УДК 681.3.093:044.3
Яременко В. С., Будьонний Д. Ю.
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

ПІДХІД ДО ВИКОРИСТАННЯ ФІЛЬТРА БЛУМА ДЛЯ БАГАТОКЛАСОВОЇ КЛАСИФІКАЦІЇ ТЕКСТОВИХ ДАНИХ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Яременко В. С., Будьонний Д. Ю. Підхід до використання фільтра блума для багатокласової класифікації текстових даних в режимі реального часу. У даній роботі розглянуто фільтр Блума, який вирішує задачу фільтрації потоків, та було запропоновано новий підхід використання цього фільтру для класифікації текстових даних. В якості вхідних даних було обрано текстові дані, які надходять у реальному часі. Розглянуто модель з точки зору точності класифікації, швидкості навчання моделі, кількості використаної пам'яті та швидкістю видачі результату класифікації. Представлено метод донавчання моделі та критерій відбору слів для покращення навчання моделі. Показано процес навчання моделі для багатокласової класифікації. Виявлені проблеми даного підходу та запропоновані проблеми їх вирішення.
Ключові слова: аналіз потокових даних, фільтр Блума, аналіз текстових даних , класифікація текстів.

Яременко В. С, Будённый Д. Ю. Подход к использованию фильтра блума для многоклассовой классификации текстовых данных в режиме реального времени. В данной работе рассмотрен фильтр Блума, который решает задачу фильтрации потоков, и был предложен новый подход использования этого фильтра для классификации текстовых данных. В качестве входных данных были выбраны текстовые данные, которые поступают в режиме реального времени. Рассмотрена модель с точки зрения точности классификации, скорости обучения, количества использованной памяти и скоростью выдачи результата классификации. Представлен метод обучению модели и критерий отбора слов для улучшения процесса обучения модели. Показан процесс обучения модели для багатокласовои классификации. Выявлены проблемы данного подхода и предложены способы их решения.
Ключевые слова: анализ потоковых данных, фильтр Блума, анализ текстовых данных, классификация текстов.

Yaremenko V. S., Budonnyi D. Y. Approach of the bloom filter application for real time text data multi-class classification
This paper examines the Bloom filter that solves the problem of streaming data filtration. A new approach was proposed to use this filter for texts classification. Existing articles in this area and the current problems of the classification of text data are studied. The theoretical part is presented, as well as the practical example of constructing a Bloom filter. The process of model training is shown by constructing the Bloom filter for multiclass classification. The opportunity to increase the number of classes for classification with possible limitations and problems is presented. The model training method and the word selection criterion for improving the model learning process are presented, as well as the process of retraining the existing model during its work using these criteria. The stages of text preprocessing for increasing the accuracy of the model were presented. As the input a real-time text data, which come from multiple resources, was selected. A solution for processing incoming data to avoid the problem of losing some of the data during the working of the system is presented. The model is considered in terms of classification accuracy, learning speed, amount of memory used and speed of classification. The influence of the components of the Bloom filter on the final result of this model, as well as the probability of the false-positive results for various system parameters, are examined. The conclusions of the work of the presented approach are drawn. The problems of this approach are revealed and the ways of their solution or improvement are suggested. Prospects for further research for the development of this model are presented.
Keywords: streaming analytics, Bloom filter, text data analysis, texts classification

Постановка проблеми. Задача класифікації текстових даних є однією з багатьох задач науки про аналіз великих масивів даних. За останні роки було розробено велику кількість методів для вирішення цієї задачі. Ще однією актуальною задачею є обробка даних у реальному часі, коли важливу роль відіграє швидкість обробки даних для видачі результатів та кількість використаної пам'яті тому, що дані, які не будуть одразу оброблені, будуть втрачені назавжди [3].

Аналіз останніх досліджень і публікацій. У статті «Role of Bloom Filter in Big Data Research: A Survey» описано застосування фільтра Блума у системі зберігання великих даних. Було виявилено, що флеш-пам'ять є найбільш перспективною дослідницькою сферою для інтеграції фільтра Блума для створення системи зберігання великих даних. Було продемонстровано, що масштабованість фільтра Блума є проблемою, яку можна вирішити, застосовуючи фільтр Блума у флеш-пам'яті. Запис у флеш-пам'ять відбувається швидше, ніж запис на диск. Однак, при незначних записах у флеш-пам'ять, потрібно видаляти цілі сторінки даних та записувати їх знову. Відповідно, запис у флеш-пам'ять відбувається повільніше, ніж в оперативну пам'ять. Оскільки, розмір оперативної пам'яті обмежений, а дуплікація даних вимагає додакового простору, тому доцільно використати фільтр Блума у флеш-пам'яті. Тим не менш, існує компроміс між продуктивністю та необхідною пам'яттю [1].

© Яременко В. С., Будьонний Д. Ю.

Було виявлено, що в більшості систем, які зберігають, великі дані, розміщений фільтр Блума, щоб уникнути випадкового доступу до диска, що значно підвищить продуктивність системи. Тому використання фільтру Блума пришвидшує обробку даних, що надходять у реальному часі [1].

У статті «Optimizing Bloom Filter: Challenges, Solutions, and Comparisons» розглянуто існуючі варіанти фільтра Блума з двох сторін: продуктивності та загальних правил. Було роглянуто більше десяти варіантів для підвищення продуктивності, зменшення помилкових позитивних результатів та витрат на впровадження. Крім того, досліджено більше десяти варіантів узагальнення фільтра Блума у більшості сценаріїв, диверсифікуючи вхідні набори та поліпшуючи вихідні функції. Зокрема для зменшення помилкових позитивних результатів, вибору оптимальних хеш-функцій, обнулення бітів у векторах або представлення елементів диференційовано. Для подальшого спрощення процесу впровадження фільтра Блума було запропоновано близько десяти варіантів для оптимізації його витрат на обчислення, доступу до пам'яті та ефективності використання простору. Крім представлення загальних множин, розглядалися варіанти для представлення мультисетів, динамічних наборів, зважених множин, ключових значень, наборів послідовностей та просторових множин [4].

У статті «Robust Bloom Filters for Large Multilabel Classification Tasks» було представлено новий метод – Надійний Фільтр Блума (Robust Bloom Filters або R-BF). Він покращується випадковими хеш-функціями, спираючись на структурну особливість наборів міток у наборах даних MLC: багато міток ніколи не спостерігаються в одному цільовому наборі або спільно виникають з ймовірністю, достатньо малою, щоб нею нехтувати. Спочатку формалізується структурна функція - це поняття взаємовиключних кластерів міток, потім описується хеш-функції та надійний алгоритм декодування [2].

Мета статті. Основною метою є створення моделі з використанням фільтра Блума для інтелектуальної обробки потокових текстових даних, а саме – для вирішення задачі багатокласової класифікації текстів, які надходять у реальному часі. Оцінка моделі відбувається за точністю класифікації, швидкістю навчання моделі, кількістью використаної пам'яті та швидкістю видачі результату класифікації.

Теоритичні відомості. Складові частини Фільтра Блума:

1. Масив *n* біт, спочатку рівних *0*.

2. Набір хеш-функцій h_1, h_2, ..., h_k, кожна з яких відображає значення «ключа» в n комірок, відовідно до *n* бітів масиву.

3. Множина *S*, що містить *m* ключів.

Призначення фільтру Блума – пропускати всі елементи потоку, ключі яких належать *S*, і відкидати більшість елементів з ключами, що не належать *S* [3].

Спочатку створюємо бітовий масив, обнуливши всі біти. Застосуємо до кожного ключу в *S* кожну з *k* хеш-функцій. Встановимо в 1 біти, номери яких збігаються з $h_i(K)$ для деякої хеш-функції h_i і деякого ключа *K* з *S* [3].

При обробці ключа *K*, що надійшов з потоку, перевіряємо, що значення всіх бітів з номерами

$$h_1(K), h_2(K), ..., h_k(K)$$

рівні 1. Якщо це так, то пропускаємо елемент. Якщо хоча б один біт дорівнює 0, то ключ К не може належати *S*, тому відкидаємо елемент [3].

Додавання елемента в фільтр відбувається наступним чином - для значення вираховується кілька різних хешів, після чого виставляються відповідні біти в бітовому полі. Перевірка на входження здійснюється схожим чином - спочатку вираховуються хеші для значення, після чого перевіряється, чи виставлені відповідні біти. Так як різні значення можуть мати однакові хеш-коди, то можливі хибнопозитивні спрацьовування. Але якщо хеш-коди різні, то однозначно, значення теж різні, тому помилково-негативних спрацьовувань буде менше зі збільшення кількості хеш-функцій.

В загальному випадку ймовірність отримання хибнопозитивного результату можна обчислити за формулою:

$$(1 - e^{-km/n})^k \text{ [3]}$$

З фомули видно, що для зменшення ймовірності отримання хибнопозитивного результату потрібно збільшити кількість хеш-функцій *k* або кількість бітів масиву *n*.

Нижче описаний фільтр, який складається з 16 біт і працює з двома хеш-функціями.

© Яременко В. С., Будьонний Д. Ю.

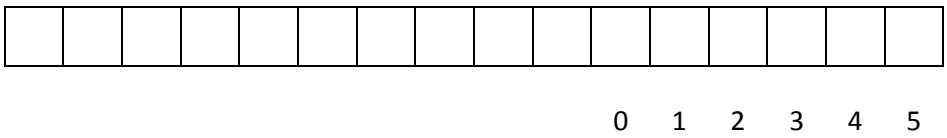

Рис. 1. Пусте бітове поле довжиною 16 біт

Додається слово "Football", перша функція дає хеш код, рівний 25425, а друга 894346520. Так як за умовою доступно всього 16 біт, тому знаходиться значення по модулю 16. Перший хеш рівний 1, а другий - 8. На відповідних позиціях виставляється значення «1».

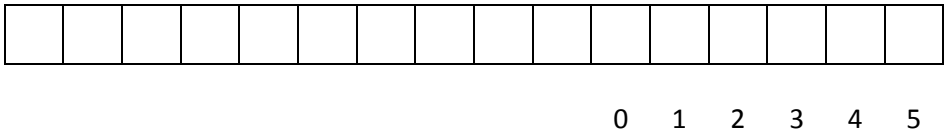

Рис. 2. Масив бітів після додавання елементу, для якого хеш-функції повернули 1 і 8

У випадку, якщо на вхід (на етапі навчання) системи подається слово слово "Cinema", перший хеш дорівнює 8, а другий - 7, виставляються відповідні біти.

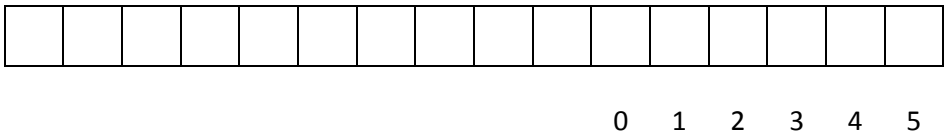

Рис. 3. Масив бітів після додавання елементу, для якого хеш-функції повернули 7 і 8

З рис. 3 видно, що перетнулися біти під номером 8 двох рядків, тому видалити одне зі слів з фільтра не вийде. Для видалення одного значення доведеться перераховувати хеші всіх інших значень до тих пір, поки хоча б одне зі значень не дасть ті ж біти, або поки не перерахуємо всі значення.

Потім перевіряється, чи проходить слово "Nation" даний фільтр. Вираховується для цього слова хеш-значення, для прикладу визначемо, що вони рівні 3 і 7 відповідно. Біт 3 не виставлений (дорівнює нулю), тому однозначно, цей рядок не входить в фільтр. Візьмемо інше слово "Sky", його хеші мають значення 1 і 8, хоча даного значення в фільтрі немає, фільтр пропустить дане слово, що і дає нам хибнопозитивне значення.

Представлення моделі. Нашою задачею є класифікація текстів у реальному часі. Для прикладу будемо виконувати класифікацію на 2 класи з подальшим розширенням. Також буде показано спосіб донавчання моделі в режимі реального часу.

Першим етапом є навчання. Для цього візьмемо по 1 тексту за еталон до кожного класу на якому буде проведено перше навчання моделі. Навчання буде відбуватися у 2 етапи:

1. Попередня обробка тексту
2. Побудова фільтра Блума

Попередня обробка тексту складається з наступних етапів:

1. Переведення всіх слів в нижній реєстр
2. Видалення чисел
3. Видалення знаків пунктуації
4. Видалення символів розділення слів

© Яременко В. С., Будьонний Д. Ю.

5. Видалення стоп-слів та слів, які не мають змістового навантаження
6. Токенізація
7. Стіммінг
8. Лематизація

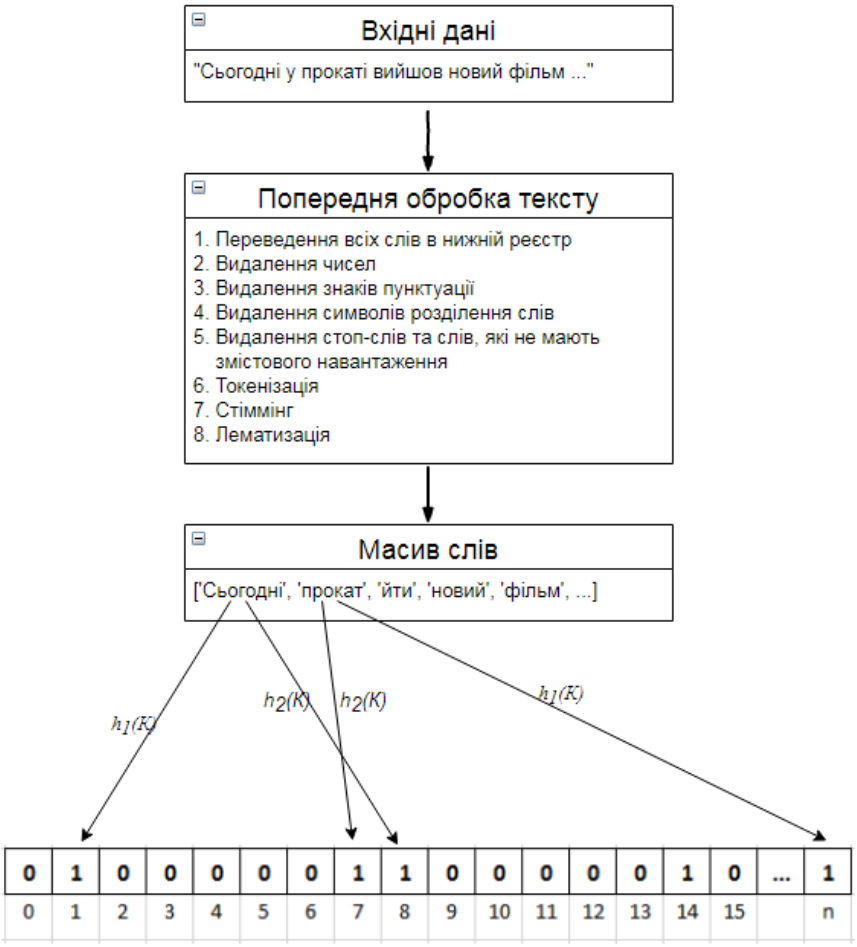

Рис.4. Навчання моделі

Після проведення попередньої обробки ми отримаємо масив слів без зайвих символів та у простій формі. Наступним етапом є створення фільтру Блума використавши декілька хеш-функцій. Після виставлення бітів в 1 для кожного слова з масиву для кожної хеш-функції, навчання моделі можна вважати завершеним.

Варто не використовувати надмірну кількість хеш-функцій тому, що із збільшенням кількості хеш-функцій збільшується кількість одиниць, які будуть виставлені в масиві бітів, а це збільшує кількість хибнопозитивних результатів. Якщо всі біти будуть рівні одиниці, то дана модель не буде працювати, бо кожне слово буде давати позитивний результат.

Далі необхідно виконати навчання моделі, в даному випадку, для двох класів. Можна створити необхідну кількість класів використавши допустиму кількість пам'яті. Але варто пам'ятати й про хибнопозитивні значення.

Так, наприклад, якщо доступно 100 Мб оперативної пам'яті і необхідно створити 4 класи при розмірі обробленого масиву в 10 000 слів. Тобто дані будуть міститись у 800 000 бітів, які будуть розділені на 4 класи по 200 000 біт у кожному. Використавши 3 хеш-функції ми отримаємо наступне значення у відсотках хибнопозитивного результату для кожного зі слів:

$(1 - e^{-km/n})^k = (1 - e^{-3 * 10\,000 / 200\,000})^3 = (1 - 0.861)^3 = 0.003$, тобто 0.3%

© Яременко В. С., Будьонний Д. Ю.

Але якщо при тих самих умовах потрібно розбити на 40 класів, то вийде 20 000 бітів на кожен клас і ми отримаємо наступні результати:

$(1 - e^{-km/n})^k = (1 - e^{-3 * 10\,000 / 20\,000})^3 = (1 - 0.223)^3 = 0.469$, тобто 46.9%

Як видно з прикладів всі параметри фільтра Блума тісно пов'язані і необхідно доцільно розраховувати можливості системи.

Після навчання обох класів ми отримаємо два фільтра Блума для кожного класу. Наступною задачею є логіка приналежності вхідного тексту до відповідного класу.

Одним методів для класифікації тексту до одного з класів є підрахунок відсотка слів з вхідного тексту, у яких значення всіх хеш-функцій дорівнює «1». Даний вираз, позначимо його *P*, можна представити у наступному вигляді:

$$P = \frac{\sum_{i=1}^{m} h_1(S[i])h_2(S[i]) \dots h_k(S[i])}{m}$$

де *S* – масив слів, *m* – кількість слів, *h* – хеш-фукція.

Робота моделі складається з наступних етапів:

1. Попередня обробка вхідного тексту
2. Розрахунок *P* для кожного з класів
3. Порівняння *P* кожного з класів
4. Видача результату

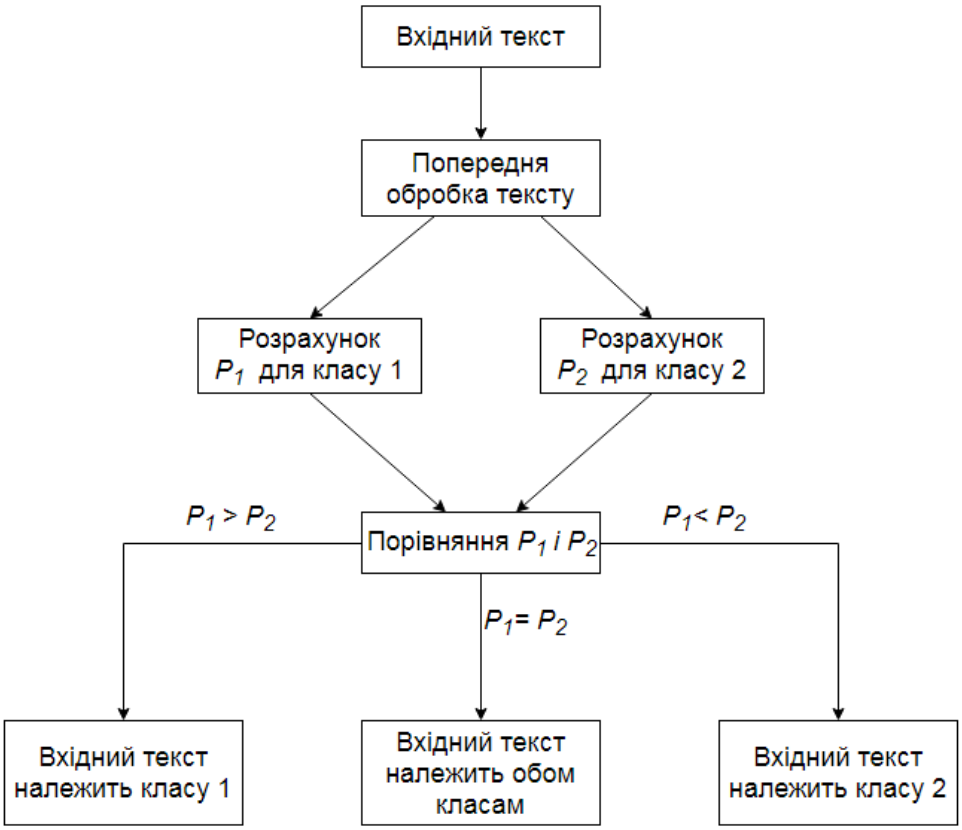

Рис.5. Робота моделі

Попередня обробка вхідного тексту відбувається таким самим чином, як і при навчанні моделі. Розрахунок *P* для кожного з класів може відбуватися паралельно, що значно прискорить роботу класифікатора. На виході ми отримаємо результат приналежності даного тексту до одного з класів або при рівності *P* рівну приналежність до обох класів. Схема роботи моделі представлена на рисунку 5.

© Яременко В. С., Будьонний Д. Ю.

Представлену модель також можна довчати, і у даній статті запропоновано наступних підхід. Визначаємо такий P, що вхідні масив слів вхідних даних буде використовуватися для подальношо навчання моделі. Припустимо, що при P > 0.5, тобто більше 50% слів входять до одного з класів, то даним масивом слів можна довчити модель класу до якого належав вхідний текст. Одінєю з проблем даного підходу є перенавчання моделі, коли майже всі біти рівні 1 і ймовність хибнопозитивного результату є достить великою.

Одним з методів покращення навчання та донавчання моделі для більш точної класифікації є створення критерію відбору слів після етапу попередньої обробки тексту. Пропонується створити мішок слів (bag of words) та обрахувати ваги кожно з них. Наприклад, можна відібрати лише слова з вагою більше 1% – це дасть точніші результати при класифікації текстів на даному наборі.

Завершальним етапом у побудові системи є обробка даних у реальному часі. Для вирішення цієї задачі доцільно використати чергу повідомлень, до якої вхідні дані будуть поступати з різних ресурсів та записуватись у чергу. А сервіс для класифікації буде звертатися до черги повідомлень та брати всі не прочитані текстові дані та класифікувати їх. З усією схемою роботи системи можна ознайомитись на рис. 6.

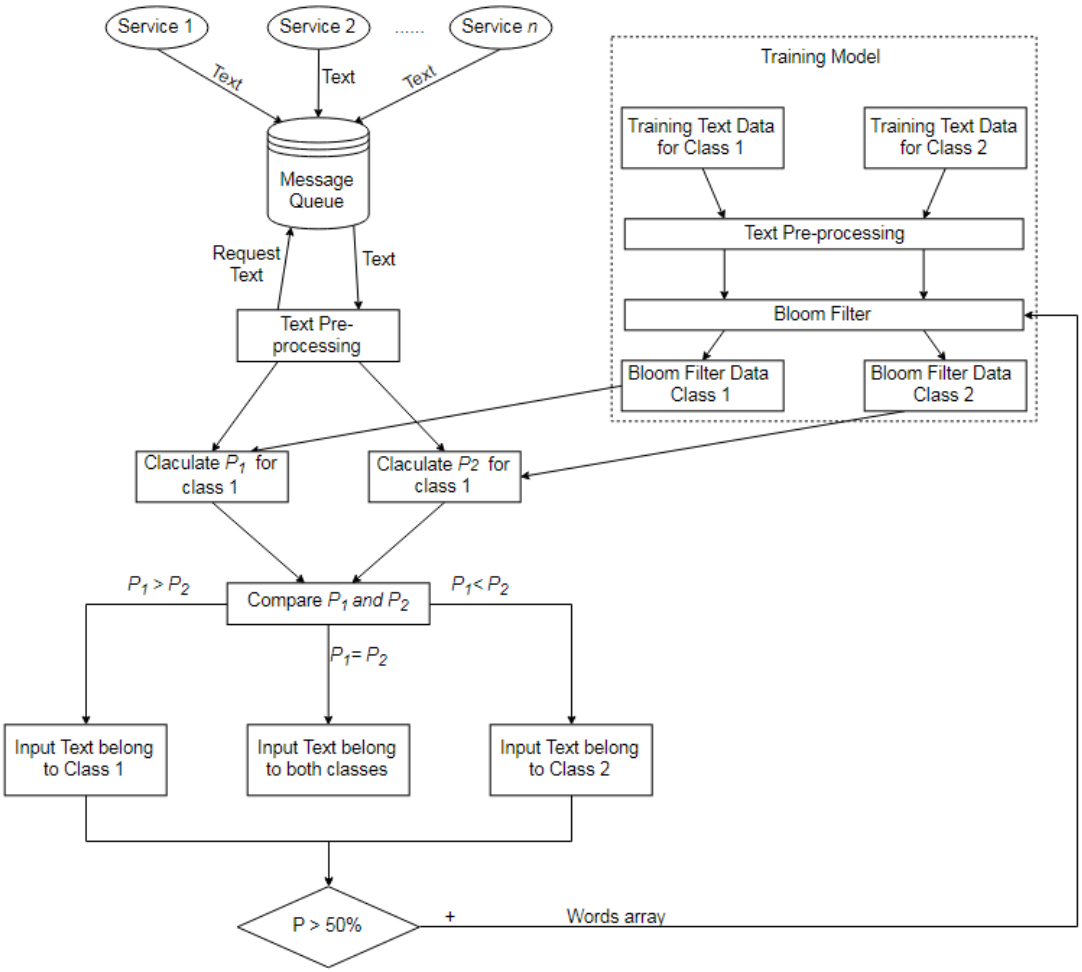

Рис.6. Робота моделі

В першу чергу навчається модель для кожного з класів на наборі текстових даних. На виході буде масив бітів фільтра Блума, які будуть використовуватися системою для підрахунку кількості слів, які проходять даний фільтр. Донавчання моделі буде відбуватися лише у випадку, коли вхідний текст належить до одного з класів та задовільняє критерій належності класу та не буде відбуватися, коли

© Яременко В. С., Будьонний Д. Ю.

вхідний текст однаково належить кожному з класів. Так як у даному випадку неможливо точно визначити, модель якого з класів донавчати.

Висновки. У даній статті було описано модель багатокласової класифікації текстів з використанням фільтра Блума для даних, які надходять у реальному часі. Було описано процес навчання моделі та проблеми, які можуть з'явитися протягом навчання. Запропоновано методи для пришвидшення та отримання більш точних результатів класифікації, а також спосіб донавчання моделі в процесі роботи класифікатора. Також були розглянуті проблеми з перенавчанням моделі та вибором складових частин фільтра Блума, що призводить до отримання хибнопозитивних результатів.

Перспективи подальших досліджень. В подальшому планується реалізація даного підходу та порівняння його з існуючими алгоритмами класифікації. Вирішення проблеми динамічного розширення виділеної пам'яті під масив бітів, що у свою чергу вирішить проблему перенавчання моделі та зменшить ймовірність хибнопозитивних результатів.

Список бібліографічних посилань

1. Role of Bloom Filter in Big Data Research: A Survey [Електронний ресурс] / Ripon Patgiri, Sabuzima Nayak, Samir Kumar Borgohain, -International Journal of Advanced Computer Science and Applications. – 2018. – Режим доступу до ресурсу: https://www.researchgate.net/publication/329327259_Role_of_Bloom_Filter_in_Big_Data_Research_A_Survey
2. Space/time trade-offs in hash coding with allowable errors / Б. Х. Блум, - Comm. of the ACM, 1970, - vol. 13, no. 7, pp. 422–426
3. Анализ Больших Наборов Данных / Ульман Дж., Раджараман А., Лесковец Ю. – ДМК Пресс, Москва, 2016. – с. 158.
4. Optimizing Bloom Filter: Challenges, Solutions, and Comparisons [Електронний ресурс] / Lailong Luo, Deke Guo, Richard T.B. Ma, Ori Rottenstreich, and Xueshan Luo. – 2018. - Режим доступу до ресурсу: https://arxiv.org/abs/1804.04777
5. A Survey of Text Classification Algorithms [Електронний ресурс] / Charu C. Aggarwal, ChengXiang Zhai. – 2012. - Режим доступу до ресурсу: https://link.springer.com/chapter/10.1007/978-1-4614-3223-4_6
6. The impact of preprocessing on text classification [Електронний ресурс] / A. K. Uysal, S. Gunal. – 2014. - Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0306457313000964
7. Text Preprocessing For Unsupervised Learning: Why It Matters, When It Misleads, And What To Do About It [Електронний ресурс] / Matthew J. Denny, Arthur Spirling. – 2018. - Режим доступу до ресурсу: https://www.cambridge.org/core/journals/political-analysis/article/text-preprocessing-for-unsupervised-learning-why-it-matters-when-it-misleads-and-what-to-do-about-it/AA7D4DE0AA6AB208502515AE3EC6989E

© Яременко В. С., Будьонний Д. Ю.