**Thema: Theoretische und praktische Aspekte eines K1 geschützten Content-Filter Systems in Verbindung mit der Entwicklung und praktischen Erprobung im Unternehmensumfeld**

**Einleitung**

Mit dem Vormarsch der neuen Technologien im Bereich der Informations- und Kommunikationstechnologie (IKT) hat die Datenmenge im Internet dramatisch zugenommen, was einige Internetsuchen ungenau werden ließ. Mit dem Aufkommen von Suchmaschinen wie Google sind die Suchanfragen jedoch einfacher und genauer geworden. Aber wie funktioniert der Suchalgorithmus von Google? Er durchsucht das Internet, um Seiten zu indexieren, sucht dann nach Seiten, die für die Suchbegriffe relevant sind und sortiert die Ergebnisse nach Relevanz für den Nutzer. So funktioniert Google, wenn man eine Suche durchführt[1]. Internetnutzer sehen sich nun aber mit Tausenden von Webseiten konfrontiert, die von einer Suchmaschine bei einer einfachen Stichwortsuche zurückgegeben werden. Die Suche unter diesen Webseiten wird für die Nutzer unmöglich. Aus diesem Grund wächst das Interesse an Werkzeugen, die dabei helfen können, eine relevante und schnelle Auswahl der gesuchten Informationen zu treffen[2].

Wir sind mit anderen Arten von Problemen konfrontiert, wie z.B. der Datenintegrität, der Manipulation von Daten durch Hacker, Websites mit gewalttätigen Inhalten oder solchen, die nur für ein bestimmtes Publikum geeignet sind, und vielen anderen Bedrohungen. Die Frage, die wir uns stellen, ist, ob es nicht eine Möglichkeit gibt, all diese Bedrohungen zu blockieren, oder die Möglichkeit, den Zugriff auf Webseiten anhand ihrer Kategorisierung zu verbieten. Dabei müssen die Kategorien nicht unbedingt bösartiger Natur sein oder in erster Linie der Strafverfolgung dienen. Die naheliegendste Lösung wäre ein System, das den Inhalt einer Webseite analysiert und sagt, zu welcher Kategorie eine Webseite gehört, um den normalen Nutzer vor potenziellen Bedrohungen zu schützen oder die Suche schneller zu machen.

1. Ziel und endgültige Lösung der Arbeit

Alles ging von einem Bedarf oder einem der oben genannten Probleme des Unternehmens Telco-Tech aus, die ein innovatives, internationales und unabhängiges Unternehmen ist, das sich auf hochwertige Sicherheitssysteme spezialisiert hat. Das Unternehmen wollte ein System, mit dem es Webseiten kategorisieren kann, um Systeme oder Benutzer besser zu schützen. Das Ziel ist es, eine Anwendung zu entwickeln, die eine URL oder Webseite (die vom Benutzer eingegeben wird) und bis zu einem gewissen Grad auch deren Unterseiten oder verwandte Webseiten (und deren verwandte Unterseiten) klassifiziert oder kategorisiert, indem sie den Inhalt der Seite analysiert und in eine Liste vordefinierter Kategorien einordnet.

Die endgültige Lösung oder das System wird wie folgt präsentiert: Das System analysiert den Inhalt der Webseite und die internen Links, die zu anderen Webseiten führen, um die richtige Kategorie zu finden. In Firewalls ist es möglich, über einen Reverse Proxy den Zugriff auf Webseiten anhand ihrer Kategorien zu verbieten. Die zu verbietenden Kategorien werden vom Administrator konfiguriert. Wenn eine Website aufgerufen wird und noch keine Kategorie im lokalen Cache gespeichert ist, fragt unser Filterserver nach der Kategorie. Für den Nutzer ist dieser Vorgang transparent, d.h. er ist nicht sichtbar.

Um dieses Problem zu lösen, werden wir uns des überwachten Lernens bedienen, denn um das Modell zu trainieren, werden wir ihm Daten zur Verfügung stellen, die eine Liste von Domainnamen mit jeweils einer zugehörigen Kategorie sind. Genauer gesagt werden wir die Klassifizierung verwenden, die eine Aufgabe in einem Problem des überwachten Lernens in der maschinellen Lehre ist. Die Ergebnisse werden anhand der Genauigkeit gemessen, die ein Maß für die Leistung eines Klassifikationsmodells im maschinellen Lernen ist. Mit anderen Worten, es ist der Prozentsatz der richtigen Antworten, die das Modell gegeben hat. Die Mindestleistung, die erforderlich ist, um das Ziel dieses Projekts zu erreichen, wird zunächst über 70% betragen, und später muss die endgültige Lösung über 90% oder besser sein.

1. Herkunft der Daten

Für dieses Projekt wurden die Daten bereitgestellt. Im Grunde handelt es sich um eine Textdatei mit einer Liste von 1 185 685 Zeilen und einer Kapazität von 206,2 MB. Sie besteht aus 257 verschiedenen Kategorien z. B. Pornografie, Shopping, Religionen, Nachrichten, Computerspiele etc. Wir wollen aber URLs haben, um auf den Inhalt der Webseiten zugreifen zu können.

Zunächst sollten wir eine Methode haben, um die Datei mit den verschiedenen Domainnamen und der entsprechenden Kategorie zu lesen, schließlich sollten wir die Domainnamen in URLs umwandeln und das Ergebnis in einer json-Datei speichern können. Der nächste Schritt besteht darin, den Inhalt der Webseiten zu erhalten. Dies wird mithilfe von Bibliotheken wie Beautifulsoup für statische Seiten, die keine menschliche Präsenz erfordern, um ihren Inhalt zu erhalten, und Selenium für dynamische Seiten, die eine menschliche Präsenz erfordern, die eine Interaktion (einen automatischen Klick auf eine Schaltfläche) erfordert, möglich sein. Das Ergebnis dieser Extraktion sollte in einer Datendatei gespeichert werden, die folgende Spalten enthält: Name, Kategorie, URL, Wörter, die auf der Seite enthalten sind, und Sprache. Anschließend wird die Datei in eine CSV-Datei umgewandelt. Dieser Schritt wird als Scraping bezeichnet.

1. Umgebung der Arbeit

Um dieses Projekt durchzuführen, benötigt man natürlich eine ausreichend leistungsfähige Arbeitsumgebung und genügend Speicherplatz. In unserem Fall werden wir einen Desktop-Computer mit Ubuntu 22.04.1 LTS, einem i8-Prozessor mit 3,7 GHz und 1,5 TB Speicherplatz verwenden. Python 3.11 als Programmiersprache , Jupyter Notebook als Programmierumgebung und schließlich die Bibliotheken für Machine Learning und Scraping.

**Datensammlung**

Dieser Teil wird nicht nur die Methoden behandeln, mit denen ich die Daten aus den Webseiten erhalten habe, sondern auch die Wahl der HTML-Elemente.

1. URL-Bearbeitung

Eine URL(Uniform Resource Locator) ist eine Zeichenfolge, mit der eine Ressource im Internet, wie z. B. eine Webseite, eine Datei, ein Bild usw., lokalisiert werden kann. Sie besteht aus mehreren Teilen, darunter das Protokoll (http, https, ftp usw.), der Domainname, der Pfad und der Dateiname.

Es besteht in der Regel aus 6 Teilen, z.B. auf dieser URL: **http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#somewhereinTheDocument**, wir werden haben:

* Scheme(**http**): er gibt das Protokoll an, das der Browser verwenden soll, um die Ressource anzufordern (ein Protokoll ist eine definierte Methode zum Austausch oder zur Übertragung von Daten über ein Computernetzwerk). Bei Websites ist das Protokoll in der Regel HTTPS oder HTTP (seine ungesicherte Version)[3].
* Authority(**www.example.com:80**): Er besteht aus dem Domainnamen (www.exemple.com) und dem Port (80).
  + Der Domainname gibt an, welcher Webserver angesprochen wird. Er wird oft unter einer IP-Adresse verwendet (was aber selten der Fall ist, da es viel unpraktischer ist).
  + Der Port gibt die technische "Tür" an, die für den Zugriff auf die Ressourcen des Webservers verwendet wird. Er wird normalerweise weggelassen, wenn der Webserver die Standardports des HTTP-Protokolls (80 für HTTP und 443 für HTTPS) verwendet, um den Zugriff auf seine Ressourcen zu ermöglichen. Andernfalls ist er obligatorisch.[3]
* Pfad zur Ressource(**path/to/myfile.html**): es ist der Pfad zu der Ressource auf dem Webserver. In den frühen Tagen des Webs stellte ein solcher Pfad einen physischen Speicherort einer Datei auf dem Webserver dar. Heute ist er hauptsächlich eine Abstraktion, die von den Webservern verwaltet wird, ohne physische Realität.[3]
* Parameter(**key1=value1&key2=value2**): Dies sind zusätzliche Elemente, die dem Webserver zur Verfügung gestellt werden. Diese Parameter sind eine Liste von Schlüssel/Wert-Paaren, die durch das Symbol & getrennt sind. Der Webserver kann diese Parameter verwenden, um zusätzliche Operationen durchzuführen, bevor er die Ressource zurückgibt. Jeder Webserver hat seine eigenen Regeln bezüglich der Parameter, und der einzige zuverlässige Weg, um herauszufinden, ob ein bestimmter Webserver die Parameter verwaltet, ist, den Besitzer des Webservers zu fragen.
* Anchor(**#somewhereinTheDocument**): es ist ein weiterer Teil der Ressource . Er stellt eine Art "Lesezeichen" innerhalb der Ressource dar und gibt dem Browser Anweisungen, wie er die Inhalte an dieser "markierten" Stelle anzeigen kann. In einem HTML-Dokument scrollt der Browser beispielsweise bis zu der Stelle, an der der Anker gesetzt ist; in einem Video- oder Audiodokument versucht der Browser, zu dem Zeitpunkt zu gelangen, den der Anker repräsentiert. Es ist zu beachten, dass der Teil nach dem #, der auch als Fragment Bezeichner bekannt ist, niemals mit der Anfrage an den Server gesendet wird.

Das Ergebnis ist, dass ein Domainname ein Teil einer URL ist, was bedeutet, dass der Domainname allein uns nicht dabei helfen kann, Informationen über Webseiten mit Hilfe von Scraping-Bibliotheken zu sammeln. Um dies zu tun, müssen wir zuerst die Vorverarbeitung von Domainnamen in URLs durchlaufen.

wie bereits erwähnt, haben wir eine Datei mit den Domainnamen wiederhergestellt. Als Domainnamen haben wir zum Beispiel **www.heise.de**, **api.livestrong.com**, **dns-dv6irbpf1.sombrero.yahoo.net**. Um sie in eine URL umzuwandeln, müssen wir nur ein Protokoll (http oder htpps) oder ein Schema hinzufügen. Da es sich um ein htpp-Protokoll handelt, wird der Port 80 sein und 443 für https und wird automatisch gesetzt. Da wir nicht auf eine bestimmte Ressource auf der Webseite zugreifen wollen, wird der Pfad zu dieser Ressource weggelassen. Dies gilt auch für die Parameter und Anchos.

Das Programm, das wir schreiben werden, funktioniert folgendermaßen, um Domainnamen in URLs umzuwandeln. Zunächst muss es jeden Domainnamen aus der Datei lesen, ihn in eine URL umwandeln (indem es das Schema hinzufügt) und dann eine Anfrage an den Server stellen, um herauszufinden, ob dieser die Anfrage des Clients, die Webseite (den Inhalt) zu liefern, verarbeiten kann. Nur diejenigen, die einen HTTP-Response-Code 200 haben, werden beibehalten und in einer JSON-Datei gespeichert. Es gibt auch andere HTTP-Antwortcodes, die die Zugänglichkeit der aufgerufenen Webseite nicht ausschließen. Am Ende erhalten wir eine kleinere Liste von eindeutigen URLs, die einen HTTP-Antwortcode von 200 haben. Da es mehr als eine Million Zeilen in dieser Datei gibt, werden wir mehrere Threads verwenden, um die Ausführung schneller zu machen.

1. Webseiten crawlen

Nachdem wir die Domainnamen umgewandelt haben, müssen wir nun die Informationen aus der Webseite abrufen. Dies wird mithilfe eines Programms geschehen, das wir schreiben werden und das den Inhalt jedes HTML-Elements mithilfe der Beautifullsoup-Bibliotek für statische Seiten und Selenium für dynamische Seiten abrufen wird. Als HTML-Elemente werden wir die Tags Titel (der im Header) und Body-Titel, die Wörter der Meta-Beschreibung und den Body auswählen. Gemäß den Empfehlungen des Unternehmens sollten wir nur Webseiten mit Inhalten in Deutsch, Englisch und Französisch in Betracht ziehen. Die meisten Webseiten enthalten ein Attribut language im Header, dass die verwendete Sprache anzeigt. Es gibt jedoch eine Bibliothek namens langdetect, die es ermöglicht, die verwendete Sprache schnell zu identifizieren. Anschließend werden die gesammelten Textinformationen mit Hilfe der Spacy-Bibliothek verarbeitet, lemmatisiert (dies ist eine Technik zur automatischen Verarbeitung natürlicher Sprache, bei der die verstümmelten Formen eines Wortes normalisiert werden, um seine kanonische Form zu erhalten), Stopwords entfernt (Stopwords sind Wörter, die keine ausreichende Bedeutung haben, um den Textinhalt zu verstehen) und alles in Kleinbuchstaben formatiert, um Unterschiede in der Groß- und Kleinschreibung zu vermeiden.

Am Ende erhält man eine CSV-Datei, die fünf Spalten enthält: eine Spalte mit Namen, Kategorie, URL, Wort und Sprache. Der nächste Schritt besteht darin, die erhaltenen Texte zu verarbeiten. Dies beinhaltet die Überprüfung auf fehlende Wörter, Sonderzeichen, Seiten, die vorübergehend gesperrt waren oder sich noch im Aufbau befinden, und die Entfernung von Duplikaten.

**Bibliografische Referenzen**

[1] S. Brin und L. Page, „The Anatomy of a Large-Scale Hypertextual Web Search Engine“, *Comput. Netw.*, Bd. 30, S. 107–117, 1998.

[2] „Webpage Classification Student Affairs System For College of science Al Zulfi Department of Computer Science and Information - Google leiting“. https://www.google.com/search?channel=crow5&client=firefox-b-d&q=Webpage+Classification+Student+Affairs+System+For+College+of+science+Al+Zulfi+Department+of+Computer+Science+and+Information (zugegriffen 11. Mai 2023).

[3] „What is a URL? - Learn web development | MDN“, 23. Februar 2023. https://developer.mozilla.org/en-US/docs/Learn/Common\_questions/Web\_mechanics/What\_is\_a\_URL (zugegriffen 13. Mai 2023).