technology).
16. Nielsen M. Neural Networks and Deep Learning / Michael Nielsen., 2006. – 212 с.

17. Chaubard F. Deep Learning for NLP [Електронний ресурс] / F. Chaubard, R. Mundra, R. Soche. – 2016. – Режим доступу до ресурсу: https://cs224d.stanford.edu/lecture_notes/notes1.pdf.
18. Mikolov T. Efficient Estimation of Word Representations inVector Space [Електронний ресурс] / T. Mikolov, G. Corrado, K. Chen. – 2013. – Режим доступу до ресурсу: https://arxiv.org/pdf/1301.3781.pdf.
19. Murphy B. The syntax of concealment: reliable methods for plain textinformation hiding [Електронний ресурс] / B. Murphy, C. Vogel. – 2014. – Режим доступу до ресурсу: https://www.researchgate.net/profile/Brian_Murphy9/publication/229025905_The_Syntax_of_Concealment_Reliable_Methods_for_Plain_Text_Information_Hiding/links/00b49519ffbe4ef1f6000000/The-Syntax-of-Concealment-Reliable-Methods-for-Plain-Text-Information-Hiding.pdf?origin=publication_detail.
20. Sabyasachi S. A real time text steganalysis by using statistical method [Електронний ресурс] / S. Sabyasachi, S. Dutta, G. Sanyal. – 2016. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/7569256.
21. Xianyi X. Coverless Information Hiding Method Based on the Chinese Mathematical Expression [Електронний ресурс] / X. Xianyi, H. Sun, Y. Tobe. – 2016. – Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-3-319-27051-7_12.
22. Zhou Z. Coverless Information Hiding Method Based on Multi-keywords [Електронний ресурс] / Z. Zhou, Y. Mu, N. Zhao. – 2016. – Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-3-319-48671-0_4.
23. Moraldo H. An Approach for Text SteganographyBased on Markov Chains [Електронний ресурс] / Hernan Moraldo. – 2014. – Режим доступу до ресурсу: https://arxiv.org/pdf/1409.0915.pdf.

Додаток А

Схема архітектури нейронної мережі