

Analisi della lingua su un dataset  
di pagine web tramite MapReduce  
- Progetto Finale Big Data -

Crosara Marco VR434403

Marzo 2019 / Aprile 2019

UNIVERSITÀ DEGLI STUDI DI VERONA  
Anno Accademico 2018/2019

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
1.1	Idea e obiettivi del progetto . . . . .	3
1.2	Il dataset e i suoi file . . . . .	4
<b>2</b>	<b>Struttura generale dell’analizzatore</b>	<b>7</b>
2.1	Input Format e Distributed Cache . . . . .	7
2.2	Map e Reduce . . . . .	8
2.3	Formato dell’output del tool . . . . .	9
<b>3</b>	<b>Funzionamento specifico moduli</b>	<b>11</b>
3.1	Avvio del job MapReduce . . . . .	11
3.2	La classe <i>Driver</i> . . . . .	11
3.3	Lo standard ISO, dictionary.json e le relative classi . . . . .	11
3.4	La classe <i>Map</i> . . . . .	12
3.5	Page split tramite RegExp . . . . .	13
3.6	<i>Language</i> , <i>Dictionary</i> e <i>JsonParser</i> . . . . .	13
3.7	Le statistiche . . . . .	14
3.8	Scelta della lingue rappresentative di una pagina . . . . .	14
3.9	Ottimizzazione del tool per le lingue orientali . . . . .	16
3.10	<i>Reduce</i> e <i>ResultChecker</i> . . . . .	16
<b>4</b>	<b>Test su cluster e Risultati finali</b>	<b>18</b>
4.1	Accuratezza dei risultati e tempo di esecuzione . . . . .	20
<b>5</b>	<b>Considerazioni conclusive</b>	<b>22</b>

# 1 Introduzione

Il progetto è partito dalla curiosità di analizzare un grosso quantitativo di pagine web, con lo scopo di farne in qualche modo categorizzazione. Come punto di partenza si è scelto di cercare un dataset opportuno a questo tipo di analisi. Dopo una ricerca approfondita è stato scelto Common Crawl[3]: una enorme collezione di pagine dati in formato *raw*, da cui sono stati estratti metadati e testo semplice. Tutti i dati in formato non elaborato sono raccolti mediante l'uso di un Crawler che esegue la scansione del web.

**Definizione.** *Il Crawler, comunemente chiamato anche Spider o Bot, è un software/script che ha lo scopo di scansione dei dati. Tale termine viene tipicamente associato alla scansione di pagine web oppure di database, con il fine di estrapolarne i contenuti. In particolare nella SEO il Crawler viene associato allo spider di Google. In generale il crawling è una delle fasi per l'indicizzazione dei siti nella SERP(Search Engine Results Page), la pagina dei risultati di un motore di ricerca.*

## 1.1 Idea e obiettivi del progetto

Il progetto che si è pensato di realizzare con il dataset Common Crawl è un riconoscitore della lingua di una pagina web. Nello specifico, a partire dall'enorme mole di pagine a disposizione, è stato progettato un programma MapReduce che sia in grado di ritornare per ogni pagina identificata da un url univoco, la o le lingue in cui tale testo è stato scritto. L'idea per la realizzazione è che a partire dal testo della pagina vengano estratte le singole parole tramite una semplice ma specifica operazione di *split*, dettagliata meglio in seguito. Successivamente, tramite un confronto di queste ultime con un campione di circa 300 parole per ogni lingua, si rende possibile stabilire in modo approssimato ma statistico le lingue di appartenenza della pagina.

Data la consapevolezza dell'assai complicata operazione di riconoscimento di una lingua di un testo e della poca precisione di un sistema unicamente basato su divisione di testo e confronti limitati, poniamo l'obiettivo di questo progetto. Il nostro goal è la realizzazione di un programma MapReduce ben strutturato e modulare che sia in grado di analizzare e riconoscere la lingua dato un qualsiasi opportuno algoritmo. L'obiettivo del progetto non è dunque quello di realizzare un sistema che riesca a riconoscere in maniera precisa il maggior quantitativo possibile di coppie pagina/lingua ma quello di progettare un MapReduce con un'architettura adatta a svolgere questo tipo di analisi in maniera semplice.

Oltre all'obiettivo di ritornare le lingue delle pagine web ce ne poniamo un altro: quello di verificare se queste lingue da noi riconosciute sono corrette. Nello specifico, supponiamo di voler dare la possibilità di testare lo script che individua le lingue. Il controllo verrà effettuato a partire da coppie  $\langle url, lingua\_corretta \rangle$  che Common Crawl fornisce tramite dei file che contengono i risultati di un analizzatore esterno(non MapReduce) che verrà introdotto in seguito. Dato un url relativo a una pagina, il nostro programma MapReduce dovrà dunque essere

in grado di confrontare se la lingua o le lingue da noi riconosciute sono le stesse che vengono proposte dall'analizzatore esistente, o in un caso più generico se sono le stesse riportate nei file *info*.

## 1.2 Il dataset e i suoi file

Di seguito una tabella presente anche sulla pagina web di Common Crawl, utile per capire i diversi formati a disposizione degli utenti per la consultazione del dataset.

Tipo File	Descrizione	#Files	Dim.dei file compressi
Segments		100	
WARC files	Dati grezzi generati dal Crawler	64000	59.86 TiB
WAT files	Contengono meta-dati relativi ai record nei file WARC	64000	18.23 TiB
WET files	Contengono solo testo semplice estratto dai WARC	64000	7.62 TiB
Robots.txt files	Files Robot.txt richiesti ai vari siti analizzati (consentono al Crawler di scansionare meglio i siti web)	64000	170 GiB
Non-200 responses files	Risposte del server con HTTP status code diverso da 200 (404, redirects, ecc.)	64000	1.79 TiB
URL index files	Indice degli url analizzati, completo di utili informazioni aggiuntive	302	210 GiB

Tra i vari formati proposti, quelli che ci interessano, poiché utilizzati nel progetto, sono ‘*WET*’ e ‘*Url Index*’. I primi contengono il testo estratto dalle pagine web, nella pratica a partire dalla sorgente html presente sul *WARC*; sono stati rimossi tutti i tag ed è stato mantenuto solo il testo semplice. Si ottiene quindi una pagina dal contenuto testuale non strutturato che ci ritorna utile per l'estrazione delle parole. Ogni pagina viene preceduta da alcune righe di header che riportano alcune informazioni, tra cui: la data di acquisizione, la lunghezza del testo in caratteri e ovviamente l'url a cui noi attribuiremo una lingua e che utilizzeremo come identificatore della pagina web stessa. Qui sotto un piccolissimo estratto di un file *WET* relativo a una pagina in inglese. Risulta semplice distinguere l'header e il relativo *plain text*.

```
WARC/1.0
WARC-Type: conversion
WARC-Target-URI: http://37...bly.com/.../the-time-we-h
WARC-Date: 2019-02-15T18:51:42Z
WARC-Record-ID: <urn:uuid:b9c6bb42-ae7a-4375-838c-e4df94644273>
WARC-Refers-To: <urn:uuid:686735d9-f7de-443d-aaac-99c1af01248d>
WARC-Block-Digest: sha1:LFNRVL6AG7X2X25BRRYTPJ5FA30IS56T
```

Content-Type: text/plain  
Content-Length: 1552

The time we have - My Learning journey 7  
Home ... All About Me ... How I learn ...  
The time we have 11/27/2014 ... 0 Comments  
This video is about how much time in our life we REALLY have. It shows that we need to use time wisely. I think we watched this video so we learn to find our passion and to do our passion. ... I am at 4380 / 28,835 days out of my life. When I watched this video my first emotions were: shock, understanding, and sadness. I love this video. ...  
Hi i'm Jordan and I like playing Clarinet and playing Minecraft. My favourite sport is hockey. And I love karate!  
RSS Feed ... Powered by Create your own unique website.

Nei nostri test utilizzeremo solo pochi file *WET* che da soli contengono migliaia di pagine web in formato *plain text*. Come detto prima l'altro tipo di file del dataset che andremo a utilizzare sono gli *Url Index*. Questi ultimi contengono infatti tra le altre innumerevoli informazioni, la lingua che *CLD2* (*Compact Language Detector 2*) ha individuato per la pagina relativa all'url in questione. *CLD2* è un analizzatore esterno che identifica probabilisticamente più di 80 lingue diverse in *Unicode UTF-8*.

Esempio di una linea appartenente a un file *Url Index* qualsiasi:

```
1,102,30,163)/modules/mylinks/... 20190218011538 {"url":"http://163.30.102.1/modules/mylinks/...", "mime": "text/html", "mime-detected": "text/html", "status": "200", "digest": "3LTEADYEETRMLYCNYW652S3ED700FLHE", "length": "7990", "offset": "605284", "filename": ".../segments/...-00001.warc.gz", "charset": "UTF-8", "languages": "zho,eng"}
```

Ora è necessario discutere di un problema affrontato nella fase di progettazione: l'elenco di lingue che *CLD2* propone per un determinato url e dunque la relativa linea nel file *Url Index*, non ha una corrispondenza specifica con un file *WET*. Questo significa che dato un url con relativa pagina su un file *WET*, le relative informazioni estratte da *CLD2* potrebbero trovarsi in uno qualsiasi delle centinaia di file *Url Index*. Detto questo le soluzioni possibili sono due. La prima è rendere disponibili a MapReduce tutti i file *Url Index* per l'analisi delle lingue relative anche a un solo file *WET* (infattibile per lo spazio necessario sul cluster a disposizione). La seconda opzione (quella scelta) è rimappare le linee dei file *Url Index* su dei nuovi file *info* in modo tale che nel primo file siano presenti tutti e soli gli url con le relative informazioni del primo file *WET*, sul secondo siano presenti gli url del secondo *WET* e così via. Anche se onerosa, l'operazione di riscrittura dei file è stata molto facilitata dall'informazione presente su ogni linea del file *Url Index* che indica in quale file *WARC* e dunque *WET* è presente la relativa pagina. Si può vedere questa ultima informazione

tra quelle evidenziate nell'estratto del file *Url Index* prima fornito. Elaborando tale linea e spostandola nel nuovo file *info-00001.info* si ottiene il risultato seguente. Il carattere separatore è un *tab*.

```
http://163.30.102.1/modules/mylinks/...    zho,eng    UTF-8
```

Prima di iniziare la descrizione dell'analizzatore MapReduce è utile menzionare che la versione utilizzata del dataset Common Crawl è quella relativa a *Febbraio 2019*, l'ultima versione disponibile durante la fase di realizzazione del tool. Infine è opportuno sottolineare che quando nelle sezioni seguenti si farà riferimento a una 'pagina', si vorrà intendere a una parte del file *WET* che rappresenta una pagina web analizzata dallo spider ed è dunque un *plain text*. Ogni pagina è preceduta dal relativo header(con url).

## 2 Struttura generale dell'analizzatore

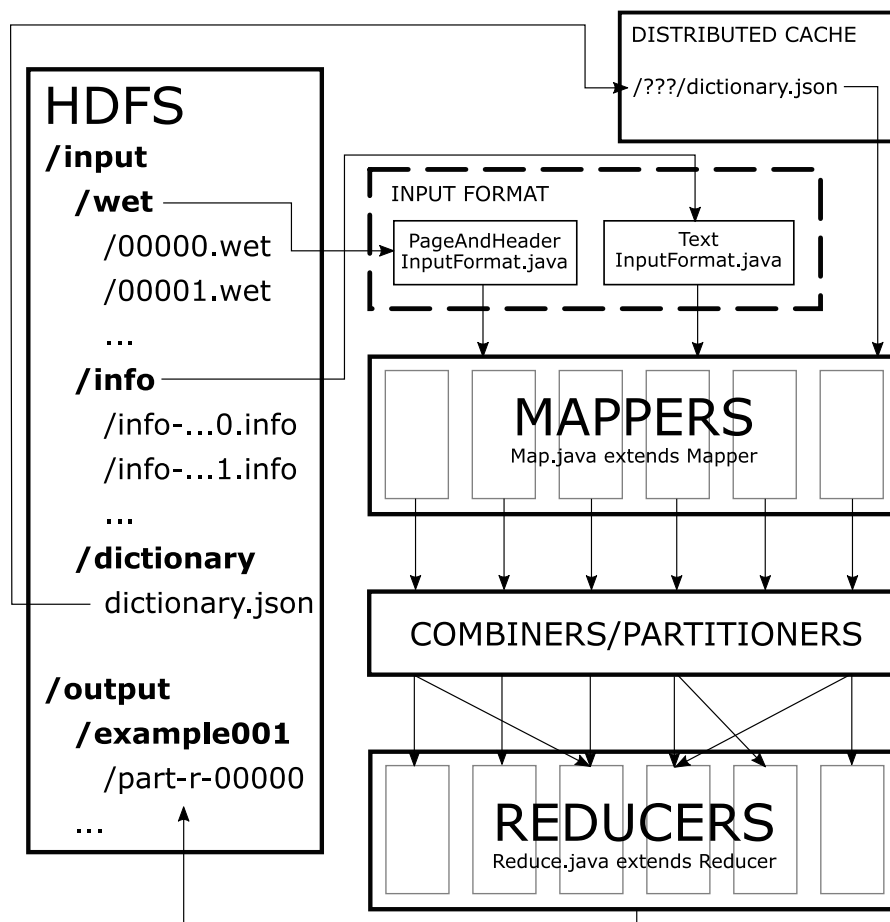


Figura 1: Schema astratto MapReduce realizzato

### 2.1 Input Format e Distributed Cache

Qui sopra uno schema astratto [Figura 1] che mostra il funzionamento generale dell'analizzatore MapReduce realizzato. Per poter cominciare l'analisi, i mapper hanno bisogno di 3 tipologie di file:

- Uno o più file *WET* che contengono come già visto in precedenza le pagine in formato *plain text* con il relativo header (e dunque l'url univoco).
- Uno o più file *info* che mappano gli url con la 'soluzione' sulla lingua proposta da *CLD2* e il charset della pagina (che nel nostro analizzatore non useremo).

- *dictionary.json* : è il dizionario in formato json che contiene una lista delle 300 parole più frequenti per ogni lingua.

Tutte le tipologie di file devono essere rese disponibili nell'*HDFS*. Avviando il tool MapReduce, è necessario specificare i vari file in input al programma *jar*, come si piega meglio in seguito. Ogni tipologia di file viene tuttavia trattata in modo diverso: infatti le prime due passano attraverso il normale processo di *split* degli input sui vari mapper, mentre il dizionario, essendo necessario ovunque nella sua interezza, viene caricato sulla *Distributed Cache*. In particolare:

- I file *WET* sono associati a un *Input Format* realizzato ad hoc per questo programma: *PageAndHeaderInputFormat* invece di inviare al mapper una singola linea del file invia l'intera pagina in *plain text* completa del suo header. Il mapper infatti ha bisogno dell'intera pagina con l'indicazione dell'url per poter fare l'analisi della lingua.
- I file *info* sono invece associati a *TextInputFormat.java*. Tale classe è predefinita per la fase di split e invia ai mapper una singola riga che come già detto più volte corrisponde alla tupla  $\langle url, lingue_{CLD2}, charset \rangle$ .
- Il file *dictionary.json* essendo necessario nella sua interezza a tutti i mapper non è associato a un *Input Format*, viene invece caricato dal *Driver* sulla *Distributed Cache* nelle fasi iniziali di MapReduce.

## 2.2 Map e Reduce

Ogni mapper quando viene creato esegue una fase di *setup()* in cui carica il dizionario, prelevandolo dalla *Distributed Cache* tramite un processo di *parsing* del json. Il processo di parsing viene svolto da *JsonParser*. Nella funzione principale il mapper riceve le *input lines* relative ai file *WET* e ai file *info*. Vengono a seguito distinti i due casi:

- Nel caso fosse un input relativo a un file *info*, il mapper esegue un semplice *emit* usando come chiave l'url e come valore l'elenco delle lingue identificate da *CLD2*.
- Nel caso in cui invece le informazioni passate dai *TextFormat* fossero relative a un file *WET*, significa che tale input è un insieme di linee che rappresentano una pagina con preposto il relativo header. Il mapper eseguirà dunque una separazione tra header e pagina, ed emetterà una coppia  $\langle key:Text, value:Text \rangle$ . La chiave sarà l'url estratto dall'header e il valore sarà invece il risultato della funzione *getLanguagesWithStats()* di *LanguageElect*, a cui viene dato in pasto l'insieme di parole presenti sul testo. Tale insieme è ottenuto con una RegExp applicata sul testo della pagina.

Ai reducer arrivano dunque due tipi di tuple:  $\langle url, lingue_{con\_Statistiche} \rangle$  e  $\langle url, lingue_{CLD2} \rangle$ . Ovviamente le due tuple relative alla stessa chiave(cioè con l'url uguale) sono a disposizione dello stesso reducer. Le due tipologie di coppie



vengono distinte l'una dall'altra grazie a dei tag specifici che vengono inseriti dal mapper all'inizio del valore nella tupla e sono poi rimossi dal reducer. Il reducer ora confronterà i risultati della lingua proposta dal tool e quelli di *CLD2*, mettendo assieme i risultati del confronto e aggiungendo il relativo simbolo che indica la precisione dei risultati. Il reducer emetterà infine l'url con tutte le informazioni relative alla lingua, e tali linee verranno salvate in *hdfs* sul relativo file di output. A seguito la descrizione dell'output finale dell'analisi MapReduce.

## 2.3 Formato dell'output del tool


L'output del tool sarà un insieme di linee emesse dai Reducer nel formato seguente:

```
URL Response:lang1,lang2... | Expected:lang1,lang2,... SYMBOL_OF
PREC.{✓,✗,+,−,÷} STATS[lang1=N.N%,... ,Not_Found:N%] [WS] _TAG
```

I singoli campi sono descritti di seguito:

- URL: indirizzo url della pagina analizzata.
- Response: elenco delle lingue che il tool identifica come rappresentative della pagina analizzata.
- Expected: elenco delle lingue identificate dall'analizzatore *CLD2* per l'url in questione.
- SYMBOL\_OF\_PRECISION: simbolo che indica la precisione delle lingue identificate come rappresentative rispetto ai risultati di *CLD2*, nelle sezioni successive elencheremo i diversi simboli e il loro significato.
- STATS: statistiche contenenti la percentuale di tutte le lingue identificate nel testo. Sono un insieme più ampio rispetto alle lingue elencate in 'Response' poichè potrebbero esserci anche lingue con una percentuale bassa che corrispondono a un errore del rilevamento della lingua e vengono dunque scartate tra i risultati ma sono presenti tra le statistiche. Vengono elencate al massimo cinque lingue, le rimanenti, se esistenti, sono quelle con la presenza più bassa sul testo e vengono raggruppate in 'Other\_Langs' con la relativa somma delle percentuali. É presente infine la percentuale di parole che non sono state rilevate in nessuna lingua. Le percentuali sono calcolate come numero di parole di una lingua rispetto al numero di parole totali, come mostrato meglio in seguito.
- WS: tag opzionale che sta per 'Words Split', se presente segnala che le parole identificate nel testo sono state divise ulteriormente per migliorare il rilevamento delle lingue orientali come cinese e giapponese.

Se nei file *info* non vi è una corrispondenza per l'url corrente (ristretto numero di casi) e dunque *CLD2* non propone delle lingue per la pagina, si assume ottimisticamente che le lingue ritornate dal tool MapReduce siano quelle corrette. A seguito un esempio di risultato di analisi su una singola pagina web.

`http://42stocks.com/cgi-bin/symbol.py?symbol=mtnb` Response:eng |  
Expected:eng  eng:6.91%;slv:1.22%;slk:1.22%;swa:1.22%;ron:1.22%  
;Other\_Langs:4.88%;Not\_Found:83.33%

## 3 Funzionamento specifico moduli

A seguito verranno spiegate le classi principali del tool e dunque in generale le scelte progettuali adottate per ottimizzare l'analisi della lingua.

### 3.1 Avvio del job MapReduce

Per sottoporre il job MapReduce ad Hadoop, lasciando le impostazioni su file e directory di default, il comando che dovrà essere dato da terminale sarà il seguente:

```
[cloudera@quickstart ~]$ hadoop jar DATA/analyzer.jar
```

Di default la cartella di output verrà creata nel formato *'yyyyMMdd\_HHmmss'* in base al *timestamp* corrente. In alternativa si possono ovviamente specificare a seguito del file jar alcuni o tutti i seguenti parametri, in ordine: numero di Reducer, file *WET* o directory contenente i file *WET*, file/directory relativi al formato *info*, file relativo al dizionario json e infine directory di output del processo MapReduce. Un esempio:

```
[cloudera@... ~]$ hadoop jar DATA/analyzer.jar 3 /input/wet/00000  
_sample.wet /input/info/info-00000.info /input/dictionary/diction  
ary.json /output/output123
```

Il numero predefinito di reducer, se non verrà altrimenti specificato è 1.

### 3.2 La classe *Driver*

La classe *Driver* si occupa di configurare e inoltrare il job che deve essere eseguito dal cluster Hadoop, iniziando così l'elaborazione. Tra tutte le operazioni che svolge questa classe vi è l'assegnamento dei file di input al corretto *InputFormat* e la configurazione della cartella di output e dei reducer in base agli argomenti dati in input (come spiegato nella sezione precedente). Un'altra operazione svolta consiste nell'impostare la classe mapper e la classe reducer, che nel nostro caso sono rispettivamente *Map* e *Reduce*. Infine la classe *Driver* si occupa anche di assegnare il file *dictionary.json* alla *Distributed Cache*, in modo tale da renderlo successivamente disponibile a tutti i mapper per la consultazione.

### 3.3 Lo standard ISO, dictionary.json e le relative classi

In tutto il progetto e dunque anche in *CLD2* e negli output del processo MapReduce, lo standard usato per rappresentare la lingua è *ISO 639-2*.

**Definizione.** *ISO 639-2 è la seconda parte dello standard internazionale ISO 639, è un elenco di codici a tre caratteri alfabetici che hanno lo scopo di identificare univocamente i nomi delle lingue. È l'evoluzione dell'ISO 639-1 che con due sole lettere identifica molte meno lingue.*

Per questioni implementative, legate alla maggiore facilità di rappresentare le lingue all'interno di un documento *json*, si è scelto lo standard *ISO 639-2* e non *ISO 639-3* anche se quest'ultimo ha ormai preso piede. Dovendo chiarire: per analizzare le pagine presenti nei file *WET* la versione uno dello standard sarebbe stata più che sufficiente, infatti in *dictionary.json* sono presenti anche i tag relativi a *ISO 639-1*. Si è scelto di ampliare la specifica dei tag a *639-2* per poter facilmente confrontare i risultati dell'analisi con quelli proposti da *CLD2*.

All'interno di *dictionary.json* le lingue implementate sono poco meno di 70, tuttavia solo per una parte di esse è stata ottimizzata e testato approfonditamente il rilevamento all'interno del dataset. Una lingua è implementata nel dizionario se la sua lista di *words* non è vuota.

Qui sotto un piccolo estratto del dizionario relativo alle lingue Afrikaans e Tedesco.

```
{
  "name": "dictionary",
  "type": "300_isoV2",
  "version": "1.1.6",
  "languages": [
    { "name": "Afrikaans",
      "ISO_639_V1": "af",
      "ISO_639_V2": "afr",
      "words": ["as", "Ek", "sy", "wat" ...] },
    ...
    { "name": "German",
      "ISO_639_V1": "de",
      "ISO_639_V2": "deu",
      "ISO_639_V2_alt": "ger",
      "words": ["ich", "sie", "das", "ist" ...] },
    ...
  ]
}
```

Per realizzare il dizionario sono state utilizzate alcune fonti che propongono un elenco delle parole più usate in ogni lingua.[1][2]

### 3.4 La classe *Map*

La classe *Map* che estende *Mapper* è ovviamente l'omonima nel processo MapReduce, il suo funzionamento è già stato spiegato brevemente nelle sezioni precedenti, rimangono però da dettagliare:

- Come il mapper passa dal testo della pagina all'insieme di parole associate.
- Come vengono rappresentate le lingue e il dizionario all'interno del tool MapReduce.

- Qual è l'esatta modalità in cui vengono calcolate le statistiche, inoltre come si passa da queste ultime alle lingue scelte come rappresentative per una determinata pagina e dunque per un determinato url.
- Quale ottimizzazione è stata adottata per migliorare il riconoscimento delle lingue orientali.

Verranno ora spiegati dettagliatamente tali punti.

### 3.5 Page split tramite RegExp

Per dividere il testo di una pagina nelle parole che lo compongono, è stata utilizzata una Regular Expression, in modo da ottenere il risultato più ottimale possibile. Questa fase è molto importante poiché determina l'insieme di parole su cui verrà fatta l'analisi della lingua.

```
private static final Pattern WORD_BOUNDARY_V2 = Pattern.compile(
    "[\\[0-9\\]\\\\s!\\\\-\\$€%^&*()_+|~='\"{ }\\\\[\\\\]°: \";'<>?,@#.\\\\/•]+");
```

La regular expression descrive cosa separa una word da un'altra, questo insieme di caratteri individuati dall'espressione regolare verrà usato dal mapper per dividere il *plain text* della pagina nell'insieme di parole corrispondenti. La RegExp costruita appositamente per questa analisi, considera come caratteri di separazione uno o più dei seguenti: cifre, spazi, punteggiatura, parentesi e caratteri vari che non possono appartenere a una parola.

A seguito un semplice esempio di suddivisione di un frammento di una pagina, nelle parole che lo compongono. Quelle evidenziate sono le *word*.

```
Profile views: 38,129 *
Last seen: 06/07/2018 - 21:06
All times are now 15/02/2019, 22:34
4 posts (0 per day)
Contact Information
• HERRY index.? System
```

### 3.6 Language, Dictionary e JsonParser

Le classi *Language* e *Dictionary* vengono usate principalmente dal mapper per l'analisi della lingua, la loro rappresentazione è intuitiva. La prima modella una lingua con i relativi tag; oltre a quello primario ce n'è anche uno alternativo, nel caso in cui la lingua ne abbia due. I tag di *ISO 639-1* sono stati disabilitati dopo l'introduzione nel progetto della specifica *639-2*.

```
public class Language{
    private String name; //nome della lingua in formato testo
    //private String tag_ISO_639_V1; //2 char
    //private String tag_ISO_639_V1_alt; //2 char
    private String tag_ISO_639_V2; //(T) 3 char
```

```
private String tag_ISO_639_V2_alt; //(B) 3 char
...
```

*Dictionary* rappresenta invece una lista di parole, presenti sul dizionario, con la relativa lingua di appartenenza.

```
public class Dictionary {
    private String dictName;
    private String dictType;
    private String dictVersion;
    private HashMap<String, Language> dict; //{word -> lang}*
    ...
}
```

*JsonParser* implementa il metodo *loadDictionary()* che viene invocato nella fase di *setup* di ogni mapper per caricare sull'istanza di *Dictionary* le parole presenti in *dictionary.json*. Il processo di *parsing* del *json* sfrutta una libreria standard.

### 3.7 Le statistiche

Le statistiche rappresentano la percentuale di presenza di ogni lingua su una pagina; tale percentuale è calcolata in base al numero di parole nel testo che vengono identificate per ogni lingua. Le statistiche sono fondamentali poiché sono la base con la quale il tool determinerà una o più lingue da associare all'url e dunque alla pagina.

$$percentuale\_lang1 = \frac{\#parole\_lang1}{\#parole\_totali\_sulla\_pagina} (\cdot 100)$$

Come già descritto precedentemente in output vengono mostrate al massimo cinque lingue, quelle aventi la percentuale più alta. I valori delle lingue rimanenti sono raggruppati in 'Other\_Langs'. Viene calcolata anche la percentuale di parole presenti sul testo per cui non è stata trovata corrispondenza in una lingua: 'Not\_Found'. Ogni lingua è rappresentata dal suo tag *ISO 639-2*. A seguito un esempio di statistiche (già riportato precedentemente).

```
eng:6.91%; slv:1.22%; slk:1.22%; swa:1.22%; ron:1.22%;
Other_Langs:4.88%; Not_Found:83.33%
```

### 3.8 Scelta della lingue rappresentative di una pagina

In *LanguageElect* viene dunque dato un penso ad ogni lingua in base al numero di parole presententi per essa sulla pagina; si trasformano poi questi punteggi in statistiche come visto nella sezione precedente; infine si passa a scegliere le lingue rappresentative per la pagina. Tale scelta si basa nel selezionare le lingue che hanno una percentuale più alta ma la scelta deve essere più studiata: se ci si limita a prendere le due o tre lingue con la percentuale maggiore si corre il rischio di fare degli errori. Immaginiamo di voler ritornare sempre le due lingue con percentuale più alta e avere banalmente delle statistiche di questo

tipo: `eng:6.12%;mlt:0.95%;slv:0.93%`; andremmo quindi a selezionare sia l'inglese che il maltese. Tuttavia ci rendiamo subito conto che il maltese pur avendo la seconda percentuale più alta è probabilmente un'errore della nostra analisi, visto che la differenza di grandezza con il valore dell'inglese è troppo ampia. Per evitare di prendere le lingue con una percentuale troppo bassa e in generale per ottimizzare questo importante procedimento, facciamo dei passaggi:

- Impostiamo un limite minimo che deve avere la percentuale di presenza di una lingua per non essere considerata un errore di analisi. Tale valore dopo alcune valutazioni è stato settato a `VALID_LANGUAGE_THRESHOLD = 1.0`. Tutte le lingue che hanno una percentuale sotto l'1% verranno dunque scartate a meno della condizione seguente.
- Se nessuna lingua supera l'1% allora pur rendendoci conto che i risultati ritornati potrebbero essere non rappresentativi, ritorniamo comunque la o le percentuali che tra tutte risultano essere le più elevate.
- Impostiamo un massimo numero di lingue che potranno essere ritornate dalla nostra valutazione, tale numero è calcolato appositamente ma alla base c'è la costante `MAX_NUMBER_LANG_TO_SHOW = 6`.
- Impostiamo `VALID_LANGUAGE_MAX_STEP = 3` che rappresenta la massima differenza che può esserci tra la percentuale più alta e le altre percentuali che vengono selezionate. e.g. `jpn:19.83%;zho:2.96%;ron:0.49%;swe:0.25`; in questo caso sia il giapponese(*jpn*) che il cinese(*zho*) superano la soglia minima. Il primo possiede tuttavia una percentuale molto più elevata rispetto al secondo e dunque nell'ottica che *zho* sia un errore, ha senso ritornare come lingua significativa per la pagina solo il giapponese. Nella pratica, data la lingua con la percentuale più alta, vengono tenute le lingue che non differiscono dalla prima più del `VALID_LANGUAGE_MAX_STEP`.
- Seguendo l'idea del punto precedente possiamo però generare un ulteriore errore nella valutazione. Supponiamo di avere ad esempio `eng:31.23%;jpn:25.83%;zho:1.16%`; in questo caso la nostra analisi ritornerebbe solo l'inglese, poichè `%eng - %jpn > VALID_LANGUAGE_MAX_STEP = 3`. Ragionando sui risultati ottenuti capiamo tuttavia che rimuovere il giapponese che ha una percentuale così alta è molto probabilmente un errore, infatti la pagina potrebbe essere scritta in due diverse lingue. Per evitare quindi di rimuovere delle lingue con una percentuale così elevata, pur mantenendo l'idea relativa al punto precedente, introduciamo un limite oltre il quale l'eliminazione con lo *step* non verrà più effettuata. Tale limite è `LIMIT_OF_CLEAR_PERCENTAGE = 10`. In questo modo nell'esempio verrà mantenuto sia l'inglese che il giapponese.

Scelte le lingue rappresentative della pagina, *LanguageElect* le ritornerà con le statistiche alla classe principale *Map*, quest'ultima effettuerà l'*emit* come descritto precedentemente.

### 3.9 Ottimizzazione del tool per le lingue orientali

L'operazione di individuazione delle lingue in pagine con testi scritti in lingue orientali, ad esempio cinese e giapponese, se non correttamente ottimizzata, risulta avere un grado di errore elevatissimo. La motivazione è nell'esempio seguente. Si supponga di avere un testo di questo tipo:

邻居的猫是红色的我的猫是绿色的蓝色条纹

In cinese quella sopra è una intera frase ma con le nostre tecniche di divisione del testo ovviamente risulterà essere una sola parola, non vi sono infatti spazi tra un 'simbolo' e l'altro. Il risultato di un'analisi su questo testo darebbe perciò zero *word* appartenenti alla lingua cinese. La soluzione adottata per risolvere questo problema è molto semplice e intuitiva ma anche efficace: se una parola inizia con un carattere orientale, la stessa viene suddivisa nei singoli caratteri che la compongono e prendendo i caratteri consecutivi a due a due. Ognuno di questi caratteri o di queste coppie di caratteri diventerà una nuova *word*. Nell'esempio precedente:

邻,邻居,居,居的,的,的猫,猫...

In questo modo i risultati migliorano notevolmente e si raggiunge quasi sempre il risultato proposto da *CLD2*. Qui sotto un piccolo estratto del codice che permette di individuare le parole in lingue orientali per poi eseguirne il successivo *split*.

```
if((word.charAt(i)>='\u3041' && word.charAt(i)<='\u309F')|| //[1]
    (word.charAt(i)>='\u30A0' && word.charAt(i)<='\u30FF')|| //[2]
    (word.charAt(i)>='\u31F0' && word.charAt(i)<='\u31FF')|| //[3]
    (word.charAt(i)>='\u3190' && word.charAt(i)<='\u319F')|| //[4]
    (word.charAt(i)>='\u4E00' && word.charAt(i)<='\uA000')|| //[5]
    (word.charAt(i)>='\u3400' && word.charAt(i)<='\u4DC0')|| //[6]
    (word.charAt(i)>='\uF900' && word.charAt(i)<='\uFB00')|| //[7]
    (word.charAt(i)>='\u9FA6' && word.charAt(i)<='\u9FCC')) //[8]
    //[1] -> Unicode - Hiragana      [2] -> Unicode - Katakana
    //[3] -> Unicode - Kat. Phon.Ext. [4] -> Unicode - Kanbun
    //[5] -> Han Ideogr. - Chinese A  [6] -> Han I. Chinese B
    //[7] -> Han I. Chinese C          [8] -> Han I. Chinese D
) { asian = true; }
...
```

### 3.10 Reduce e ResultChecker

La classe *Reduce* estende *Reducer* e ovviamente ne implementa il metodo *reduce()*. Come detto precedentemente, quest'ultimo riceve le due tipologie di coppie  $\langle key, value \rangle$  e le mette assieme per poi effettuare un ultimo *emit*, la tupla verrà quindi memorizzata sul relativo file nella cartella di output in *hdfs*. Nel fare l'emissione del risultato, il reducer aggiunge il carattere che identifica quanto



è stato preciso *LanguageElect* a valutare la/le lingue della pagina e riporterà il tag [WS] se è stato fatto lo *split* relativo alla lingue orientali. La precisione ricordiamo ancora una volta essere calcolata rispetto ai risultati di *CLD2*. Le operazioni di confronto per ricavare la precisione sono svolte da *ResultChecker* (che viene chiamato dal reducer) e sono delle semplici operazioni di confronto tra i due insiemi di lingue.

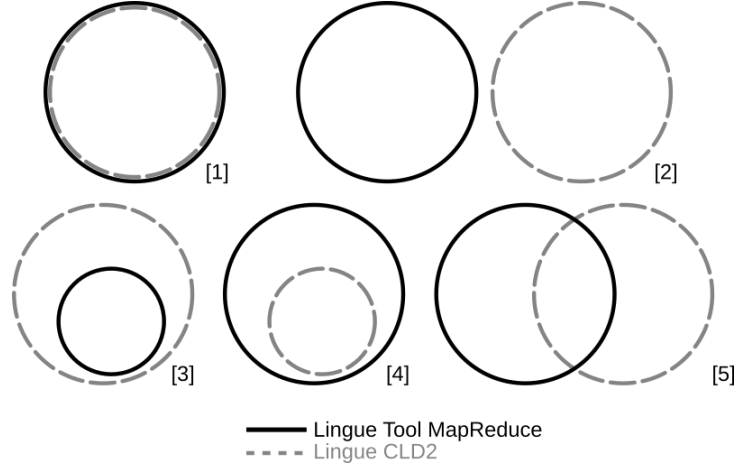


Figura 2: Precisione del tool MapReduce

Descriviamo qui a seguito tutti i simboli che indicano la precisione dell'analisi effettuata sulla pagina rispetto al risultato indicato da *CLD2*, sia a parole che con l'indicazione degli insiemi [Figura 2]. Verrà rappresentato con *ToolLangs\_Set* l'insieme delle lingue trovate dal tool MapReduce e con *CLD2Langs\_Set* l'insieme di lingue che propone *CLD2* per l'url corrente.

- [1] ✓ : indica che MapReduce ha rilevato lo stesso identico insieme di lingue proposte da *CLD2*.  $ToolLangs\_Set = CLD2Langs\_Set$
- [2] ✗ : nessuna delle lingue rilevate con il tool coincide con quelle di *CLD2*.  $ToolLangs\_Set \cap CLD2Langs\_Set = \emptyset$
- [3] — : il tool MapReduce ha proposto meno lingue di quelle rilevate da *CLD2*.  $ToolLangs\_Set \subset CLD2Langs\_Set$
- [4] + : il tool ha ritornato come risultato le lingue di *CLD2* ma ne ha proposte anche altre che *CLD2* non ha rilevato. MapReduce ha dunque rilevato più lingue del previsto.  $CLD2Langs\_Set \subset ToolLangs\_Set$
- [5] ÷ : il tool realizzato ha proposto alcune lingue che *CLD2* non ha rilevato e ci sono anche alcune lingue che *CLD2* ha rilevato ma che il tool non ha segnalato.  $(ToolLangs\_Set \cap CLD2Langs\_Set \neq \emptyset) \wedge (ToolLangs\_Set \not\subset CLD2Langs\_Set) \wedge (CLD2Langs\_Set \not\subset ToolLangs\_Set)$

## 4 Test su cluster e Risultati finali

Sono stati svolti numerosi test su cluster, a seguito viene riportato come esempio l'esecuzione del tool MapReduce su 2 file *WET*, con 5 reducer.

```
st-crosara@hadoopmaster:~$ hadoop jar analyzer.jar 5 /user/st-cro
sara/wet/ /user/st-crosara/info/ /user/st-crosara/dictionary.json
/user/st-crosara/output_2files_5reducer
07:18:42 INFO client.RMPProxy: Connecting ...aster/10.0.2.100:8032
07:18:42 INFO input.FileInputFormat: Total input files to ... : 2
07:18:42 INFO input.FileInputFormat: Total input files to ... : 2
07:18:42 INFO mapreduce.JobSubmitter: number of splits:14
07:18:42 INFO mapreduce.JobSubmitter: ... job_1552309452177_3067
07:18:42 INFO impl.YarnClientIm... application_1552309452177_3067
07:18:42 INFO mapreduce.Job: The url to track the job: ...
07:18:42 INFO mapreduce.Job: Running job: job_1552309452177_3067
07:18:48 INFO mapreduce.Job: Job job_1552309452177_3067 ...
07:18:48 INFO mapreduce.Job: map 0% reduce 0%
...
07:19:59 INFO mapreduce.Job: map 100% reduce 100%
07:19:59 INFO mapreduce.Job: Job ... completed successfully
07:19:59 INFO mapreduce.Job: Counters: 52
File System Counters
  FILE: Number of bytes read=1804556693
  FILE: Number of bytes written=2780400941
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=1619778453
  HDFS: Number of bytes written=16734386
  HDFS: Number of read operations=57
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=10
Job Counters
  Killed map tasks=3
  Killed reduce tasks=1
  Launched map tasks=16
  Launched reduce tasks=6
  Data-local map tasks=13
  Rack-local map tasks=3
  Total time spent by all maps in occupied slots (ms)=1569702
  Total time spent by all reduces in occupied s... (ms)=1246350
  Total time spent by all map tasks (ms)=261617
  Total time spent by all reduce tasks (ms)=207725
  Total vcore-milliseconds taken by all map tasks=261617
  Total vcore-milliseconds taken by all reduce tasks=207725
```

```

    Total megabyte-milliseconds taken by all map tasks=1607374848
    Total megabyte-millis... taken by all reduce tasks=1276262400
Map-Reduce Framework
  Map input records=11491183
  Map output records=11491175
  Map output bytes=949070828
  Map output materialized bytes=972860698
  Input split bytes=3258
  Combine input records=0
  Combine output records=0
  Reduce input groups=11197552
  Reduce shuffle bytes=972860698
  Reduce input records=11491175
  Reduce output records=88791
  Spilled Records=32845860
  Shuffled Maps =70
  Failed Shuffles=0
  Merged Map outputs=70
  GC time elapsed (ms)=1997
  CPU time spent (ms)=305350
  Physical memory (bytes) snapshot=13561917440
  Virtual memory (bytes) snapshot=139105951744
  Total committed heap usage (bytes)=12807307264
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=16734386

```

Il risultato del processo MapReduce non viene analizzato ora poiché, appurata la completa esecuzione del job senza errori, la parte rilevante risulta essere unicamente legata ai tempi di esecuzione. In particolare è interessante notare come cambiano le tempistiche, rispetto alla variazione del numero di file in input al tool; tale correlazione verrà discussa in seguito. Notiamo tuttavia un dettaglio: durante l'esecuzione dello script MapReduce, quando Hadoop ci mostra le percentuali di completamento delle due fasi (rimosse nel risultato sovrastante per questioni di spazio), la fase di reduce sembra iniziare prima che quella di map sia stata completata:

```

...
19/03/24 11:50:25 INFO mapreduce.Job:  map 84% reduce 19%

```

```
19/03/24 11:50:27 INFO mapreduce.Job: map 87% reduce 24%
...
```

Tale particolarità si nota poco quando si fa funzionare lo script su piccoli input di test ma nel nostro caso è invece molto evidente. Per ovvi motivi i reducer non possono iniziare a elaborare i risultati dei mapper prima di essere sicuri che questi ultimi abbiano finito di emettere tutte le coppie  $\langle key, value \rangle$ , ma possono comunque fare lo *shuffle*. La fase di *shuffle* consiste concettualmente in un mix di sorting e group-by, ovvero vengono raggruppate tutte le coppie  $\langle key, value \rangle$  per chiave e si ordinano in modo da poter eseguire al meglio le operazioni nel reducer. Questa fase può quindi iniziare prima che i mapper abbiano completato tutti gli *emit*.

Un piccolissimo estratto del risultato di MapReduce:

```
http://42stocks.com/cgi-bin/symbol.py?symbol=mtnb Response:eng |
Expected:eng ✓ eng:6.91%;slv:1.22%;slk:1.22%;swa:1.22%;ron:1.22%
;Other_Langs:4.88%;Not_Found:83.33%
http://43.fsin.su/index.php?day=6&month=12&year=2018 Response:ukr
,rus | Expected:rus + ukr:1.93%;rus:2.9%;srp:0.48%;Other_Langs:0
.0%;Not_Found:94.69%
http://monolite.vk-online.ru/kods-film-ple-xfo.html Re
sponse:ukr | Expected:rus ✗ ukr:10.29%;srp:2.94%;rus:5.88%;ceb:1
.47%;Other_Langs:0.01%;Not_Found:79.41%
http://45.77.110.76/ Response:eng,slv,slk | Expected:eng + eng:2
.22%;slv:2.22%;slk:2.22%;Other_Langs:0.01%;Not_Found:93.33%
http://4510m.in/tag/18.../ Response:jpn | Expected:jpn ✓ jpn:5.6
2%;slv:0.35%;zho:1.87%;swe:0.31%;spa:0.35%;Other_Langs:2.3%;Not_F
ound:89.2%[WS]
http://45rpmrecords.com/KY/Summit.php Response:eng | Expected:eng
✓ eng:2.47%;tur:0.93%;sqi:0.93%;slv:0.93%;nor:0.31%;Other_Langs
:2.45%;Not_Found:91.98%
http://4646234.org/www_26701_com/baixiaojie...html Response:zho
| Expected:zho ✓ jpn:4.55%;sqi:0.23%;cym:0.23%;zho:10.24%;swe:0.
23%;Other_Langs:1.24%;Not_Found:83.28%[WS]
```

#### 4.1 Accuratezza dei risultati e tempo di esecuzione

Andando a contare il numero di linee in output del tool MapReduce per ogni livello di accuratezza, i risultati ottenuti dall'analisi di due file *WET* e dunque su un totale di 88791 linee di output sono: 49668✓, 5371✗, 14058+, 14941-, 4753÷. Visualizzando tali valori su un istogramma il risultato è quello riportato in [Figura 3]. Tenendo conto di tutti i miglioramenti che si potrebbero ancora apportare per ottimizzare l'analisi, i risultati ottenuti sono molto buoni e superiori alle aspettative.

Risulta molto indicativo osservare quanto impiega il tool a terminare il processo MapReduce sul cluster, per questo è stato realizzato un grafico [Figura 4] che relaziona il numero di file *WET* analizzati e il tempo di esecuzione. I valori

dei tempi mostrati tengono conto solo del processo di Map e di quello di Reduce, non viene quindi sommato il tempo di 'setup' iniziale del *Driver*. Guardando l'andamento, possiamo notare che al crescere del numero di file *WET* elaborati dal tool il tempo di esecuzione non cresce proporzionalmente. Per essere più precisi: se rapportiamo il tempo di esecuzione ottenuto con due file in input a 4, 6, 8 e 10 file input, otteniamo quella che nel grafico è la linea grigia; i risultati reali sono invece rappresentati dalla linea blu. Il cluster risulta quindi comportarsi bene al crescere delle dimensioni dell'input, come d'altronde ci si aspetta.

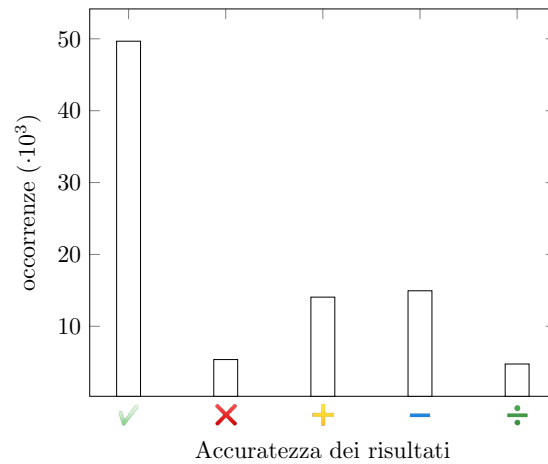


Figura 3: Istogramma dell'accuratezza dei risultati

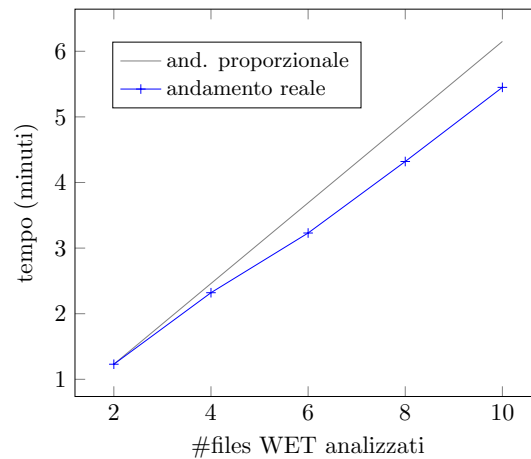


Figura 4: Grafico del tempo di esecuzione dei job MapReduce

## 5 Considerazioni conclusive

Nel tool realizzato il punto focale è stato la progettazione di una buona struttura MapReduce, più che la ricerca di un sistema che permettesse di fare un'analisi linguistica il più precisa e completa possibile. Lo stato dell'arte per l'analisi delle lingue prevede l'uso di *machine learning* oppure nei sistemi più basilari uno studio di tipo *letter frequency* nelle parole. Visto l'obiettivo limitato e non avendo sufficienti conoscenze di linguistica per uno studio completo, non ci si è soffermati sulla ricerca del metodo migliore per fare *language detection* ma si è scelto di sviluppare la tecnica intuitiva di riconoscere la lingua tramite un confronto con le parole più usate nella stessa. Nonostante questa 'semplificazione' è stato possibile ottenere una buona precisione nei risultati, l'accuratezza ha inoltre un ampio margine di miglioramento, che può essere ottenuto tramite perfezionamento del dizionario e aggiungendo condizioni specifiche per ogni lingua. In ottica quindi di voler migliorare il tool realizzato e la relativa analisi delle lingue, i punti su cui concentrarsi in futuro potrebbero essere:

- Aumentare il numero di lingue analizzabili dal tool, implementandone di nuove sul dizionario.
- Ottimizzare il dizionario stesso e le lingue già implementate: è possibile ad esempio migliorare i *word set* facendo riferimento a più fonti e valutando di togliere le parole che pur essendo molto utilizzate in una lingua sono molto presenti anche in altre. Si consideri ad esempio le parole molto corte che possono facilmente avere un significato in molti paesi, queste ultime possono peggiorare notevolmente i risultati dell'analisi poiché rendono difficile disambiguare le diverse lingue.  
e.g. 'as' è una parola molto diffusa che ha diversi significati in inglese, ceco, afrikaans, portoghese e molte altre lingue.
- Migliorare l'analizzatore con tecniche di *letter frequency* sulle parole individuate nelle pagine.
- Raffinare le statistiche e le tecniche usate per la scelta finale delle lingue, date le percentuali di presenza di ciascuna di esse. Imparando le caratteristiche sintattiche e semantiche di ciascuna lingua è possibile inoltre creare delle strutture nel codice che possano aiutare nell'individuazione e nella disambiguazione. Un esempio è la già implementata divisione in sottostringhe delle parole relative ai testi delle lingue orientali per migliorare il riconoscimento che viene ottenuto dalle stesse.

Ovviamente per molti dei punti qui descritti sarebbe necessario avere una buona conoscenza linguistica per poter conoscere i metodi chiave su cui far leva per migliorare il riconoscimento di una lingua. Nel concludere possiamo dunque considerare raggiunto l'obiettivo che è stato posto all'inizio: la struttura MapReduce implementata nel tool realizzato si è rivelata adatta al tipo di *dataset analysis* effettuata, i risultati ottenuti si sono rivelati essere molto buoni nonostante la semplicità della tecnica scelta per l'individuazione della lingua.

## Riferimenti bibliografici

- [1] 1000 most common words. [www.1000mostcommonwords.com](http://www.1000mostcommonwords.com).
- [2] 101 languages. [www.101languages.net/common-words/](http://www.101languages.net/common-words/).
- [3] Common crawl. [www.commoncrawl.org](http://www.commoncrawl.org).
- [4] Glossario: Crawler. [www.tagmanageritalia.it/glossario-crawler](http://www.tagmanageritalia.it/glossario-crawler).
- [5] Iso 639. [www.iso.org/iso-639-language-codes.html](http://www.iso.org/iso-639-language-codes.html).