Università di Torino

Tecnologie e Linguaggio Naturale - Parte I

ChatBot Danny

Luca Bonamico 945815
Federico di Geronimo 945865

31 maggio 2024

Indice

1 Introduzione

In questo documento viene spiegata l'implementazione del dialogue system *Prof. Danny*. Il ruolo del chatbot è interrogare uno studente sugli argomenti della prima parte del corso di TLN. Le domande sono state estratte casualmente dal corpus fornito dal docente e si trovano nel file *question_pool.py*.
Di seguito spiegheremo le componenti principali che permettono il funzionamento dell'applicazione.

1.1 Librerie utilizzate

Per questo progetto abbiamo utilizzato le seguenti librerie:

- *spaCy* utilizzato per fare l'analisi a dipendenze delle risposte dell'utente. L'utilizzo di questa libreria nel nostro progetto viene spiegato in maniera più approfondita nel capitolo 4.
- *simpleNLG* utilizzato per la generazione delle domande e delle risposte da parte del *Prof. Danny*. L'utilizzo di questa libreria nel nostro progetto viene spiegato in maniera più approfondita nel capitolo 3.

2 Dialogue managment

dialog_manager.py è il componente che gestisce l'interazione con lo studente.
L'esame è composto da un numero di domande principali variabile e da alcune domande "secondarie" che vengono poste nel caso in cui le risposte fornite alle domande principali siano parziali o errate.
Per ogni domanda le fasi dell'interazione sono:

- scelta della domanda "principale": viene scelta in maniera causale una delle domande del corpus;
- generazione della domanda: la domanda viene effettivamente generata (come spiegato nel capitolo 3.1 e stampata a schermo;
- lettura e memorizzazione della risposta dell'utente;
- analisi della risposta (come spiegato nel capitolo 4);
- generazione del commento del professore e valutazione della risposta;
- eventuale scelta di una domanda "secondaria" nel caso di risposta parziale o errata per permettere allo studente di correggersi o aggiungere qualcosa.

Alla fine dell'esame viene comunicata la valutazione complessiva.
Per quanto riguarda la gestione delle domande "principali/secondarie" si è proceduto in questo modo:

```
1  while self.num_questions < 4:
2      question = self.generate_question()
3      answer = self.clean_answer(input("Student: "))
4      # analisi risposta dello studente
5      if (not question_complete):
6          self.print_danny(f"Do you want to add something else about\
7                           {self.dependency_parser.get_topic()}?")
8          answer = self.clean_answer(input("Student: "))
9          # analisi risposta dello studente
10         self.num_questions += 0.5
11     # calcolo del punteggio parziale
12     self.num_questions += 1
13 # calcolo e stampa del punteggio finale
```

Listing 1: Struttura semplificata della funzione principale di *dialog_manager.py*

Come si può notare nel Listing 1, la variabile *self.num_questions* è stata utilizzata per tener conto di quante domande sono state fatte e per capire quando terminare l'interrogazione. La variabile viene incrementata di 1 quando Prof. Danny pone allo studente una domanda "principale" (che viene selezionata tramite il metodo *generate_question()* a riga 2) e di 0.5 nel caso in cui viene posta una domanda "secondaria". Quando il valore di *self.num_question* supera 4, il ciclo termina e il chatbot stampa il risultato dell'esame allo studente.

2.1 Calcolo del voto finale

Ogni domanda ha un peso diverso sul voto finale in base al suo tipo (spiegazioni dei tipi delle domande al capitolo 5):

- peso 2 per le domande aperte;
- peso 1 per gli altri tipi.

Dopo ogni risposta dello studente il punteggio parziale è così calcolato:

$$((32 * keyword\_inserite)/numero\_keyword\_totali) * peso\_domanda$$

Come è stato già spiegato, nel caso in cui lo studente non risponde in maniera completamente corretta alla domanda "principale", il chatbot ne propone una "secondaria". In questa particolare circostanza il punteggio parziale sarà moltiplicato per un coefficiente (0.85) in modo da dare una penalizzazione.
Alla fine dell'esame il voto totale sarà calcolato sfruttando i punteggi parziali nel seguente modo:

$$\sum punteggi\_parziali / \sum pesi\_domande$$

3 Generazione

Come scritto nel capitolo di introduzione, per generare il testo abbiamo utilizzato la versione inglese della libreria *simpleNLG*. Questo tool è stato progettato per generare frasi in linguaggio naturale.
Nel nostro progetto ci è servita per tre scopi principali:

- generare le domande per interrogare lo studente sugli argomenti della prima parte del corso;
- generare i commenti del professore relativi alle risposte date dallo studente;
- generare il commento finale che indica il risultato dell'esame allo studente.

3.1 Generazione domande

Una volta che la domanda è stata selezionata (come spiegato nel capitolo 2) il *dialog_manager.py* richiama la funzione *generate_question(...)* di *generator.py* opportuna. I parametri della funzione sono:

- *keywords*: una lista che contiene parole chiave tutte corrette oppure tutte false;
- *topic*: è una stringa che si riferisce all'argomento della domanda;
- *type*: indica il tipo della domanda che è stata selezionata.

In base al *type*, il metodo genera domande con due strutture differenti:

- Is the **keyword** **part of** the **topic**?
- Is the **topic keyword**?

In entrambi i casi la **keyword** è scelta casualmente dalla lista che viene passata come parametro. Il componente **part of** è scelto casualmente in una lista statica di complementi partitivi presente nel file *sentence_pool.py*.
Il Listing 2 mostra come vengono generate le frasi del tipo 1, mentre il Listing 3 mostra come vengono generate le domande di secondo tipo.

```
keyword_phrase = self.nlgFactory.createNounPhrase(random.choice(keywords))
topic_phrase = self.nlgFactory.createNounPhrase("the", topic)
p = self.nlgFactory.createPrepositionPhrase(random.choice(inclusion_terms))
p.addComplement(topic_phrase)
s2 = self.nlgFactory.createClause(keyword_phrase, "be")
s2.addPostModifier(p)
s2.setFeature(simplenlg.Feature.INTERROGATIVE_TYPE,
              simplenlg.InterrogativeType.YES_NO)
```

Listing 2: Is the **keyword** **part of** the **topic**?

```
keyword_phrase = self.nlgFactory.createNounPhrase(random.choice(keywords))
topic_phrase = self.nlgFactory.createNounPhrase("the", topic)
s2 = self.nlgFactory.createClause(topic_phrase, "be")
s2.addPostModifier(keyword_phrase)
s2.setFeature(simplenlg.Feature.INTERROGATIVE_TYPE,
              simplenlg.InterrogativeType.YES_NO)
```

Listing 3: Is the **topic** **keyword**?

3.2 Generazione commenti

Una volta che la risposta è stata analizzata (spiegato nel capitolo 4), il *dialog_manager.py* richiama la funzione *generate_answer(...)* di *generator.py*. Quindi in base al tipo di domanda e alla correttezza della risposta vengono generati dei commenti differenti.
La struttura generale dei commenti è la seguente: "Comment. Teacher evaluation.". Il *Comment* viene scelto casualmente da una lista di commenti statica, mentre la *Teacher evaluation* viene generata dinamicamente con soggetto, verbo e oggetto e degli object_modifier.
In base al grado di correttezza delle risposte fornite dallo studente vengono selezionate le parti della frase corrette in modo da generare un commento appropriato.
Per aumentare la variabilità dei commenti, abbiamo usato probabilità differenti nella seguente modalità:

- con il 30% di probabilità vengono formati commenti con la struttura "Teacher evaluation. Comment.";
- con il 30% di probabilità vengono formati commenti con la struttura "Comment. Teacher evaluation."
- con il 40% di probabilità vengono formati commenti con la struttura "Comment." se il commento che dobbiamo generare è positivo. Nel caso in cui dobbiamo generare un commento negativo allora la struttura sarà "Teacher evaluation. Comment.".

3.3 Generazione commento finale

La struttura del commento finale è "Comment. Result.", dove la frase *Result* riporta il voto in 30 esimi se la valutazione è positiva altrimenti viene comunicata la bocciatura. Il *Comment* viene scelto casualmente in base alla positività del voto.
Anche in questo caso per aumentare la variabilità dei commenti nel 50% dei casi la struttura "Comment. Result." viene invertita.

4 Analisi della risposta

Per analizzare le risposte dello studente abbiamo sfruttato le funzionalità offerte dalla libreria *spaCy*. Abbiamo utilizzato spaCy per l'estrazione delle keywords dalle risposte fornite dallo studente e per il riconoscimento del *topic* della domanda.

4.1 Riconoscimento delle keywords

L'approccio utilizzato per il riconoscimento delle keywords è stato quello di prendere in considerazione i *noun chunks*, ovvero quei gruppi di parole che insieme fungono da nome nella frase (tipicamente sono composti da un nome e dai suoi modificatori), e le *entities*, cioè parole che rappresentano concetti ben definiti come persone, organizzazioni, ecc...
Una volta ottenuti i *tokens* della risposta (ovvero noun chunks e entities), si procede a confrontarli con la lista delle keywords corrette della domanda posta. Ogni domanda ha un **frame** di riferimento che ha il compito di controllare se lo studente ha risposto correttamente alla domanda. Per ogni token identificato nella risposta, il frame controlla se è una keyword (ovvero una parola corretta) e la salva. Nel momento in cui il frame è completo, cioè ha riconosciuto tutte le keywords necessarie, lo studente avrà risposto correttamente alla domanda.

4.2 Identificazione del topic

Come spiegato nel paragrafo 3.1, il topic della domanda selezionata casualmente è utile per generare le domande e i commenti.
Per identificare il topic sfruttiamo le relazioni tra le parole che compongono una frase, cioè le *syntactic dependencies*. In particolare inseriamo tutti i soggetti (le parole con dipendenza *"nsubj"*) in un vettore e poi recuperiamo il token (tra i tokens della domanda) che fa parte di questo vettore.

```
def get_topic(self):
    subjects = []
    sentence = next(self.doc.sents)
    topic = ""
    for word in sentence:
        if word.dep_ == "nsubj":
            subjects.append(word)
    for token in self.get_tokens():
        for subject in subjects:
            if str(subject) in token:
                if len(token) > len(topic):
                    topic = token
    if topic == "":
        return None
    return topic
```

Listing 4: Funzione per ricavare l'argomento della domanda

5 Question

Ogni domanda è formata da 4 elementi:

- *text*: è una stringa che contiene il testo della domanda così come è nel corpus;

- *keywords*: è una lista che contiene tutte le keywords che lo studente deve inserire per rispondere in modo corretto alla domanda;
- *false_keywords*: è una lista che contiene tutte le "false" keywords che possono essere proposte nelle domande a trabocchetto (solo per domande di tipo 1 e 3);
- *type*: rappresenta la tipologia della domanda. Abbiamo pensato a 5 tipologie. I primi 3 types vengono utilizzati per classificare le domande prese direttamente dal corpus:
  - 1: si riferisce a domande "binarie".
    Ad esempio: *"HMM model is generative or discriminative?"*
  - 2: si riferisce a domande aperte.
  - 3: si riferisce a domande che si aspettano una lista di elementi come risposta. Ad esempio: *"What are the NLG symbolic phases?"*

  I types rimanenti vengono assegnati alle domande generate dinamicamente (come spiegato nel paragrafo 3.1):
  - 4: domande vero/falso. Ad esempio: *"Is the HMM model generative?"*
  - 5: domande vero/falso a trabocchetto. Ad esempio: *"Is the HMM model partial?"*

6 Valutazione

In questo capitolo abbiamo esaminato 3 dialoghi tra il chatbot e lo studente e abbiamo descritto gli errori più comuni che abbiamo notato durante le interazioni. Alla fine abbiamo inserito la Trindi tick-list per una valutazione finale del sistema di dialogo.

6.1 Analisi dei dialoghi

Di seguito sono presentati 3 esempi di interazione.
In questa prima interrogazione lo studente risponde correttamente a tutte le domande, inserendo tutte le *keywords* necessarie per prendere il voto massimo.

```
Prof. Danny: Welcome to TLN exam. I will ask you some questions
             about the first part of TLN course. Let's start!
             Is linguistic realization one of the nlg symbolic phases?
Student: Yes, it is.
Prof. Danny: You know the NLG symbolic phases.
             Is the natural languages unrestricted grammar?
Student: No, they aren't
Prof. Danny: You know the topic of the natural languages.
             What is POS Tagging?
Student: POS tagging is a way to remove ambiguity from words.
         It works assigning for each word of a sentence a
         Part of Speech tag representing its grammatical role.
Prof. Danny: You get the subject of the POS Tagging. Well done.
             Is politeness an element in the 6 remarkable phenomena?
Student: No, politeness isn't in that list.
Prof. Danny: You know the topic of the 6 remarkable phenomena. Well done.
Prof. Danny: Congrats, you finished the exam! Wait for the result...
             Your evaluation is 30L. You've shown great dedication
             to your studies, and it's evident in your results.
```

Listing 5: Primo esempio di interazione con l'utente: esame completamente corretto.

In questo secondo esempio, l'utente risponde correttamente solo ad alcune domande. Si può notare come il bot risponde negativamente quando si accorge che una risposta è sbagliata e quindi non contiene le *keywords* stabilite.

```
Prof. Danny: Welcome to TLN exam. I will ask you some questions
             about the first part of TLN course. Let's start!
             Is tone part of the 6 remarkable phenomena?
Student: Yes, it is.
Prof. Danny: You do not get the subject of 6 remarkable phenomena.
             Incorrect response.
             What is POS Tagging?
Student: POS taggin is a way to remove grammatical ambiguity.
Prof. Danny: You know the concept of POS Tagging. Excellent response.
             What is the PARSEVAL measure?
Student: Perseval measure is a way to evaluate parser
         through precision and recall.
Prof. Danny: You presented a partial answer.
             Do you want to add something else about PARSEVAL measure?
Student: No, I don't remember anything else.
Prof. Danny: That's not it. You do not know
             the topic of PARSEVAL measure.
             What is a Probabilistic CFG?
Student: It's a CFG with a probability associated to each
         of its productions.
Prof. Danny: You know the subject of Probabilistic CFG. Brilliant.
Prof. Danny: Congrats, you finished the exam! Wait for the result...
             Your grade is 23. Your understanding is developing,
             keep up the good work.
```

Listing 6: Secondo esempio di interazione con l'utente: esame discreto.

Nell'ultimo esempio abbiamo simulato uno studente che non ha studiato per l'esame, non riuscendo ad accumulare abbastanza punti per superare l'interrogazione.

```
Prof. Danny: Welcome to TLN exam. I will ask you some questions
             about the first part of TLN course. Let's start!
             What are the 6 remarkable phenomena of the human dialogue?
Student: The 6 remarkable phenomena are: turns,
         speech acts and grounding.
Prof. Danny: You specified a few of the 6 remarkable phenomena.
             Do you want to add something else
             about 6 remarkable phenomena?
Student: No, I don't remember the others.
Prof. Danny: You described none of the 6 remarkable phenomena.
             According to Chomsky's hierarchy,
             do natural languages fall under 'context free grammar'
             or 'mildly context sensitive'?
Student: Natural languages fall under context free grammar.
Prof. Danny: You do not understand the concept of natural languages.
             Try again.
             HMM model is generative or discriminative?
Student: HMM model is discriminative.
Prof. Danny: That's not it. You do not know the subject of HMM model.
             Is sentence planning one of the nlg symbolic phases?
Student: Yes, it is
Prof. Danny: You know the topic of NLG symbolic phases.
Prof. Danny: Congrats, you finished the exam! Wait for the result...
             There were some gaps in your understanding of the subject.
             You failed the exam.
```

Listing 7: Terzo esempio di interazione con l'utente: studente bocciato.

6.2 Problemi riscontrati

I problemi principali incontrati durante la valutazione delle interazioni sono stati:

- *parole chiave troppo generali*: in quanto non abbiamo usato tecniche per controlli più sofisticati sulle parole chiave, abbiamo dovuto inserire delle keywords generali in modo da essere sicuri che una risposta corretta le contenesse.
  Ad esempio un utente potrebbe rispondere utilizzando *"grammatical categories"*, mentre un altro *"assign to each word a grammatical category"*. In questo caso *"category"* e *"categories"* non sarebbero andate bene come parole chiave. Una possibile soluzione a questo problema potrebbe essere utilizzare un meccanismo di stemming per confrontare le keyword in base alla loro radice.

- *generazione di feedback ripetuti*: può capitare che il bot risponda più volte commentando in modo simile una risposta dell'utente. Questo capita perché il corpus utilizzato per la generazione dei feedback è limitato, quindi, anche se il bot genera casualmente le frasi, potrebbe proporre commenti già utilizzati.
  Per evitare queste ripetizioni, si potrebbe implementare una memoria che tiene

conto dei commenti già generati dal bot, in modo che ne produca sempre di differenti.

- *problema generazione domande*: il nostro bot, oltre a stampare a video domande direttamente prese dal corpus, è in grado di creare delle domande a trabocchetto utilizzando delle *false keywords*. Per fare ciò abbiamo bisogno dell'argomento (topic) della domanda originale. Quest'ultimo viene ricavato utilizzando *spaCy*, in particolare facendo un'analisi delle dipendenze e andando a recuperare il token *"nsubj"*. Il problema nasce nel momento in cui il soggetto della domanda originale non coincide con l'argomento della domanda e vengono generate come quella seguente: *"Do you want to add something else about 6 remarkable phenomena?"*. Si nota infatti come il soggetto ricavato, "6 remarkable phenomena" non è il vero topic della domanda: *"What are the 6 remarkable phenomena of the human dialogue?"*.

- *problemi grammaticali*: sempre riguardo la generazione delle domande, ci siamo accorti che *simpleNLG* ogni tanto genera delle frasi grammaticalmente non completamente corrette. Ad esempio:
  "Is the natural languages unrestricted grammar?" (come si può notare dal listing 5 che rappresenta il primo esempio di interazione)
  In questo particolare esempio "Is" è ovviamente sbagliato in quanto la frase andrebbe formulata con "Are". Quello illustrato è solo un esempio, durante varie prove abbiamo notato altri piccoli errori grammaticali simili.

- *problema generazione commenti nelle domande di follow-up*: se il bot propone una domanda di follow-up (perché non si ha dato una risposta completa alla domanda "principale") e, nella risposta, non si aggiunge nulla di nuovo (ovvero non si inseriscono nuove keywords) il bot genera un commento completamente negativo anche se alla domanda principale è stata inserita almeno una keyword corretta.

6.3 Trindi Tick-list

Abbiamo usato 3 simboli per indicare quali comportamenti assume il bot tra quelli "desiderati":

- ✗ indica che quel comportamento non viene adottato;
- ✓ indica che quel comportamento viene adottato;
- ~ indica che il comportamento viene adottato dal bot ma in maniera non ottimale.

Di seguito sono riportate le domande della Trindi Tick-list relative ai comportamenti di un sistema di dialogo:

- Qn1 - L'interpretazione dell'enunciato è sensibile al contesto? (Is utterance interpretation sensitive to context?) ✗
- Qn2 - Il sistema è in grado di gestire risposte a domande che forniscono più informazioni di quelle richieste? (Can the system deal with answers to questions that give more information than was requested?) ✓

- Qn3 - Il sistema è in grado di gestire risposte a domande che forniscono informazioni diverse da quelle effettivamente richieste? (Can the system deal with answers to questions that give different information than was actually requested?) ✗
- Qn4 - Il sistema è in grado di gestire risposte a domande che forniscono meno informazioni di quelle effettivamente richieste? (Can the system deal with answers to questions that give less information than was actually requested?) ✓
- Qn5 - Il sistema è in grado di gestire designatori ambigui? (Can the system deal with ambiguous designators?) ✗
- Qn6 - Il sistema può gestire informazioni specificate negativamente? (Can the system deal with negatively specified information?) ~
- Qn7 - Il sistema può gestire l'assenza di risposta a una domanda? (Can the system deal with no answer to a question at all?) ✗
- Qn8 - Il sistema è in grado di gestire input "noisy"? (Can the system deal with noisy input?) ✗
- Qn9 - Il sistema è in grado di gestire i sotto-dialoghi di "aiuto" avviati dall'utente? (Can the system deal with 'help' sub-dialogues initiated by the user?) ✗
- Qn10 - Il sistema è in grado di gestire i sotto-dialoghi "non di aiuto" avviati dall'utente? (Can the system deal with 'non-help' sub-dialogues initiated by the user?) ✗
- Qn11 - Il sistema pone solo domande di follow-up appropriate? (Does the system only ask appropriate follow-up questions?) ✓
- Qn12 - Il sistema è in grado di gestire informazioni inconsistenti? (Can the system deal with inconsistent information?) ✗
- Qn13 - Il sistema può gestire la revisione delle convinzioni? (Can the system deal with belief revision?) ~
- Qn14 - Il sistema può gestire interruzioni da parte dell'utente? (Can the system deal with barge-in input?) ✗
- Qn15 - È possibile ottenere un tutorial sul sistema riguardante il tipo di informazioni che può fornire e quali sono i vincoli sull'input? (Is it possible to get a system tutorial concerning the kind of information the system can provide and what the constraints on input are?) ✗
- Qn16: Il sistema verifica la sua comprensione delle espressioni dell'utente? (Does the system check its understanding of the user's utterances?) ✗