Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

TextBlob

SEMESTRÁLNÍ PRÁCE

4IZ470 – Dolování znalostí z webu

Autor: Bc. Jonáš Matějka

Vyučující: prof. Ing. Vojtěch Svátek, Dr.

Praha, červen 2024

Obsah

Úvod

Už uběhlo několik desetiletí, co se Alan Turing podílel na sestavení prvního počítače a s obrovským posunem technologie dokáží počítače a inteligentní stroje mnohem více. V dnešní době je již zcela normální, že jsou lidé schopni vést s počítačem smysluplnou konverzaci, téměř na takové úrovni, že by člověk ani nemusel poznat, zda komunikuje právě s počítačem nebo jiným člověkem. Tato komunikace stojí na zpracování přirozeného jazyka (Natural language processing), která umožňuje počítači nebo počítačovému programu zpracovat vstup v podobě jazyka, kterým mluví lidé (přirozený jazyk) a na základě toho vygenerovat výstup v lidském jazyce dávající smysl a dodržující kontext vstupu např. odpověď na otázku.

Zpracování přirozeného jazyka se využívá ve více oblastech, ve kterých je potřeba pracovat s lidským jazykem jako je strojový překlad (čeština do angličtiny), generování textu a mluveného slova (chatboti, virtuální asistent od Apple Siri) nebo porozumění textu. Právě porozumění textu bude oblastí zaměření této semestrální práce. Jedním z cílů porozumění textu je získat informaci jaký má text emocionální tón, který bývá pozitivní, negativní nebo neutrální. Na základě emocionálního tónu textu můžeme zjistit jaký má autor textu postoj ke danému tématu. To lze následně využít v různých oblastech jako je digitální marketing, zkvalitňování produktů a služeb nebo různé analýzy mínění lidí ke konkrétnímu tématu. Tento proces se nazývá analýza sentimentu nebo také dolování názoru (opinion mining).

Cíle práce

Smyslem této práce je prozkoumat možnosti a funkce nástroje pro zpracování přirozeného jazyka v textové podobě python knihovny TextBlob. K tomuto nástroji jsem se dostal při hledání nástrojů pro analýzu sentimentu. Na mnoha webech byl TextBlob popsán jako jednoduchá knihovna pro začátečníky v oblasti zpracování přirozeného jazyka, a proto jsem se rozhodl jej zvolit. Cílem této práce bude otestovat knihovnu na datech z platformy twitter o problémech velkých amerických leteckých společností. Analýzu bych chtěl provést dvěma rozdílnými způsoby. Nejprve využít základní možnosti knihovny a následně vyzkoušet natrénování a analýzu sentimentu pomocí klasifikačních modelů strojového učení nabízených knihovnou.

Metoda dosažení cíle

Na volně dostupném datasetu obsahující příspěvky z twitteru a jejich ohodnocený sentiment budou vyzkoušeny možnosti knihovny TextBlob pro pochopení její funkčnosti a nedostatků. Dataset bude podroben lingvistickému předzpracování a následně analýzám sentimentu se základní funkčností a s využitím klasifikačních modelů knihovny TextBlob. Výsledky budou poté zhodnoceny a případně navržena doporučení.

Předpoklady a omezení práce

Důležitým předpokladem této práce bylo zvolit dobrá data. Při psaní jsem zkoušel několik různých datasetů, nicméně kvalita dat byla nízká. V mnoha případech chyběly hodnoty nebo data obsahovala spoustu duplikátů a TextBlob si s daty nevěděl rady. Nejvíce omezující faktor byla schopnost TextBlob knihovny zpracovat větší množství dat. Modely na datech s více než 10 000 záznamy měly problém data zpracovat. Výpočty trvaly několik desítek minut a výsledek byl neuspokojivý (přesnost 0.0).

Souvislost s jiným semestrálními nebo kvalifikačními pracemi

Tato semestrální práce nemá textový ani věcný překryv, ani jinou věcnou souvislost, s jinými semestrálními nebo kvalifikačními pracemi, které jsem zpracovával.

TextBlob – hlavní principy fungování

TextBlob je open source knihovna naprogramovaná v jazyce python sloužící jako nástroj pro zpracování přirozeného jazyka v textové podobě. TextBlob je založený na přední platformě pro vytváření programů v python pro práci s lidskými daty NLTK a využívá jeho rozhraní pro více než 50 korpusů a lexikálních zdrojů. TextBlob přejímá některé funkce z NLTK a ty pak upravuje pro jednodušší použití ze strany uživatele. [1]

Základním kamenem pro práci s TextBlob knihovnou je využití *TextBlob()* objektu (instance třídy) představující textový řetězec. Operace s tímto řetězcem jsou stejné jako s obyčejným textovým řetězcem v jazyce python, nicméně *TextBlob()* je obohacen o možnosti zpracování přirozeného jazyka. S instancí třídy reprezentující textový řetězec lze aplikovat následující výčet nejdůležitějších úloh:

- Noun phrase extraction
- Part-of-speech tagging
- Sentiment analysis
- Classification
- Tokenization
- Word and phrase frequencies
- n-grams
- Word inflection (pluralization and singularization) and lemmatization
- Spelling correction

Noun phrase extraction

TextBlob provádí extrakci podstatných jmenných frází na základě POS Taggingu. Po tokenizaci vstupu je každé slovo v textu označeno gramatickou kategorií (sloveso, podstatné jméno, přídavné jméno atd.) a následně se s využitím regulárních výrazů identifikují podstatné jmenné fráze podle jejich POS tagů, které se extrahují.

Typická pravidla mohou vypadat takto:

- Podstatné jméno následované přídavným jménem (např. "natural language")
- Přídavné jméno následované podstatným jménem (např. "great tool")

Part-of-speech tagging

Tato vlastnost vrací seznam dvojic ve tvaru datové struktury tuple *(token, POS tag)*. Zde TextBlob využívá NLTK knihovnu, do které posílá text již převedený na tokeny a POS tagger z NLTK přiřadí každému tokenu POS tag. POS tagger je založen na Penn Treebank tag set, což je široce používaná sada tagů pro angličtinu. [2]

Příklady jednotlivých taggů:

- NN – podstatné jméno
- VB – sloveso v základním tvaru
- VBD – sloveso v minulém tvaru
- DT – člen

Sentiment analysis

Analýza sentimentu textu probíhá na základě metriky **polarity**, kde každému slovu je podle sentimentálního slovníku, který je převzat z knihovny Pattern [3] přiřazena právě jedna číselná hodna v rozmezí od -1 do 1, kde:

- -1 představuje negativní polaritu
- 0 představuje neutrální polaritu
- 1 představuje pozitivní polaritu

Při hodnocení sentimentu je hodnoceno každé slovo (token) zvlášť a následně jsou hodnoty jednotlivých tokenů skombinovány a určen celkový sentiment textu. K analýze sentimentu TextBlob také umožňuje přiřadit textu hodnotu subjektivity, která značí, jak moc se daný text opírá o fakt nebo jak moc je text subjektivní názor. Vyhodnocení probíhá totožně, jen rozmezí přižení hodnoty subjektivity slovu se pohybuje mezi 0 a 1, kde:

- 0 text je objektivní (opřen o fakta)
- 1 text je subjektivní (názor)

Classification models

TextBlob nabízí několik klasifikačních modelů, které je možno natrénovat a následně používat pro klasifikaci textu. Vstupem do modelu je trénovací množina dat v datové struktuře tuple (základní python), vypadající takto: (*klasifikovaný text, sentiment)*. Klasifikátory také umožňují generaci jednotlivých metrik a nalezených znalostí.

Tokenization

Pro tokenizaci vět a slov TextBlob využívá NLTK knihovnu, ve které volá *sent_tokenize* a *word_tokenize* metody, které rozdělí text podle určitých pravidel na tokeny vět a slov.

Při tokenizaci vět funkce identifikuje koncové znaky vět (tečky, otazníky atd.) s použitím pravidel, aby byl zajištěn kontext a znak opravdu ukončoval větu (tečka za zkratkou není konec věty). Modely mají natrénovaná pravidla z trénovacího korpusu textu od NLTK.

Při tokenizaci slov se nejprve identifikují mezery a oddělovače slov. Dále interpunkční znaky a v poslední řadě mohou být využity regulární výrazy pro oddělení zkratek, čísel nebo jiných speciálních sekvencí.

Word and phrase frequencies

Počítání frekvence slov a frází se provádí na základě tokenizace. Vstupní text je převeden na list tokenů a pro zjištění frekvence slov v textu se použije metoda *count(„token“)* se vstupním parametrem tokenu, jehož frekvenci chceme zjistit. Metoda následně porovná vstupní parametr s každým tokenem v textu a zaznamená si počet jeho výskytů.

n-grams

Tvorba n-gramů v TextBlob probíhá také na základě tokenizace. Text v *TextBlob()* je převeden na list tokenů a podle zadaného parametru *n* vrací metoda *ngrams(n=3)* n-tici po sobě jdoucích tokenů (slov) z textu.

Word inflection (pluralization and singularization) and lemmatization

Při úpravě slov TextBlob kromě lemmatizace slov, kdy jsou jednotlivá slova převedena na jejich základní tvar („běží“ má lemma „běžet“), také umožňuje provádět pluralizaci a singularizaci. Metoda pro pluralizaci převádí jednotné podstatné jméno na množné např. „auto“ na „auta“. Singularizace provádí opak, převádí množné podstatné jméno na množné „auta“ na „auto“.

Spelling correction

Natrénovaný pravděpodobnostní model porovnává vstupní slova s jeho slovníkem a v případě detekce nesprávně napsaného slova se na základě editační vzdálenosti snaží najít nejbližší shodu, kterou navrhne jako opravenou verzi slova.

Experiment analýza sentimentu

V této části semestrální práce bude proveden experiment analýzy sentimentu na volně dostupném datasetu z webu www.kaggle.com.

Dataset

Dataset jsem si vybral na základě kladného hodnocení uživatelů a přispěvatelů webu www.kaggle.com. Data byla původně získána webovým scrapingem z platformy twitter v únoru roku 2015 a v mírně upravené podobě poskytnuta volně ke stažení na webu www.kaggle.com. Data obsahují informace o problémech velkých amerických leteckých společností. Přispěvovatelé datasetu byli požádání, aby nejprve klasifikovali pozitivně/negativně/neutrálně jednotlivé příspěvky z twitteru (tweety) k dané společnosti a poté kategorizovali negativní důvody příspěvků (např. „late flight“, „rude service“). [4]

Dataset tvoří **14 640** záznamů (řádky) s **15** sloupci (vlastnosti, atributy). Z přehledu datasetu (Obrázek 1) lze vidět, jaké informace o příspěvcích jsou v datasetu uchovány. Jsou to převážně informace o charakteristice příspěvku jako jeho identifikace, čas, kdy byl napsán, kdo ho napsal nebo k jaké společnosti se příspěvek váže. Pro účely mého experimentu využiji z dat pouze 3 sloupce a to:

- tweet_id – identifikační číslo příspěvku (celé číslo)
- airline_sentiment – sentiment klasifikovaný přispěvovateli datasetu (negative/neutral/positive)
- text – obsah příspěvku (textový řetězec)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 15 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   tweet_id                      14640 non-null  int64
 1   airline_sentiment             14640 non-null  object
 2   airline_sentiment_confidence  14640 non-null  float64
 3   negativereason                9178 non-null   object
 4   negativereason_confidence     10522 non-null  float64
 5   airline                       14640 non-null  object
 6   airline_sentiment_gold        40 non-null     object
 7   name                          14640 non-null  object
 8   negativereason_gold           32 non-null     object
 9   retweet_count                 14640 non-null  int64
 10  text                          14640 non-null  object
 11  tweet_coord                   1019 non-null   object
 12  tweet_created                 14640 non-null  object
 13  tweet_location                9907 non-null   object
 14  user_timezone                 9820 non-null   object
dtypes: float64(2), int64(2), object(11)
memory usage: 1.7+ MB
```

Obrázek 1 Přehled datasetu

Předzpracování dat

Pro provedení experimentu analýzy sentimentu, jsem musel data nejprve prozkoumat a předzpracovat. Jak jsem již zmínil v popisu datasetu, pro účely mého experimentu využiji pouze 3 sloupce i původního datasetu (*id, airline_sentiment, text*). Po filtraci datasetu bylo potřeba zkontrolovat, zda se v datech nevyskytují nulové hodnoty jako prázdný obsah tweetu (sloupec *text*) nebo nevyplněná klasifikace sentimentu (sloupec *airline_sentiment*). Žádný sloupec nulové hodnoty neobsahuje, tudíž k analýze lze využít všech 14 640 záznamů.

Následně jsem vytvořil metodu pro normalizaci textu, kterou jsem použil na *text* každého záznamu datasetu a výsledný předzpracovaný text jsem uložil do nově vytvořeného sloupce *text_clean*, který pak využiji u klasifikačních modelů.

```python
def normalize_text(txt):
    # Lower casing
    txt = txt.lower()

    # Removing URLs
    txt = re.sub(r'http\S+|www.\S+', '', txt)

    # Removing HTML tags
    txt = re.sub(r'<.*?>', '', txt)

    # Remove punctuation
    txt = re.sub(r'[^\w\s]', '', txt)

    # Remove numbers
    txt = re.sub(r'\d+', '', txt)

    # Remove extra spaces
    txt = ' '.join(txt.split())

    # Replacing repetitions of punctuation
    txt = re.sub(r'(\W)\1+', r'\1', txt)

    # Converting emojis to text
    txt = emoji.demojize(txt)

    # Remove special characters once again
    txt = re.sub(r"[^\w\s]", '', txt)

    # Remove contractions
    txt = contractions.fix(txt)

    txt = TextBlob(txt)
    # Tokenization and stopwords removal
    txt_tokens = [word for word in txt.words if word not in stopwords.words('english')]

    # Lemmatization
    lemmatized_tokens = [word.lemmatize() for word in txt_tokens]

    return ' '.join(lemmatized_tokens)
```

Obrázek 2 Metoda pro normalizaci textu

Vstupní parametr metody je textový řetězec, jehož znaky jsou v prvním kroku úprav převedeny na malá písmena. Následně je řetězec podroben různým úpravám textu tak, aby výstupem metody byly jen lemmatizované tokeny.

Vstupní text je pomocí regulárních výrazů očištěn o URL adresy, HTML tagy, interpunkčních znamének, číslic a speciálních znaků. Úprava probíhá tak, že regulární výraz najde v řetězci shodu, kterou nahradí nevyplněným znakem (‚‘), tudíž shodu odstraní.

V původních datech se v příspěvcích vyskytovaly různé smajlíky (anglicky emoji) pro hlubší vyjádření názoru, a proto jsem se rozhodl i tyto informace využít při analýze a pomocí knihovny *emoji*, která má definovanou databázi smajlíků a jejich kódů jsem na celý řetězec využil metodu *demojize*, která identifikuje na základě své databáze emoji a následně jej převede do textové podoby. Například klasický smějící se smajlík byl převeden do tvaru: „:grinning_face:“.[5]

V neposlední řadě využívám metodu *fix* z knihovny *contractions*, která na základě své databáze zkratek pomáhá odstranit zkrácená slova v angličtině jako „you’re“ na „you are“. Nevýhoda této knihovny může ovšem nastat, dojde-li ke konfliktu kdy knihovna nebude vědět, jak si se zkratkou poradit („he’s“ na „he is“ nebo „he has“?). V takovém případě knihovna vrátí nejběžnější případ, tedy „he is“.[6]

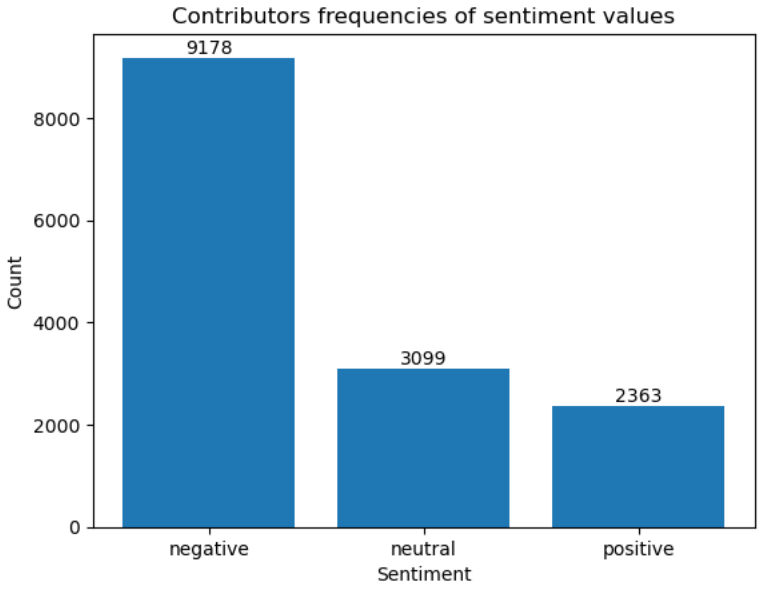
Normalizaci textu zakončuji již využitím TextBlob a jeho funkcemi pro tokenizaci a lemmatizaci, které byly popsány v hlavních principech knihovny TextBlob. Při tokenizaci řetězce využívám seznam *stopwords* z knihovny NLTK, získaný z jejího korpusu, pro odstranění stop slov. Metoda tedy vrací předzpracovaný tokenizovaný a lemmatizované text, který ukládám do nového sloupce datasetu *text_clean*.

Výstupem předzpracování dat je dataset s předzpracovaným textem pro analýzu.

Analýza

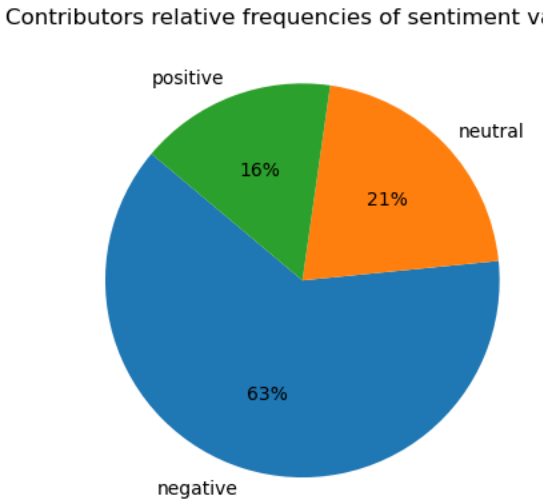
Analýzu sentimentu jsem se rozhodl udělat dvěma způsoby. Při prvním způsobu bych chtěl využít základní možnosti knihovny TextBlob, při druhém bych chtěl vyzkoušet natrénovat klasifikační modely které knihovna nabízí a zhodnotit jejich výkon.

Pro porovnání analýzy sentimentu pomocí TextBlob jsem vytvořil graf zastoupení jednotlivých postojů sentimentu podle jejich četnosti v datasetu (viz Obrázek 3). Jak již bylo zmíněno při představení datasetu, klasifikaci příspěvků prováděli lidé, kteří se podíleli na tvorbě datasetu, tudíž se nejedná o klasifikaci na základě stroje nebo výpočtů.



Obrázek 3 Četnosti zastoupení sentimentů v datasetu

Pro zobrazení relativní četnosti sentimentu v datasetu jsem vytvořil ještě koláčový graf (Obrázek 4).



Obrázek 4 Relativní četnosti zastoupení sentimentů v datasetu

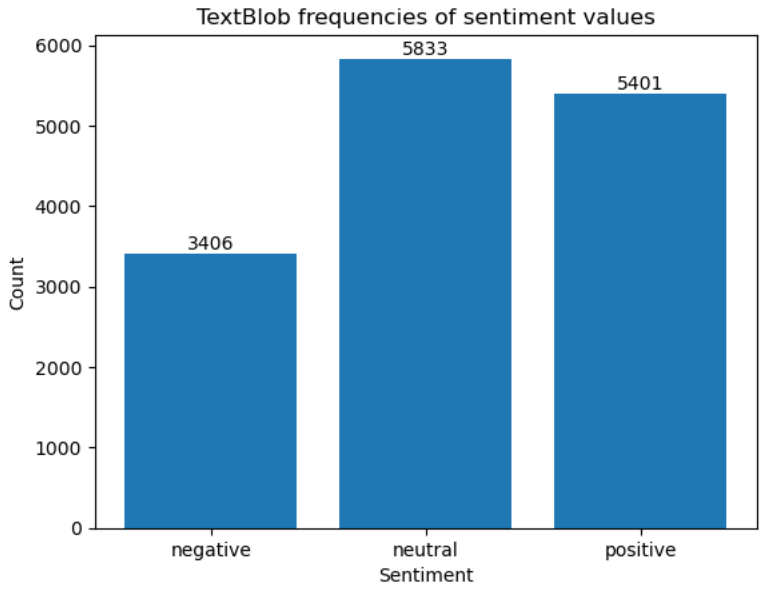
Analýza se základním rozhraním TextBlob

Jak již bylo vysvětleno TextBlob využívá metriku polarity, kterou přiřadí každému slovu, které najde ve slovníku sentimentu přejatého z knihovny Pattern. Při tomto experimentu jsem se chtěl zaměřit na analýzu sentimentu každého příspěvku zvlášť a výsledek porovnat s opravdovou hodnotou, kterou určili přispěvovatelé. [7]

Analýza je tedy založená čistě na slovníku sentimentu, který TextBlob využívá. Za účelem této analýzy jsem se rozhodl přiřadit sentiment textu podle následujících pravidel:

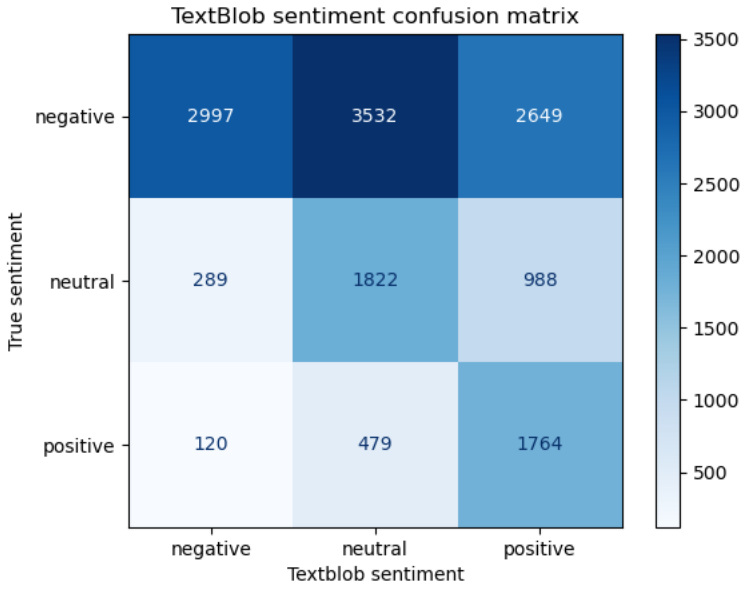
- když je polarita větší než 0 sentiment je klasifikován jako pozitivní
- když je polarita menší než 0 sentiment je klasifikován jako negativní
- když je polarita rovna 0 sentiment je klasifikován jako neutrální

Každému záznamu jsem podle polarity předzpracovaného textu vyhodnotil sentiment a výsledek je zobrazen na sloupcovém grafu (Obrázek 5).



Obrázek 5 Analýza sentimentu na základě TextBlob polarity

V porovnání s původním sentimentem určeným lidmi je vyhodnocení TextBlob velmi odlišné. Největší skupina sentimentu je neutrální. Na dalším obrázku matice záměn (Obrázek 6) lze vidět, kolik chybných určení bylo TextBlob provedeno. Analýza, kterou TextBlob provedl s využitím svého slovníku má přesnost pouze **45 %**. Model často chyboval v určení negativního sentimentu a upřednostňoval spíše neutrální a pozitivní sentiment, jejichž počet téměř zdvojnásobil.



Obrázek 6 Matice záměn TextBlob analýzy sentimentu

Důvodem těchto chybností je může být, že slovník použitý na analýzu nemusí nutně obsahovat sentiment pro veškerá slova, která jsou předmětem klasifikace, speciálně pokud se jedná o nějakou oblast (letectví), která může mít svůj slang a speciální slova. Dalším důvodem špatné klasifikace může být způsob, jakým TextBlob vyhodnocuje sentiment, protože pokud je ve větě obsaženo nějaké negativní slovo, které může být zesíleno negativním přídavném jménem, vynásobí se jejich negativní sentiment a výsledek je kladný. Jak může vypadat takový výpočet polarity je ukázáno níže (Obrázek 7). [7, 8]

$$polarity = (-1) \times p_{--} + (-0.5) \times p_{-} + 0 \times p_{0} + 0.5 \times p_{+} + 1 \times p_{++}$$

Obrázek 7 Ukázka výpočtu polarity

Analýza s využitím klasifikátorů

Při pokusu analýzy sentimentu za využití dostupných klasifikátorů knihovnou TextBlob jsem se rozhodl vyzkoušet NaiveBayesClassifier, MaxEntClassifier, DecisionTreeClassifier. Abych mohl natrénovat jednotlivé modely bylo potřeba předzpracovaná data rozdělit na trénovací a testovací množinu. Tento poměr jsem rozdělil 70 % ku 30 %. Následně bylo potřeba obě množiny přetransformovat do datové struktury *list*, kde každý jeden záznam je reprezentován tuplem (*text, sentiment*).

NaiveBayesClassifier

Klasifikační algoritmus NaiveBayesClassifier generuje během trénování pro každé slovo z tréninkového korpusu samostatný prvek, který lze považovat za jeho slovník. Pokud se ve větě nachází jedno z příznakových slov, pak bude mít jedna z tříd vyšší pravděpodobnost a sentiment nebude neutrální. V případě absence příznakového slova budou

pravděpodobnosti tříd (pozitivní/negativní) stejné a polarita bude neutrální. Tento klasifikátor má problém s rozpoznáním negativního sentimentu, což způsobuje předpoklad podmíněné nezávislosti. Ten říká, že pravděpodobnost nalezení určitého slova závisí pouze na třídě textu a nikoli na ostatních slovech v textu. Pokud se tedy „not“ objeví v trénovacím datasetu ve více pozitivních případech než v negativních, přispěje k pozitivnímu sentimentu bez ohledu na to, jaká slova jsou za ním. [7]

MaxEntClassifier

Klasifikátor maximální entropie fungující na základě odhadu pravděpodobnosti, zda určitý vzorek textu patří do určité třídy (má určitý sentiment). Tento klasifikátor je parametrizován sadou vah, které se používají ke kombinaci společných znaků, jež jsou generovány ze sady znaků pomocí kódování. Kódování mapuje každou dvojici (množinu rysů tedy vstupní text, sentiment) na vektor. Zmíněná pravděpodobnost pro odhad se počítá podle následujícího vzorce. [1]

$$\text{prob}(fs|label) = \frac{\text{dotprod}(weights, \text{encode}(fs,label))}{\sum(\text{dotprod}(weights, \text{encode}(fs,l)) \text{ for } l \text{ in } labels)}$$

Obrázek 8 Vzorec pro výpočet odhadu pravděpodobnosti MaxEntClassifier

DecisionTreeClassifier

Algoritmus fungující na principu strojového učení, kde se při trénování model snaží zjistit hlavní vlastnosti (vlastnost s největší informační hodnotou) podle kterých klasifikuje jednotlivé prvky do tříd. Z pohledu analýzy sentimentu se jedná o klasifikaci textu do „třídy“ sentimentu na základě vlastností textu, tedy jaké fráze se v textu vyskytují apod. [9]

Vyhodnocení modelů

V této kapitole bych rád shrnul vyzkoušené klasifikační modely. Při mých experimentech jsem se potýkal s problémem, že modely se nedokázaly provést vyhodnocení na celou množinu dat. Jediný model, který zvládl zpracovat všech původních 14 640 záznamů byl NaiveBayesClassifier, který měl přesnost na testovacích datech 78 %. V případě MaxEntClassifier vyšla přesnost 16 % nicméně v průběhu validace modelu na testovacích datech byl výpočet několikrát přerušen z důvodu složitých výpočtů, které python nebyl schopen provést. Poslední DecisionTreeClassifier jsem nechal vyhodnocovat hodinu čistého času, nicméně bez výsledku.

I přesto, že měl NaiveBayesClassifier vcelku dobrou přesnost, při mém testovacím pokusu vyhodnotit pozitivně větu „Today was a good flight“ ji přiřadil negativní sentiment. Ovlivnění této klasifikace může vzniknout z podstaty dat. Dataset který jsem zvolil obsahuje převážně negativní sentiment a po podrobnějším náhledu na data je obtížné najít nějaký příspěvek, který by nebyl stížnost na leteckou společnost, ale naopak chválil let.

Po nezdaru vyhodnocování natrénovaných modelů na velkém objemu dat, jsem se rozhodl data zmenšit na 1 000 záznamů, avšak podíly sentimentů v datasetu jsem nastavil stejně, aby relativní četnost reprezentovala původní dataset, tedy 61 % negativní, 21 % neutrální a 16 % pozitivní sentiment.

Z tabulky přesnosti modelů (Tabulka 1) je zřejmé, že zmenšení objemu, výrazně pomohlo klasifikačním modelům zpracovat data. Výpočetní doba se rapidně snížila a výsledky zůstaly podobné s pozitivní změnou, že se podařilo vyhodnotit i DecisionTreeClassifier, který má vcelku dobrou přesnost. NaiveBayesClassifier se mírně zhoršil, nicméně to zapříčiňuje můj kód, který vybere každé spuštění jiný vzorek 1 000 záznamů z původního datasetu. Tato přesnost po několika spuštěních kolem 70 % oscilovala.

Po tomto zlepšení jsem se rozhodl vzorek dat zmenšit ještě jednou, a to stejným způsobem na 100 záznamů. Z výsledků plyne, že NaiveBayesClassifier funguje na datech obstojně s vysokou pravděpodobností. Přesnost modelu DecisionTreeClassifier je nestabilní a pro jiný objem dat generuje velmi rozdílné hodnoty přesnosti. Pozitivním zlepšením po nastavení vzorku na 100 záznamů je MaxEntClassifier, který byl při tomto malém objemu dat schopen provést vyhodnocení ve 100 iterací.

Tabulka 1 Přesnosti modelů

Model	**Celý dataset**	**1 000 záznamů**	**100 záznamů**
NaiveBayesClassifier	78 %	70 %	70 %
MaxEntClassifier	16 %	14 %	67 %
DecisionTreeClassifier	x	68 %	47 %

Závěr

Práce byla zaměřena na otestování open source python knihovny TextBlob při analýze sentimentu na zvoleném datasetu.

V úvodní části byla nastíněna problematika, cíl a metody dosažení cíle. Cílem bylo otestovat knihovnu TextBlob při analýze sentimentu na volně dostupném datasetu obsahující příspěvky z twitteru o problémech velkých amerických letových společností.

Dále byla představena knihovna TextBlob a její základní principy fungování a nabízené funkce.

V poslední části práce byl proveden experiment analýzy sentimentu na konkrétním datasetu. Prvně byl představen dataset, jeho zdroj a charakteristiky. Následně byla data předzpracována a podrobena dvěma experimentům. První experiment využíval základní funkce TextBlob pro analýzu sentimentu, druhý se zabýval vytvořením klasifikačních modelů a jejich následné evaluace.

S knihovnou TextBlob se mi pracovalo vcelku dobře a z kapitoly, kde je definován cíl mohu jen potvrdit, že je jednoduchá na použití a vhodná pro začátečníky v oblasti zpracování přirozeného jazyka. Nedostatkem knihovny se mi místy osobně zdála dokumentace, protože jsem častokrát nedokázal vyhodnotit, jak daný kus kódu funguje, zejména u klasifikačních modelů. U základních funkcí byla dokumentace v pořádku.

Osobně bych knihovnu nedoporučil zkušenějším lidem v oboru zpracování přirozeného jazyka, protože jsem se potýkal s velkými výpočetními problémy, i přesto že mám velmi výkonný hardware. Některé klasifikační modely trvaly desítky minut, a to jsem měl „jen“ okolo 14 000 záznamů. Mým původním plánem bylo použít dataset s 70 000 záznamy, ale při prvních testech se mi nápad zdál zcela nereálný.

Použitá literatura

[1] *TextBlob: Simplified Text Processing — TextBlob 0.18.0.post0 documentation* [online]. 15. únor 2024 [vid. 2024-06-01]. Dostupné z: https://textblob.readthedocs.io/en/dev/

[2] *Penn Treebank P.O.S. Tags* [online]. [vid. 2024-06-01]. Dostupné z: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[3] *pattern en · clips/pattern Wiki* [online]. [vid. 2024-06-01]. Dostupné z: https://github.com/clips/pattern/wiki/pattern-en

[4] *Twitter US Airline Sentiment* [online]. [vid. 2024-06-01]. Dostupné z: https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment/data

[5] *emoji: Emoji for Python*. OS Independent. Python

[6] contractions on Pypi. *Libraries.io* [online]. 15. listopad 2022 [vid. 2024-06-01]. Dostupné z: https://libraries.io/pypi/contractions

[7] PHD, Pavlo Fesenko. Best open-source models for sentiment analysis — Part 1: dictionary models. *Medium* [online]. 5. říjen 2023 [vid. 2024-06-01]. Dostupné z: https://medium.com/@pavlo.fesenko/best-open-source-models-for-sentiment-analysis-part-1-dictionary-models-ece79e617653

[8] *Sentiment Analysis with Textblob and Vader in Python* [online]. [vid. 2024-06-01]. Dostupné z: https://www.analyticsvidhya.com/blog/2021/10/sentiment-analysis-with-textblob-and-vader/

[9] *Python Decision Tree Classification Tutorial: Scikit-Learn DecisionTreeClassifier* [online]. [vid. 2024-06-01]. Dostupné z: https://www.datacamp.com/tutorial/decision-tree-classification-python