PROGETTO NLP A.A. 2022-2023

INFORMATICA MAGISTRALE
– UNIVERSITÀ DI BOLOGNA
PROFESSOR FABIO TAMBURINI

REPORT

Sistema di traduzione automatica da frasi in linguaggio naturale a frasi in simboli C.A.A. in lingua italiana [1]

Sofia Cerè **Nicandro Potena**

Contents

1 Introduzione

1.1 Sistemi di C.A.A. e simboli C.A.A.

I **sistemi di Comunicazione Aumentativa e Alternativa (C.A.A.)** sono forme di espressione diverse dalla lingua parlata, che mirano ad aumentare (aumentantative) e/o a compensare (alternative) le difficoltà di comunicazione e di linguaggio verbale di molte persone con disabilità. [9]

La C.A.A. è composta da molti strumenti differenti, ma di solito utilizza simboli, ovvero immagini che rappresentano i concetti. Solitamente ogni simbolo C.A.A è accompagnato da un'etichetta testuale (parola) ed è racchiuso all'interno di un quadrato.
Qui di seguito un esempio dei simboli rappresentanti la frase "ciao mondo" 1 tramite il sistema simbolico Arasaac:

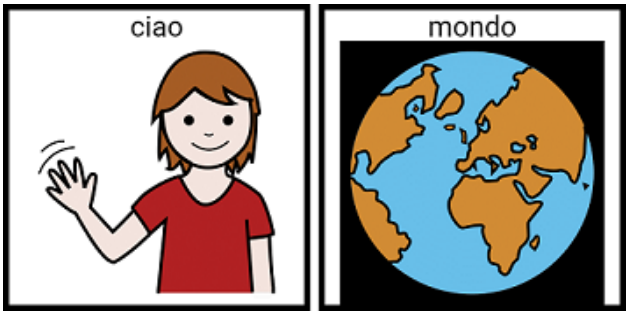

Figure 1: Frase "Ciao Mondo" in C.A.A.

Il sistema simbolico Arasaac dispone di un'amplia raccolta di circa 10.000 simboli (anche detti pittogrammi).

Principale punto di forza di Arasaac è la sua disponibilità in forma open source, insieme a software per l'elaborazione dei contenuti e per il loro utilizzo con dispositivi mobili.

Questo approccio è anche quello che sembra favorire una maggiore possibilità di intervento evolutivo, fatta salva la necessità di una solida cornice concettuale e scientifica.
Esiste una presentazione di Sergio Palao, il produttore dei simboli in Arasaac ,che descrive molto sinteticamente la logica della sua produzione. 2
Sembrano esistere 8 possibili rappresentazioni per ciascun referente, per ciascuna parola, sulla base di tre caratteristiche:

- stile: descrittivo, schematico
- colore: col, bn
- genere: maschile, femminile [3]

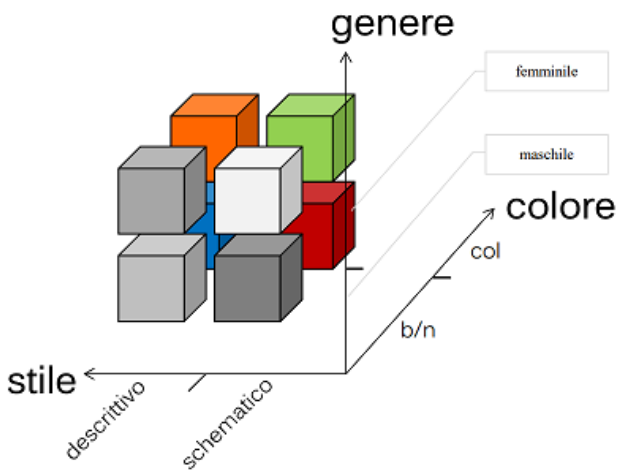

Figure 2: Frase "Ciao Mondo" in C.A.A.

Oltre alle 8 possibili rappresentazioni ne sono state inserite ulteriori, quali l'indicazione di plurale/singolare, di negazione e di passato/futuro nel caso dei verbi.
Queste caratteristiche sono rappresentate attraverso dei watermark all'interno del simbolo C.A.A.

I watermark nel sistema Arasaac sono posizionati in alto a destra all'interno del riquadro contenente il pittogramma e l'etichetta testuale.
Il watermark che indica il plurale è rappresentato dal simbolo +. 3

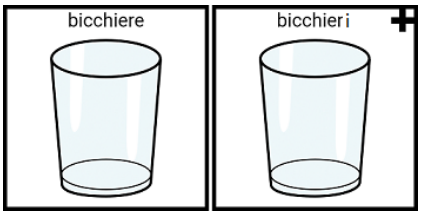

Figure 3: Uso del watermark del plurale

Qui di seguito un esempio di uso dei watermark per il verbo "mangiare": rispettivamente l'uso della freccia indietro per indicare il passato e l'uso della freccia avanti per indicare il futuro.4

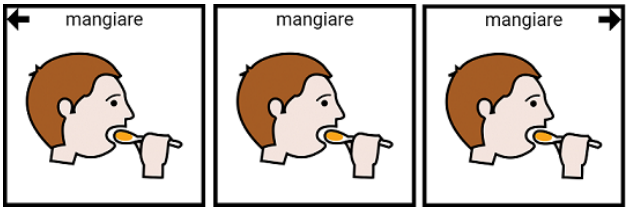

Figure 4: Uso dei watermark dei tempi verbali con il verbo mangiare

Nel sistema Arasaac l'etichetta testuale di un verbo è rappresentata dalla forma dell'infinito del verbo, ma solitamente per rafforzare la comprensione e lo sviluppo dell'uso del linguaggio verbale l'etichetta testuale del verbo viene cambiata coerentemente alla forma verbale. 5

Per le peculiarità sopra elencate si è scelto di adottare Arasaac come sistema simbolico per lo svolgimento del progetto di NLP.

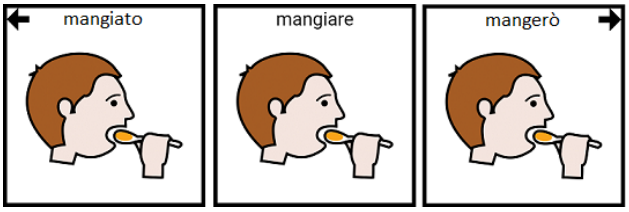

Figure 5: Uso dei watermark dei tempi verbali con il verbo mangiare coniugato

1.2 Descrizione del problema

Ad oggi esistono diversi editor per la creazione di frasi in C.A.A. in lingua italiana, ma come indicato nel documento "Comunicatori e compositori simbolici" del Centro Sovrazonale di Comunicazione Aumentativa [2], vi è la mancanza di un supporto della composizione di frasi sintatticamente corrette, oltre che semanticamente significative.
Per creare una possibile risposta all'esigenza appena espressa si è scelto di realizzazione un traduttore automatico da frasi in linguaggio naturale in lingua italiana a frasi in simboli C.A.A. in lingua italiana attraverso l'impiego di strumenti di NLP (Natural Language Processing), quali il POS tagging, la tokenization e lemmatization per estrapolare le informazioni necessarie al fine di effettuare una traduzione automatica.

1.3 Descrizione della soluzione proposta

Nella nostra soluzione la frase da tradurre viene data in input ad una raccolta di strumenti accurati ed efficienti per l'analisi linguistica denominata Stanza.
Viene utilizzata la pipeline di Stanza che consiste in tokenization, lemmatization e POS tagging.
In questo modo otteniamo tutte le informazioni che riguardano

ogni parola della singola frase, quali il tipo, il numero, il genere e, nel caso dei verbi, il tempo verbale.
Una volta ottenute le informazioni, sono state impiegate le API (Application Programming Interface) di Arasaac per trovare il simbolo C.A.A. corrispondente alla parola (o in alcuni casi al gruppo di parole) ed aggiungere eventuali watermark. La spiegazione di questa procedura avviene nel capitolo 3.

1.4 Presentazione dei risultati ottenuti

Il calcolo dei risultati ottenuti può essere considerato soltanto come approssimativo, per una serie di motivi. Il problema principale che non permette una valutazione delle performance ottimale è che non esistono dataset con traduzioni da frasi in linguaggio naturale in frasi in pittogrammi Arasaac in lingua italiana, ma esistono solo un set di pochi documenti formato principalmente da storie per bambini tradotte in pittogrammi.
In [8], che sviluppa il progetto esposto in [15], hanno impiegato un Book Corpus per svolgere la valutazione dei risultati, allo stesso modo nel progetto in esame si è scelto di impiegare la stessa procedura, avendo però le seguenti limitazioni:

- Per effettuare un confronto con i pittogrammi scelti dal sistema di traduzione sono stati estratti gli identificativi dei pittogrammi del Book Corpus manualmente, tramite l'individuazione degli stessi all'interno del database Arasaac. Tale individuazione non permette quindi di estrarre un grande numero di identificativi essendo una procedura lenta;
- Molti dei pittogrammi presenti nel Book Corpus non sono

presenti attualmente nel sistema C.A.A. di Arasaac, si presuppone quindi che essi siano pittogrammi datati, eliminati in seguito ad aggiornamenti, oppure pittogrammi creati ad hoc dai traduttori per essere impiegati all'interno della storia narrata;

- Una parola potrebbe corrispondere a più pittogrammi presenti su Arasaac rappresentanti lo stesso significato semantico. Per cui una frase potrebbe avere diverse rappresentazioni pur mantenendo lo stesso significato semantico.
  Dunque si reputa che la miglior valutazione della correttezza di questo sistema dovrebbe essere effettuata tramite un giudice umano in grado di interpretare il significato delle traduzioni e fornirne un giudizio.

Tenendo ben presenti le premesse appena descritte si è preso in considerazione un Book Corpus formato da 40 frasi, con un totale di 189 parole, reputate aggiornate e corrette. Successivamente è stato effettuato un confronto tra le sequenze di pittogrammi estratte dal nostro sistema, date le frasi, e le sequenze di pittogrammi presenti nel corpus.
Nello specifico si è deciso di calcolare la performance del sistema utilizzando la classe *SequenceMatcher* della libreria *difflib*, che da la possibilità di confrontare due o più *Integer* serie anche con lunghezze diverse. In questo caso, la precisione è data dalla media dei risultati dei confronti su ciascuna frase, e risulta del 43%.
Riteniamo che, avendo a disposizione un ampio dataset con pittogrammi aggiornati, la precisione tenderebbe ad aumentare.

2 Metodo proposto

2.1 Ricerca della soluzione e giustificazione della scelta

La scelta della libreria di NLP da utilizzare per sviluppare il progetto è stata effettuata dopo averne testate quattro, tra cui: *Spacy [11], NLTK [7], Tint [14], Stanza [12].* Tali librerie, sia per la fase di Pos-Tagging, che per quella di Lemmatization, ovvero per le fasi più rilevanti per il progetto in analisi, registrano un accuratezza media molto alta in riferimento alla lingua italiana, cioè tra il 97% e il 98%. Dopo aver testato le librerie sul corpus di frasi del progetto si è ritenuto Stanza lo strumento migliore da impiegare sia per i migliori risultati ottenuti sia per la facilità d'integrazione in *Python.*

3 Risultati sperimentali

3.1 Elenco delle tecnologie usate per gli esperimenti

Il progetto è stato sviluppato in Python nella versione 3.9.6, sia sulla piattaforma Google Colab [6] che sull'IDE Visual Studio Code. Le librerie Python utilizzate sono state *pandas, numpy, request, wordnet, PIL, stanza, matplotlib, PySimpleGUI, difflib, WordNet.*
É possibile visionare il codice sorgente su [5]

3.1.1 Stanza

Stanza[12] è un pacchetto di analisi del linguaggio naturale, costruito con componenti di rete neurale. La pipeline di Stanza permette una completa analisi del testo, tra cui tokenization,

Text input	Maria	era	molto	stanca
Tokenization	Maria	era	molto	stanca
POS Tagging	PROPN	AUX	ADV	ADJ
Lemmatization	Maria	essere	molto	stanca
Pictogram ID	7301	5465	5521	2314

Table 1: Analisi effettuata dalla pipeline di Stanza della frase "Maria era molto stanca".

multi-word token espansion (MWT), lemmatization, part-of-speech (POS) ed altre funzionalità aggiuntive. Le performance di Stanza per ciascuna fase della pipeline, per il linguaggio italiano, sono:
-Tokenization: 99.94%
-POS: 97.93%
-Lemmatization: 98.01%

Come mostrato nella tabella 1 l'analisi effettuata dalla pipeline di Stanza fornisce le informazioni di ciascuna parola. Inoltre, il modulo POS Tagging, effettua anche un'analisi morfologica, fornendoci le "features" delle parole, ovvero *gender, number, e mood, tense, person* per i verbi come illustrato nel seguente codice:

```
stanza.download('it')
nlp=stanza.Pipeline('it',processors='tokenize,pos,lemma',
    tokenize_pretokenized=True)
frase=re.sub(r"[-()\"/@;:<>{}`+=~|$%&]", "", frase)
doc=nlp(frase)
for sentence in doc.sentences:
    posT=[(w.text, w.pos,w.lemma,w.feats) for w in sentence.
    words]
```

Successivamente per sfruttare al meglio le API di Arasaac, andremo ad effettuare una fase di pre-processing delle informazioni ottenute dalla pipeline di Stanza, in particolare delle informazioni

delle ”features” delle parole. Un esempio, per quanto riguarda le parole di tipo ”VERB”, è quello di estrazione del tempo verbale (Arasaac supporta solo due tempi verbali: passato o futuro):

```
    if (type == 'VERB'):
        if not word[3].split("=")[1]=="Inf":
            tense=word[3].split("|")[2].split("=")[1]
            if(tense=="Past" or tense=="Imp"):
              action="past"
            if(tense=="Fut"):
              action='future'
```

Oltre al tempo verbale, un altro elemento estratto è stato il numero, ovvero singolare o plurale. Le informazioni estratte sono state impiegate per effettuare una richiesta API ad Arasaac più specifica, includendo tra i parametri tempo e numero. In tal caso, i pittogrammi ottenuti presenteranno i watermark nella forma esposta nella sezione 1.1 che li distingueranno dai pittogrammi standard.

3.1.2 API Arasaac

Per la ricerca dell'identificativo del pittogramma del sistema simbolico Arasaac corrispondente ad una determinata parola sono state impiegate le API messe a disposizione dal Governo aragonese, gestori di Arasaac.
Sono disponibili alcune varianti per ogni pittogramma corrispondenti ai seguenti possibili parametri aggiuntivi:

- *action* per il tempo verbale (past, future);
- *hair* per il colore dei capelli (brown A65E26, blonde FDD700, red ED4120, black 020100, gray EFEFEF, darkGray AAABAB, darkBrown 6A2703);

- *skin* per il colore della pelle (white F5E5DE, black A65C17, assian F4ECAD, mulatto E3AB72, aztec CF9D7C);
- *plural* per identificare la presenza di pluralità;
- *nocolor* per ottenere un pittogramma senza l'uso dei colori.

I pittogrammi di Arasaac sono associati solo alla forma base delle parole, per cui per utilizzare le Api abbiamo necessariamente bisogno di uno strumento di lemmatization, come quello offerto da Stanza. Per effettuare una traduzione automatica in simboli, con la migliore rappresentazione semantica della frase possibile, abbiamo considerato due ulteriori scenari oltre al caso standard, ovvero nel caso in cui ad una parola è associato un simbolo.
Il primo scenario è quello in cui, Arasaac, dispone di un pittogramma multi-word, ovvero che descrive un insieme di parole invece che una singola, in modo tale fornire una miglior rappresentazione semantica della frase. Un esempio è quello del pittogramma multi-word "Campo da tennis", esso può essere infatti rappresentato tramite un singolo pittogramma 7 invece che tre 6:

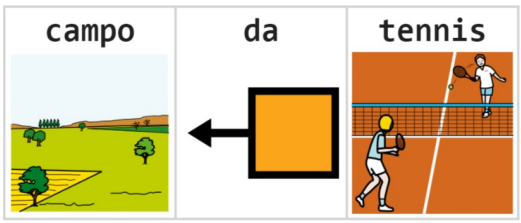

Figure 6: Rappresentazione di "Campo da tennis" tramite tre pittogrammi

Figure 7: Rappresentazione di "Campo da tennis" tramite il pittogramma multi-word "Campo da tennis"

Il seguente frammento di codice utilizza le API search di Arasaac, per controllare se, data una parola, ci sono pittogrammi multi-word che contengono la parola stessa. Tali pittogrammi, se esistono, vengono memorizzati in un array e successivamente viene verificato se, nella frase da tradurre, è presente un sotto testo corrispondente a un pittogramma memorizzato. Se l'esito della verifica è positivo, allora si memorizza l'id del pittogramma, e si incrementa l'index, che tiene conto della posizione in cui ci troviamo all'interno della frase durante la traduzione. Se invece il pittogramma multi-word non viene trovato, si passa all'utilizzo delle API best search.

```
response = requests.get("https://api.arasaac.org/api/pictograms/
    it/search/" + lemma)
    status = response.status_code
    if status == 200:
      j = response.json()
      keywords=[]
      for word2 in j:
          try:
            keyword=word2["keywords"][0]["keyword"]
            keywords.append(keyword)
          except:
            print("Something went wrong")
      if(len(keywords)>0):
        keywords=[s for s in keywords if s != lemma and s !=
    word[2] ]
          present=False
          for keyword in keywords:
```

```
            if lemma_sentence.find(keyword)!=-1:
                present=True
                words=len(keyword.split(" "))
                print(words,": ",keyword)
                global index
                index+=words
                global words_for_images
                words_for_images.append(keyword)
                response2 = requests.get("https://api.arasaac.
    org/api/pictograms/it/bestsearch/" + keyword)
                j2=response2.json()
                id = j2[0]['_id']
                result.append(id)
                index+=1
                return result
                break
```

Il secondo scenario da considerare è quello in cui, una parola non possiede nessuna rappresentaizone all'interno di Arasaac. Si entra in questo scenario quando la richiesta di API fallisce. In tal caso, si è deciso di utilizzare la libreria *WordNet* per trovare i sinonimi di una parola: tali sinonimi saranno utilizzati per inviare una nuova richiesta API ad Arasaac. In caso di successo, verrà utilizzato il pittogramma della parola sinonimo.

```
def try_synset(lemmax):
  global synsetFound
  from nltk.corpus import wordnet as wn
  synsetsList=[]
  for synset in wn.synsets(lemmax,lang=('ita')):
      for lemma in synset.lemma_names(u'ita'):
          if(lemmax!=lemma):
            synsetsList.append(lemma)
  for i in range(len(synsetsList)):
    response = requests.get("https://api.arasaac.org/api/
    pictograms/it/search/" + synsetsList[i])
    status = response.status_code
    if status == 200:
      synsetFound=True
      j = response.json()
      id = j[0]['_id']
      return id
```

Nel caso peggiore, ovvero quello in cui non esiste un pittogramma neppure per le parole sinonime, al posto del pittogramma, viene semplicemente prodotta un'immagine contenente la parola. Per creare un'immagine contenente una stringa è stata utilizzata la libreria *Python PIL*.
Il caso standard è invece quello in cui vengono impiegate le API di Arasaac *best search* per estrarre l'id del pittogramma maggior rappresentativo.
Dopo aver ottenuto gli identificativi di tutti i pittogrammi essi vengono scaricati e salvati:

```
#extract pictogram and save in a Drive
def getImg(id, plural_status,action): #parameter:id,
    plural_status[true/false], action[past, future,none]
  #print('id','id')
  if plural_status:
    #Copy a network object denoted by a URL to a local file
     #urllib.request.urlretrieve(url, filename=None, reporthook=
    None, data=None)
    urllib.request.urlretrieve('https://static.arasaac.org/
    pictograms/'+str(id)+'/'+str(id)+'_plural_300.png',
   path_CAA_pictograms+str(id)+'.png')
  elif not action == 'indef':
    print(action)
    #Copy a network object denoted by a URL to a local file
    urllib.request.urlretrieve('https://static.arasaac.org/
    pictograms/'+str(id)+'/'+str(id)+'_action-'+action+'_300.png'
    ,
   path_CAA_pictograms +str(id)+'.png')
  else:
     #Copy a network object denoted by a URL to a local file
    urllib.request.urlretrieve('https://static.arasaac.org/
    pictograms/'+str(id)+'/'+str(id)+'_300.png',
   path_CAA_pictograms+str(id)+'.png')
```

3.2 Descrizione del metodo per la misurazione delle performance

Per effettuare una misurazione delle performance come riferimento è stato impiegato un Book Corpus ottenuto dall'estrazione di un set di frasi da alcuni documenti di storie per bambini con pittogrammi Arasaac [4] [13] [1], è stata effettuata la traduzione con il modello progettato ed è stato effettuato un confronto tra i simboli risultanti ed i simboli presenti sul libro tradotto manualmente.
Di seguito il codice sviluppato relativo alla funzione di valutazione:

```
def evaluation():
  from difflib import SequenceMatcher
  df = pd.read_csv("./book.csv", delimiter=";")
  avg_score=0
  for i in range(len(df)):
    translation=translate(df["sentence"][i])
    lst=df["array_id"][i]
    lst=lst.split(",")
    for i in range(len(lst)):
      lst[i]=int(lst[i])
    sm=SequenceMatcher(None,translation,lst)
    avg_score+=sm.ratio()
  avg_score=avg_score/len(df)
```

La classe *SequenceMatcher* confronta due liste di *Integer* alla volta, che rappresentano gli ID dei pittogrammi.
La funzione *ratio()* [10] restituisce una misura della somiglianza delle sequenze come *float* nell'intervallo $[0, 1]$.
La formula corrisponde a:

$$Ratio = 2.0 * \frac{M}{T}$$

dove T equivale al numero totale di elementi in entrambe le sequenze e M equivale al numero di corrispondenze, ed il tutto

viene moltiplicato per 2.
Si è scelto di utilizzare questa metrica perchè permette di confrontare anche sequenze di lunghezza diversa.
L'analisi del modello di traduzione restituisce un ratio medio di 0.4243.

4 Discussione e conclusioni

L'obiettivo del progetto in esame consiste nella realizzazione di un traduttore automatico da frasi in linguaggio naturale a frasi in simboli C.A.A. Arasaac in lingua italiana attraverso l'impiego di strumenti di NLP, quali il POS tagging, la tokenization e la lemmatization per estrapolare le informazioni necessarie al fine di effettuare una traduzione automatica.
É stato utilizzato un Book Copus formato da solo 40 frasi e il modello proposto restituisce buoni risultati, considerando le prestazioni del modello proposto in [8] e il fatto che la valutazione ha diverse limitazioni per i motivi esposti precedentemente.
Un possibile sviluppo futuro è quello di costruire ed utilizzare un ampio dataset con frasi italiane e i corrispondenti pittogrammi aggiornati, in questo modo si potrebbe studiare in modo accurato la performance del modello e poterlo migliorare.

References

[1] Scritto e illustrato dal "Gruppo Garrulo Pappo " dell'Associazione Arca Comunità l'Arcobaleno Onlus. *Istantanee d'amore*. Quest'opera è rilasciata sotto licenza Creative Commons Attribuzione 4.0 Inter-

nazionale. Per visualizzare una copia di questa licenza, visitare il sito http://creativecommons.org/licenses/by/4.0/. 2022. URL: http://www.larchebologna.it/istantanee-damore/?fbclid=IwAR0WqkP6uR70hXOmnOcYY3nQZ2inX_kPEGFtYj9aqDUDlSMwvAsDcb7Sq9I.

[2] Centro Sovrazonale di Comunicazione Aumentativa. "Comunicatori e compositori simbolici". In: (). URL: https://sovrazonalecaa.org/documenti_condivisi/comunicatori%20e%20compositori%20simbolici-stato%20dell%20arte.pdf.

[3] Centro Sovrazonale di Comunicazione Aumentativa. "Sistemi simbolici". In: (). URL: https://sovrazonalecaa.org/documenti_condivisi/i%20sistemi%20simbolici.pdf.

[4] Testi: Maria Grazia Fiore - Simboli: Sergio Palao (ARASAAC.ORG) - Illustrazioni: Giuliano Ferri. *Il dono del Natale*. Quest'opera è rilasciata sotto licenza Creative Commons Attribuzione 4.0 Internazionale. Per visualizzare una copia di questa licenza, visitare il sito http://creativecommons.org/licenses/by/4.0/. URL: https://catechistico.chiesacattolica.it/il-dono-del-natale-in-caa/.

[5] *GitHub*. 2022. URL: https://github.com/NicandroP/AAC-Natural-Language-Processing.

[6] *Google Colaboratory*. 2022. URL: https://research.google.com/colaboratory/faq.html.

[7] *NLTK - Natural Language Toolkit*. 2022. URL: https://www.nltk.org/.

[8] Magali Norré et al. "Extending a Text-to-Pictograph System to French and to Arasaac". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. Held Online: INCOMA Ltd., Sept. 2021, pp. 1050–1059. URL: https://aclanthology.org/2021.ranlp-1.118.

[9] Sergio Palao. *ARASAAC - Centro Aragonese di Comunicazione Aumentativa e Alternativa*. Quest'opera è rilasciata sotto licenza Creative Commons Attribuzione 4.0 Internazionale. Per visualizzare una copia di questa licenza, visitare il sito http://creativecommons.org/licenses/by/4.0/. 2022. URL: https://arasaac.org/.

[10] *Ration function in difflib library*. 2022. URL: https://docs.python.org/3/library/difflib.html?highlight=ratio#difflib.SequenceMatcher.ratio.

[11] *Spacy - Industrial-Strength Natural Language Processing*. 2022. URL: https://spacy.io/models/it.

[12] *Stanza - A Python NLP Package for Many Human Languages*. 2022. URL: https://stanfordnlp.github.io/stanza/.

[13] Amaya Padilla Collado - Traduttore = Anna Giraudo Testi:José Manuel Marcos Rodrigo. *LIBRO DEL BEBE'-LIBRO DEI BAMBINI-LIBRO DEL CANE*. Quest'opera è rilasciata sotto licenza Creative Commons Attribuzione 4.0 Internazionale. Per visualizzare una copia di questa licenza, visitare il sito http://creativecommons.org/licenses/by/4.0/. 2022. URL: https://arasaac.org/materials/it/66.

[14] *Tint - The Italian NLP Tool*. 2022. URL: https://github.com/dhfbk/tint.

[15] VINCENT VANDEGHINSTE, INEKE SCHUURMAN LEEN SEVENS, and FRANK VAN EYNDE. "Translating text into pictographs". In: *Natural Language Engineering* 23.2 (2017), pp. 217–244. DOI: 10.1017/S135132491500039X.