

Il Pre-processing per testi in lingua italiana estratti da Social Media

Lorenzo Imbonati 846577,
Susanna Maugeri 839365,
Veronica Morelli 839257

Appello di Statistica Spaziale e Ambientale del 23 aprile 2021

1 Introduzione

Il Text Mining, anche chiamato Knowledge Discovery in Text (KDT), è un campo del Data Mining che comprende un insieme di tecniche con le quali è possibile estrarre informazioni utili da dati sia strutturati che non strutturati e ricercare patterns che non sarebbero rilevabili tramite la semplice lettura dei testi da parte di una persona. È impiegato in vari contesti quali il Natural Language Processing, l'Information Retrieval, la Text Classification e il Text Clustering.

Per *Pre-processing* si intende l'insieme di tecniche volte alla pulizia e alla preparazione dei dati testuali per le successive analisi. È una delle componenti chiave del Text Mining e ha un impatto importante sulla qualità delle analisi successive. [1]

L'importanza della fase di Pre-processing del testo è stata approfondita da diversi studiosi, quali

- Song, Liu e Yang, che hanno analizzato l'utilità di rimuovere le stop-words e di effettuare lo stemming su un dataset di articoli di giornale in lingua inglese; [2]
- Toman, Tesar e Jezek, che si sono occupati degli effetti di lemmatizzazione, stemming e rimozione delle stop-words su un dataset di testi in lingua inglese e ceca; [3]

Il codice Python implementato ed utilizzato è reperibile alla cartella di Google Drive *StatisticaSpaziale.Imbonati.Maugeri.Morelli* al link <https://tinyurl.com/z673pmp6> accedendo tramite account Unimib.

- Méndez, Fdez-Riverola, Díaz, Iglesias e Corchado, che hanno studiato la rimozione delle stop-words, lo stemming e diversi tipi di schemi per la tokenizzazione in un task di rilevazione di spam nelle emails; [4]
- Pomikálek e Rehurek, che si sono occupati di studiare l'influenza del pre-processing (tokenizzazione, rimozione delle stop-words e stemming) su parte del dataset Reuters 21578; [5]
- Duwairi, Al-Refai e Khasawneh hanno studiato gli effetti dello stemming nell'analisi di testi in lingua araba; [6]
- Torunoglu, Cakirman, Ganiz, Akyokus e Gurbuz hanno approfondito l'impatto dello stemming e della rimozione delle stop-words nei testi di lingua turca; [7]
- Infine, anche Toraman, Can e Kocberber hanno studiato gli effetti dello stemming nell'analisi di testi in lingua turca. [8]

2 Focus sui Social

Per quanto riguarda il Text Mining nel contesto dei Social Media o dei Social Network è opportuno effettuare delle considerazioni importanti.

Al giorno d'oggi i Social Media fanno parte della nostra quotidianità e sono uno strumento per poter comunicare in modo molto diretto e rapido con altre persone. Con questi strumenti è più immediata anche la condivisione di idee, interessi e conoscenze. Quasi chiunque può scrivere un post, scrivere un commento o tenere un blog.

Per questo motivo i Social Media rappresentano una preziosa fonte di informazioni; ad esempio le aziende li utilizzano per analizzare o prevedere le esigenze dei clienti e per comprendere la percezione del loro marchio. È quindi diventata essenziale l'analisi di questo tipo di testi e la definizione di tecniche e metodologie più specifiche per il raggiungimento di risultati migliori.

L'aspetto più rilevante è il fatto che le persone tendano a non scrivere utilizzando frasi strutturate, una grammatica accurata e uno spelling corretto, ma che si rifacciano ad abbreviazioni e slang linguistici piuttosto che ad un linguaggio formale. Per questo motivo potrebbe essere difficile determinare corretti e regolari pattern di dati da analizzare.

L'obiettivo del Pre-processing in questo contesto è quello di rendere questo tipo di testi fruibili dai classici strumenti di analisi già utilizzati per tasks di Natural Language Processing.

3 Fasi di Pre-processing

3.1 Trasformazione Lowercase

La fase iniziale del Pre-processing consiste nel ridurre tutto il testo in carattere minuscolo, in inglese *lowercase*. Questa operazione è necessaria per poter trattare parole uguali ma scritte con formati diversi allo stesso modo. È una delle tecniche più semplici ma maggiormente efficaci nel Text Pre-processing.

Ad esempio, le parole *Data*, *data* e *DATA* hanno tutte lo stesso significato, senza la trasformazione in lowercase verrebbero considerate come termini differenti e questo non sarebbe corretto.

Un caso in cui questa operazione diventa fondamentale è la ricerca testuale. Se si ipotizza infatti di dover effettuare una ricerca per parole all'interno di un documento senza aver fatto la trasformazione in caratteri omogenei, si potrebbe non ottenere alcun risultato perché il termine ricercato è semplicemente scritto con caratteri diversi. In questi casi questa tecnica diventa essenziale.

Analogamente, si potrebbe decidere di trasformare tutti i testi in caratteri uppercase, cioè maiuscoli. L'importante è che la scelta sia coerente in tutti i documenti.

3.2 Rimozione di elementi di disturbo: punteggiatura, hashtags, mentions, tags HTML, URLs e numeri

Punteggiatura, hashtags e mentions

Una fase importante, soprattutto se il corpus proviene da un Social Media, è la rimozione di elementi di disturbo come i cancelletti degli hashtags e le menzioni, oltre che la punteggiatura. Un tweet che non è stato sottoposto a Pre-processing è altamente non strutturato e può contenere informazioni non rilevanti per l'analisi; è quindi necessario pulirlo per poterne usufruire in modo efficace.

Per quanto riguarda gli hashtags, è utile rimuovere il # ma non il termine che lo segue in quanto portatore di significato (addirittura potrebbe esserlo più del resto del testo). È possibile effettuare questa procedura con l'utilizzo di Regex.

Per quanto riguarda le mentions, invece, poiché si tratta di richiami al nome utente di un account, dovranno essere eliminate: non forniscono infatti informazioni e, se tenute, potrebbero anche portare l'analisi fuori strada.

Infine è utile effettuare la rimozione dei segni di punteggiatura: si tratta di simboli non utili in quanto l'obiettivo è quello di analizzare il testo.

Tags HTML e URLs

Quando si estraggono dati da siti web tramite scraping, è comune trovarvi anche tags HTML, che non apportano ulteriore significato al testo e che per questo devono essere rimossi.

URL è l'acronimo di Uniform Resource Locator, si tratta di codici che servono per identificare i siti nel *www*. Solitamente, da questi non si ottengono informazioni, si possono quindi rimuovere.

Numeri

È possibile rimuovere le stringhe di numeri dai testi qualora il task che si stia implementando non ne richieda il mantenimento. In alternativa, si possono trasformare in parole tramite la libreria *Num2Words* [9] di Python che supporta anche la lingua italiana.

3.3 Conversione caratteri speciali in caratteri ASCII

In italiano, diversamente che in lingue come l'inglese, non è raro incontrare parole accentate come *è* e *più*. I caratteri finali di questi termini potrebbero rappresentare un problema in quanto non si tratta di caratteri ASCII. Tuttavia è possibile trasformarli attraverso la libreria *Unidecode* [10] di Python.

3.4 Tokenization

La Tokenization è una tecnica che suddivide il testo di un documento in tokens, blocchi atomici ed indivisibili (in genere parole).

Un token è una qualsiasi sequenza di caratteri circondata da dei delimitatori. Vi è la necessità, dunque, di definire questi delimitatori, che possono essere spazi, tabulazioni e caratteri di ritorno ma anche i segni di punteggiatura. Ad esempio, la corretta tokenizzazione della frase *"vado felice all'università"* sarebbe [*"vado"*, *"felice"*, *"all"*, *"università"*].

Altri caratteri che, a seconda del contesto, possono essere o meno dei delimitatori sono: `<>` `()` `,` `.` `-` `"` `&` `/` `#`. Infatti, si possono creare delle ambiguità, per esempio:

- I caratteri `"`, `,` e `."` usati come punteggiatura sono diversi da quelli che separano le cifre di un numero decimale;
- Il carattere `"-"` usato come operazione di differenza tra due numeri è diverso da quello usato come trattino in un testo.

Viste le difficoltà descritte, è importante creare degli strumenti di tokenizzazione ad hoc in base al contesto in cui ci si trova. Infatti, la tokenizzazione risulta facile per un umano ma può diventare difficile per un calcolatore.

3.5 Stop-words

Le Stop-words sono parole che, data la loro elevata frequenza in una lingua, sono ritenute poco significative. Si tratta principalmente di articoli determinativi, pronomi, preposizioni e congiunzioni, cioè di elementi che non apportano ulteriore significato al testo.

Ci sono due metodi per costruire un elenco di parole non significative, che spesso vengono utilizzati assieme: la costruzione manuale e l'estrazione automatica. Il primo consiste nello stilare una lista di parole o nell'utilizzo di liste già disponibili online, mentre il secondo nel calcolare la frequenza dei vari termini e associare ad alte frequenze un elevato valore di noise. Si può affermare che la costruzione automatizzata di liste di stop-words sia il futuro in questo campo poiché presenta numerosi vantaggi tra cui la capacità di lavorare con qualunque lingua, cosa che né l'analista umano né l'elenco di parole preimpostato possono ad ora fare. Mentre l'uso di un vocabolario è di semplice utilizzo ma poco flessibile poiché non specifico per ciascun dominio, l'estrazione automatica è duttile e permette di costruire un elenco di parole non significative appropriato per diversi contesti.

Data la complessità della lingua italiana, non esiste un vocabolario di stop-words universale. È opportuno dunque utilizzare un vocabolario di

base e successivamente ampliarlo con termini specifici del contesto in cui si opera.

Inoltre, poiché i corpora possono contenere un numero molto ingente di testi, è importante utilizzare un algoritmo che sia il più efficiente (e quindi rapido) possibile nell'eliminazione delle stop-words. Gli studiosi Yao e Zewen hanno approfondito l'argomento implementando diverse funzioni e concludendo che la migliore fosse l'Hash Filter, che associa ad ogni stop-word e ad ogni parola nel corpus un valore numerico: se i valori di due parole coincidono, la parola nel corpus è una stop-word e viene eliminata.[11]

Con l'eliminazione di queste parole molto frequenti si ottengono diversi vantaggi, come un aumento della velocità del processo di analisi e la riduzione della dimensionalità del vocabolario, senza perdita di informazione.

3.6 Normalizzazione

La normalizzazione è un procedimento volto all'eliminazione della ridondanza informativa e del rischio di incoerenza in un database. Viene applicata per evitare che parole che portano lo stesso significato ma sono declinate in modo diverso siano considerate diverse in un'analisi di tipo semantico.

3.6.1 Stemming

Lo Stemming è il processo che riconduce una parola in forma flessa alla sua radice o tema (*stem* in inglese) tramite l'eliminazione di prefissi e suffissi. È una procedura molto utilizzata nella fase di pre-processing di un testo per le analisi successive.

Ad esempio, le forme verbali *correre*, *corro*, *corriamo* e *correremo* vengono tutte ricondotte al tema *corr-*. Altri esempi riguardano le parole al singolare, plurale, maschile e femminile, come *bambino*, *bambini*, *bambina* e *bambine*, che vengono tutte ricondotte alla radice *bambin-*.

In generale, la radice non deve necessariamente essere una parola di senso compiuto, come nei due esempi appena mostrati.

Problema Risalire alla radice di una parola a partire dalla sua forma flessa non è sempre semplice e, per quanto sia possibile definire regole ben precise, il linguaggio naturale presenta sempre diverse eccezioni, soprattutto per quanto riguarda la lingua italiana. Inoltre, ad ogni forma flessa possono corrispondere più radici e questo significa che potrebbero esserci diverse

radici corrette per la stessa forma flessa. La scelta della radice “corretta” dipende dal significato della parola e dal contesto nel quale è inserita.

Ad esempio, per le parole *esplodo*, *esplodono* ed *esploderanno*, probabilmente la radice individuata sarebbe *esplod-*, mentre per le parole *esplosione*, *esplosiva*, *esploso*, la radice potrebbe essere *esplos-*.

Esistono numerosi algoritmi che svolgono lo stemming, uno dei più utilizzati, l'algoritmo **Snowball Stemmer** [12], restituisce come radice *explosion-* per *esplosione* ed *esplos-* per *esploso* ed *esplosiva*.

Davanti a questo risultato sorgono sicuramente delle domande:

- È corretto assegnare ad *esplosione* un tema diverso rispetto a quello di *esplosivo*?
- È corretto assegnare temi diversi a *esplodo* e *esploso*?

La risposta potrebbe essere affermativa o negativa a seconda del contesto.

Questo semplice esempio dimostra che lo stemming non si riduce ad un'unica regola deterministica ed universalmente accettata. Se si decide di utilizzare la procedura di stemming in analisi di Natural Language Processing è necessario analizzare con cura i testi a cui viene applicato e i risultati che ne scaturiscono.

L'ambiguità dello stemming varia a seconda della lingua in cui sono scritti i documenti del corpus. Lingue morfologicamente ricche come l'italiano hanno un numero maggiore di forme flesse rispetto all'inglese.

Ad esempio, gli aggettivi in inglese non si declinano in singolare, plurale, maschile o femminile, naturalmente con alcune eccezioni. Per un aggettivo che in inglese non viene declinato, in italiano si hanno 4 forme flesse. Le medesime considerazioni valgono anche per i verbi, per i quali le forme flesse variano in base al numero totale di combinazioni di tempi, modi e persone; in italiano ce ne sono molte di più che in inglese.

Per questo motivo, non esiste uno stemmer universale, cioè che vada bene per tutte le lingue. Il pacchetto NLTK (Natural Language Tool Kit) in Python offre 3 diversi operatori di stemming: lo **Snowball Stemmer**, disponibile per molte lingue tra cui l'italiano, il **Porter Stemmer** e il **Lancaster Stemmer**, disponibili solo per la lingua inglese.

3.6.2 Lemmatization

Un'alternativa allo Stemming è la Lemmatization: a differenza del primo, la lemmatizzazione riporta ogni parola al suo lemma, parola di senso compiuto che si troverebbe nel vocabolario, e non alla sua radice.

Ad esempio, le parole *correre*, *corro*, *corriamo* e *correremo* verrebbero ricondotte dallo stemming alla radice *corr-*, mentre dalla lemmatization al lemma *correre*.

Il risultato dello stemming non è errato, ma mentre per l'inglese, che presenta poche forme flesse e di semplice morfologia, i risultati sono già soddisfacenti, per l'italiano e per altre lingue più complesse potrebbe funzionare meglio la lemmatizzazione.

Una parola potrebbe ancora essere ricondotta a più stem o a più lemmi, il vantaggio della lemmatization è che il lemma è una parola canonica e di senso compiuto.

In Python esistono numerosi pacchetti per implementare la lemmatization, una classe molto utilizzata è **WordNetLemmatizer**, che fa parte del pacchetto NLTK. WordNet [13] è un esteso database lessicale della lingua inglese realizzato dalla Princeton University nel 2010. È gratuito e disponibile online per l'utilizzo ed il download. Successivamente sono stati costituiti nuovi database di lessico per altre lingue, come ItalWordNet e MultiWordNet, che contengono lessico per la lingua italiana.

Un'altra libreria open-source popolare per il Natural Language Processing è Spacy [14], rilasciata nel 2015. In Spacy sono implementati numerosi strumenti per svolgere funzioni di Pre-processing per più di 64 lingue. Spacy può quindi essere utilizzata per eseguire la lemmatizzazione su corpus di documenti in lingua italiana.

Problema La lemmatizzazione svolta con Spacy in italiano non è molto efficiente e precisa. È possibile che sostantivi che semanticamente sono oggetti, ma letteralmente forme verbali in participio passato, vengano riconosciuti come verbi e successivamente trasformati al lemma di verbo infinito.

Ad esempio, la parola *fatto* in un corpus potrebbe definire un evento, ma il lemmatizer di Spacy lo riconosce come participio passato e lo trasforma in *fare*. Questo è un tipo di errore che comporta la compromissione del senso della frase.

3.7 Spell Check

Il controllo ortografico, in inglese “Spell Check” è una fase importante del pre-processing di un testo. L’obiettivo è quello di individuare e successivamente correggere errori di scrittura. Esistono numerose librerie in Python per l’auto-correzione dell’ortografia, le più popolari sono:

- **Pyspellchecker**

Questa libreria [15] permette di correggere errori di spelling con il metodo *correction()*. Allo stesso tempo vengono proposti dei candidati per queste parole con il metodo *candidates()*, così che l’utente possa scegliere il termine da sostituire a quello errata.

Ad esempio,

```
spell.correction('becuse') = 'because'  
spell.candidates("conceed") - 'concede', 'conceded'
```

Pyspellchecker è disponibile per l’inglese e per altre lingue, ma non per quella italiana.

- **Textblob**

Questa libreria [16] di Python ha numerose funzionalità tra cui il controllo e la correzione dell’ortografia. Con il metodo *correct()* di TextBlob si può ottenere una correzione per qualsiasi testo o parola.

È disponibile anche la classe *Word*, insieme al metodo *spellcheck()*, dove per ogni parola per cui si vuole ottenere una correzione si ha come risultato una lista di tuple con correzione e punteggio di confidenza (confidence score) che valuta l’affidabilità della correzione proposta.

Ad esempio,

```
word = Word()  
word.spellcheck('percieve') [('perceive', 1.0)]
```

In questo caso viene proposta una sola correzione che di conseguenza ha affidabilità del 100%. In generale TextBlob ha un’accuratezza di circa il 70%.

TextBlob come Pyspellchecker non è disponibile per la lingua italiana.

- **Autocorrect**

Autocorrect [17] è un pacchetto per controllo e correzione dell’ortografia molto utilizzato. La libreria supporta lingue come l’inglese, il polacco, il greco, il portoghese e altre, ma non l’italiano.

È possibile importare la classe *Speller()* per le lingue disponibili e procedere alla correzione.

Ad esempio,

```
spell = Speller() per la lingua inglese
```

`spell("I'm not sleapy and tehre is no place I'm giong to.") = "I'm not
sleepy and there is no place I'm going to."`

Per la lingua italiana è possibile svolgere la correzione dello spelling in maniera differente, sfruttando le funzionalità della libreria di Python **RapidFuzz** [18]. RapidFuzz è una libreria di corrispondenza rapida di stringhe per Python e C++, utilizza i calcoli di similarità tra stringhe del pacchetto **FuzzyWuzzy** [19]. FuzzyWuzzy utilizza la distanza di Levenshtein per calcolare differenze tra sequenze di parole. Attraverso il metodo `fuzz.ratio()` viene valutata la similarità tra due termini.

Ad esempio,
`fuzz.ratio('Mi piace leggere lbri', 'Mi piace leggere libri') = 97.67`.
La similarità tra le due frasi è molto alta, non è pari a 100 perché nella prima frase 'libri' è scritto erroneamente come 'lbri'.

Per costruire uno spell check è possibile scaricare un vocabolario di parole italiane e costruire un ciclo che sostituisca la parola con errori di ortografia con la parola scritta correttamente nel vocabolario, se il valore del *ratio* tra i due termini è maggiore di una soglia prestabilita. In questo modo è possibile dunque correggere gli errori di ortografia.

3.8 Emoticon ed Emoji

Le Emoticon sono riproduzioni stilizzate delle principali espressioni facciali umane, gli Emoji sono piccole immagini. Entrambi hanno un ruolo molto importante online, gli utenti li usano per esprimere le proprie emozioni o rappresentare oggetti. I testi, specialmente se scaricati da un Social Media, potrebbero contenere delle Emoticon o degli Emoji.

In base al task che si vuole implementare, potrebbe essere utile rimuovere Emoticon ed Emoji oppure trasformarli nella loro descrizione. Per esempio potrebbe essere utile mantenerli per la Sentiment Analysis, in quanto sono portatori di significato e di opinione a tutti gli effetti.

Vi sono diversi metodi che si possono utilizzare per rimuoverli dai testi oppure per sostituirli con una stringa di testo che indichi ciò che rappresentano, qualora lo si desideri. Alcuni di questi metodi si basano sul fatto che ciascun Emoji sia identificato da un codice UTF-8 e che le Emoticon siano sequenze riconoscibili di caratteri.

3.8.1 Rimozione degli Emoji

Entrambi i metodi di rimozione degli Emoji presentati si basano sul fatto che questi abbiano una codifica in codice UTF-8.

- *Rimozione tramite la libreria `Emoji` di Python*: l'utilizzo della libreria `Emoji` di Python per la rimozione degli Emoji da stringhe di testo consiste nella definizione di una funzione che decodifica ogni carattere della stringa di input in codice UTF-8 e restituisce come risultato la stringa stessa a cui siano stati rimossi i codici corrispondenti agli Emoji contenuti nella lista `unicode_emoji` della libreria.













 :laughing:	 :blush:	 :smiley:
 :relaxed:	 :smirk:	 :heart_eyes:
 :kissing_heart:	 :kissing_closed_eyes:	 :flushed:
 :relieved:	 :satisfied:	 :grin:

Figura 1: Esempio di classificazione di Emoji nella libreria `Emoji` di Python

- *Rimozione tramite i codici UTF-8*: questo metodo utilizza la libreria `Re` delle Regular Expressions e consiste nella definizione di una funzione che identifica come Regex i codici UTF-8 corrispondenti agli Emoji e le sostituisce con una stringa vuota.

3.8.2 Rimozione delle Emoticon

Le Emoticon sono ritratti di espressioni umane fatti con punteggiatura, simboli e lettere. È possibile che vengano eliminate già con la funzione di eliminazione della punteggiatura. Nel caso in cui, invece, quello di rimozione delle Emoticon fosse un task specifico, è possibile importarne una lista, come quella che si può trovare in [20] e definire una funzione di sostituzione di ogni sottostringa che contiene quelle sequenze di caratteri in stringa vuota.

3.8.3 Trasformazione di Emoji ed Emoticon in testo

Come accennato in precedenza, per alcuni tasks di Text Mining potrebbe essere utile mantenere gli Emoji e le Emoticon, anche se in una forma diversa da quella di visualizzazione online, poiché questi contribuiscono con il loro significato all'analisi in questione. Un esempio è il task di Sentiment Analysis

su delle recensioni: faccine felici potrebbero indicare soddisfazione, faccine tristi o arrabbiate, al contrario, insoddisfazione per il prodotto o servizio.

- *Sostituzione di Emoji tramite VADER*: acronimo di Valence Aware Dictionary and sEntiment Reasoner, VADER è uno strumento per la Sentiment Analysis basato su lessico e su regole. Nasce per il contesto specifico dei Social Media e tra le sue risorse ha anche un dizionario specifico di Emoji nel quale, per ognuna, è riportata la relativa descrizione. [21]











	winking face
	smiling face with smiling eyes
	face savoring food
	smiling face with sunglasses
	smiling face with heart-eyes
	face blowing a kiss
	smiling face with 3 hearts
	kissing face
	kissing face with smiling eyes
	kissing face with closed eyes

Figura 2: Esempio di classificazione di Emoji in VADER

Potrebbe essere utile per sostituire a ciascun Emoji una stringa di testo che ne indica il significato, qualora se ne volesse tener conto.

- *Sostituzione di Emoji ed Emoticon tramite codici UTF-8*: la sostituzione può essere effettuata anche con l'utilizzo di un dizionario che associa ad ogni codice UTF-8 degli Emoji o delle Emoticon, la relativa descrizione dell'emozione o dell'oggetto. Un esempio di dizionario utile per questo task si può trovare in [20].

3.9 N-Grammi

Un n-gramma è una sotto-sequenza di n elementi di una data sequenza. Nel Text Mining per n-grams, con $n < 4/5$, si intendono gruppi di n parole che prese insieme hanno un significato diverso rispetto alla somma del significato di ogni parola. Ad esempio, *Corriere della Sera* e *New York Times* sono rispettivamente un bigramma ed un trigramma.

È utile considerare, nei tasks di NLP, non tutti gli n-grammi possibili, cioè ogni combinazione di parole in sequenza, ma quelli probabilistici. Vengono dunque rese un unico token solo le parole che appaiono insieme

più frequentemente di quanto appaiano separate e che probabilmente quindi non avrebbero lo stesso senso non unite.

Gli autori di [22] propongono un metodo data-driven che sostituisce coppie di parole col relativo bigramma se la probabilità di trovarle una di seguito all'altra è maggiore di una certa soglia arbitraria.

La necessità di individuare queste coppie o gruppi di parole nasce dal fatto che, nell'analisi del testo, una frase non ha un significato che è semplicemente la composizione dei significati delle singole parole da cui è formata. Questa procedura permette inoltre di ridurre la dimensionalità del corpus e aiuta, nelle fasi successive al pre-processing, la comprensione.

4 Possibili miglioramenti per l'italiano

Durante lo studio delle tecniche di Pre-processing presentate, ci siamo resi conto che, mentre per l'inglese questi strumenti funzionano quasi sempre molto bene, per l'italiano ci sono degli aspetti che necessitano di essere migliorati. Infatti non abbiamo trovato una lista di stop-words completa e soddisfacente, lo stemming e la lemmatizzazione non performano bene e non vi è una libreria per la correzione dello spelling per la nostra lingua.

Riferimenti bibliografici

- [1] Alper Kursat Uysal and Serkan Gunal. The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104–112, 2014.
- [2] Fengxi Song, Shuhai Liu, and Jingyu Yang. A comparative study on text representation schemes in text categorization. *Pattern analysis and applications*, 8(1):199–209, 2005.
- [3] Michal Toman, Roman Tesar, and Karel Jezek. Influence of word normalization on text classification. *Proceedings of InSciT*, 4:354–358, 2006.
- [4] Jose Ramon Mendez, Florentino Fdez-Riverola, Fernando Diaz, Eva Lorenzo Iglesias, and Juan Manuel Corchado. A comparative performance study of feature selection methods for the anti-spam filtering domain. In *Industrial Conference on Data Mining*, pages 106–120. Springer, 2006.

- [5] Jan Pomikálek and Radium Rehurek. The influence of preprocessing parameters on text categorization. *International Journal of Applied Science, Engineering and Technology*, 1:430–434, 2007.
- [6] Rehab Duwairi, Mohammad Nayef Al-Refai, and Natheer Khasawneh. Feature reduction techniques for arabic text categorization. *Journal of the American society for information science and technology*, 60(11):2347–2352, 2009.
- [7] Dilara Torunoğlu, Erhan Çakirman, Murat Can Ganiz, Selim Akyokuş, and M Zahid Gürbüz. Analysis of preprocessing methods on classification of turkish texts. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pages 112–117. IEEE, 2011.
- [8] Cagri Toraman, Fazli Can, and Seyit Koçberber. Developing a text categorization template for turkish news portals. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pages 379–383. IEEE, 2011.
- [9] Pypi.org - Num2words.
- [10] Python.org - Unicode.
- [11] Zhou Yao and Cao Ze-wen. Research on the construction and filter method of stop-word list in text preprocessing. In *2011 Fourth International Conference on Intelligent Computation Technology and Automation*, volume 1, pages 217–221. IEEE, 2011.
- [12] Snowball.tartarus.org - Snowball Stemmer.
- [13] Wordnet.princeton.edu - Wordnet.
- [14] Spacy.io - Spacy.
- [15] Pypi.org - Pyspellchecker.
- [16] Textblob.readthedocs.io - Textblob.
- [17] Github.com - Autocorrect.
- [18] Github.com - RapidFuzz.
- [19] Github.com - FuzzyWuzzy.
- [20] GitHub.com - Emot.
- [21] GitHub.com - VaderSentiment.
- [22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *arXiv preprint arXiv:1310.4546*, 2013.