ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO DI INFORMATICA - SCIENZA E INGEGNERIA

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

TESI DI LAUREA TRIENNALE

IN

INTELLIGENZA ARTIFICIALE

Sviluppo di un chatbot in linguaggio Python con l'integrazione di Argumentation Mining

Candidato:
Federico Spurio

Matricola **0000825241**

Relatore:
Prof. Ing. Paolo Torroni

Correlatore:
Dott. Federico Ruggeri

Indice

Sommario

Introduzione

L'elaborazione del linguaggio naturale (NLP – Natural Language Processing) ha assunto sempre più importanza nella ricerca scientifica, insieme all'introduzione di nuovi strumenti di *machine learning*. In particolare, si è distinta la branca dell'*argumentation mining*, capace di individuare argomenti a partire da un documento non strutturato e scritto da un essere umano. Poiché questa branca è relativamente recente, non esiste ancora nessuno strumento di facile utilizzo per analizzare argomenti e trovarne di simili. Pertanto, l'obiettivo di questa tesi è la creazione di un chatbot argomentativo, che riconosca gli argomenti in un testo in input di un utente e restituisca argomenti simili, ovvero attinenti allo stesso *topic* che sostengano o contraddicano l'idea dell'utilizzatore. Questo chatbot può avere applicazioni in diversi ambiti, poiché l'*argumentation mining* è una branca multidisciplinare. Un interessante utilizzo potrebbe essere nel campo della letteratura scientifica, per poter trovare documenti attinenti ad un dato argomento tramite il riconoscimento di esso; gli strumenti esistenti si basano soltanto sul riconoscimento di parole chiave.

Ma non solo nell'ambito scientifico, questo chatbot trova applicazioni anche nell'ambito del Web, e in particolare dei *social media*, nelle scienze sociali, nell'ambito legale e nella medicina, dove potrebbe essere un valido strumento per collegare sintomi e malattie. Ancora, nell'ambito dei dibattiti esiste già uno strumento in grado di sostenere dibattiti con esseri umani riguardo a *topic* complessi: si tratta dell'intelligenza artificiale IBM *Debater*, sviluppata, appunto, da IBM.

Per il raggiungimento dell'obiettivo preposto, ovvero fornire agli utenti degli argomenti a supporto o in contrasto con una loro tesi, ho prodotto due versioni del chatbot; entrambe fanno uso dell'*argumentation mining* per il riconoscimento degli argomenti in input, ma la prima versione (chatbot "statico") è un sistema idem potente, ovvero per un dato input l'output è sempre lo stesso. La seconda versione (chatbot "dinamico"), invece, utilizza il *reinforcement learning* con il

fine di addestrarsi e produrre un output sempre più attinente alla soggettività dell'utilizzatore.

Dal punto di vista della struttura, questo elaborato è suddiviso in due capitoli principali: il primo verte sull'introduzione e sul funzionamento dell'*argumentation mining* in generale, dove vengono illustrati anche i campi di interesse e applicazione; il secondo riguarda la parte di progettazione e implementazione di un chatbot argomentativo, soffermandosi sullo strumento per il riconoscimento di argomenti MARGOT e sulla tecnica di *reinforcement learning* dei *k-armed bandits*.

Capitolo 1: Argumentation Mining (AM)

1.1 Introduzione

L'***Argumentation mining*** è una branca dell'elaborazione del linguaggio naturale (NLP – Natural Language Processing), che mira ad estrarre automaticamente argomentazioni strutturate da documenti con testo non strutturato.

L'argumentation, in generale, è un'area di ricerca multidisciplinare, che studia dibattiti e ragionamenti e collega aree diverse tra loro come la logica, la filosofia, psicologia e l'ingegneria informatica.

Nell'ambito della ricerca scientifica, questa branca sta assumendo sempre più importanza e attenzioni, per via del suo potenziale nel processamento innovativo di informazioni derivanti dal Web, in particolare dai social media. Inoltre, recenti sviluppi nel *machine learning* promettono delle applicazioni innovative in campi quali scienze sociali ed economiche, l'elaborazione di nuove politiche e le tecnologie di informazione.

In questo elaborato verranno analizzati i principi fondanti dell'argumentation mining, verranno studiati i modelli dell'argomentazioni presenti in letteratura e gli ambienti di applicazione.

La struttura e i contenuti di questo capitolo si basano sul lavoro di Lippi e Torroni [1].

1.2 Argomentazione

L'argomentazione è un campo di ricerca multidisciplinare, che si occupa di studiare il dibattito e i processi dietro al ragionamento, e collega tra loro aree diverse come la logica e la filosofia, il linguaggio, la retorica e il diritto, la psicologia e l'informatica. Il concetto di argomentazione ha assunto un ruolo centrale nell'ambito dell'intelligenza artificiale, grazie alla sua capacità di coniugare esigenze rappresentative con modelli cognitivi relativi all'utente e modelli computazionali per il ragionamento automatizzato.

In particolare, lo studio dell'argomentazione, nel campo dell'intelligenza artificiale, ha dato origine a nuova disciplina chiamata *computational argumentation*. Il tema dell'argomentazione sta guadagnando importanza non solo nell'informatica, ma anche in alcuni ambiti delle scienze cognitive, dove recenti studi sembrano indicare che il funzionamento del ragionamento umano stesso sia argomentativo. Inoltre, anche nell'ambito delle scienze sociali computazionali sono stati recentemente proposti modelli di simulazione basati su agenti; i principi fondamentali di questi modelli fanno riferimento in maniera esplicita alle teorie relative al concetto di argomentazione.

Un'importante fonte di dati, per molte delle discipline interessate in questi studi, è il Web e in particolare i *social media*. Giornali online, recensioni di prodotti, blog e altro ancora, forniscono un flusso in continua crescita di informazioni eterogenee tra loro, nelle quali è possibile trovare, isolare ed analizzare argomentazioni. Ma i *social media* non sono l'unica fonte di dati; infatti viene data importanza anche a contesti specifici, come il contesto legale e i siti di dibattito. L'interesse per il settore di ricerca in questione non è volto solamente ad un fattore di sfida in termini di successo scientifico, ma rappresenta uno strumento applicativo dal grande potenziale. Questo è dimostrato dall'interesse di alcuni grandi aziende, come ***IBM*** *Research,* che ha finanziato un progetto di computazione cognitiva dal valore di milioni.

Ricapitolando, la disponibilità di questi dati, insieme agli enormi progressi nella *computational linguistics* e nel *machine learning*, hanno creato un terreno fertile per la nascita e la crescita di una nuova area di ricerca chiamata *argumentation* (o *argument*) *mining.*

1.3 Definizione

L'obbiettivo principale dell'*argumentation mining* è l'estrazione automatica di argomentazioni da corpora testuali generici e non strutturati, al fine di fornire dati organizzati per modelli computazionali in grado di catturare, a partire dai testi presi in esame e per mezzo di particolari rappresentazioni, ragionamenti e argomentazioni.

Un esempio concreto di quanto descritto è osservabile in Figura 1, dove viene mostrata l'estrazione automatica di argomenti a partire da documenti testuali grazie all'utilizzo di *argumentation mining*.

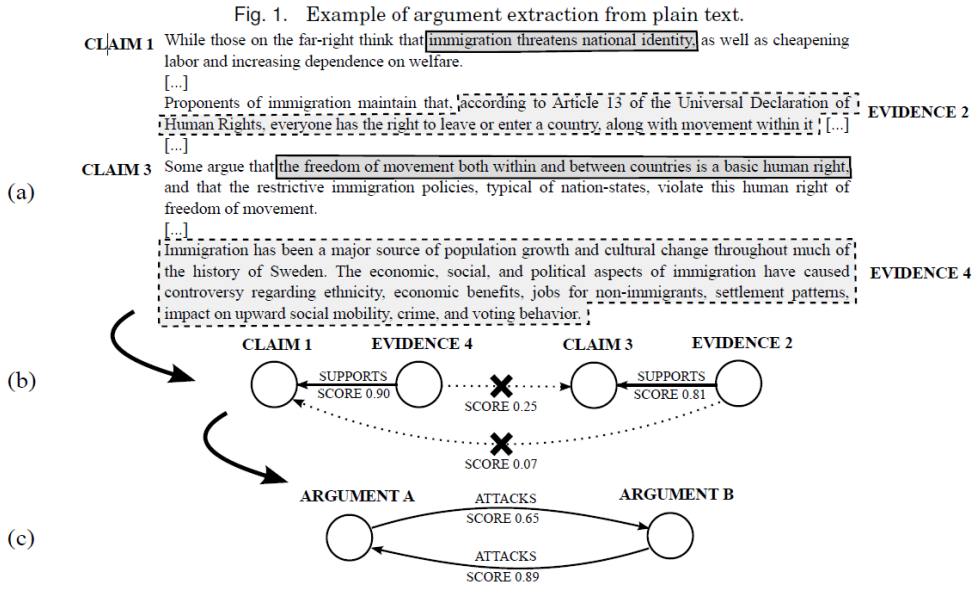

Figura 1) Esempio di estrazione di argomenti da un testo non strutturato

Per prima cosa, vengono individuate le frasi riconosciute come argomentative ed estratte dal documento dato in input. In seguito, le parti che compongono l'argomentazione, in questo caso dai *claim* (affermazioni) e dalle *evidence* (prove a supporto dell'affermazione) vengono individuate all'interno delle frasi riconosciute (figura 1(a)). Successivamente, vengono ipotizzati dei collegamenti tra gli argomenti estratti (figura 1(b)) e infine vengono inferite le connessioni tra i singoli componenti all'interno di ciascun argomento (figura 1(c)), producendo così una struttura relazionale facilmente espandibile (per esempio un grafo).

1.4 Possibili ambiti di applicazione

Prima di soffermarsi nel dettaglio sulla struttura interna e sui punti di ricerca aperti riguardanti l'*argumentation mining*, è opportuno evidenziarne il potenziale applicativo, che, come già detto, può essere espresso non solo nel campo dell'intelligenza artificiale e dell'informatica, ma anche nelle scienze sociali e umanistiche.

L'introduzione di strumenti di *argumentation mining* apporterebbe tangibili miglioramenti a diversi ambiti di studio e di ricerca, come per esempio nell'apprendimento di politiche o la formulazione di decisioni, dove i modelli e le scelte strategiche di ausilio sarebbero migliorate dall'utilizzo di argomenti estratti automaticamente. Non solo, l'utilizzo di *argumentation mining* potrebbe portare benefici all'interno di processi di ingegnerizzazione con obbiettivo la valutazione automatica di soluzioni di design alternative. Ancora, un sistema di *argumentation mining* porterebbe ad un incremento massiccio all'analisi qualitativa dei commenti postati sui social network e allo stesso modo per quanto riguarda gli articoli dei giornali specializzati, fornendo strumenti senza precedenti ai *policy-maker* e ai ricercatori negli ambiti delle scienze sociali e politiche, oltre a creare nuovi scenari per il marketing e le imprese. Pertanto, l'*argumentation mining* potrebbe portare allo sviluppo di nuove modalità di organizzazione, supporto e visione dei dibattiti online, utilizzando, per esempio,

il *clustering* dei post degli utenti e fornendo nuovi filtri che si basano sulla forza delle argomentazioni e sui post dei partiti coinvolti. Anche per quanto riguarda il mondo del Web l'*argumentation mining* porterebbe sostanziali miglioramenti. Infatti, potrebbe agire come tecnologia di supporto per la *Argument Web vision*, aiutando nell'interpretazione di argomenti presenti nelle pagine web, al fine di abilitare la navigazione tra molteplici risorse e siti web, tramite una struttura di collegamento tra argomenti.

Infine, una delle più grandi sfide dell'intelligenza artificiale è sempre stata riuscire ad emulare il livello di comprensione e di ragionamento dell'essere umano, ma deve tenere conto dei costi di costruzione e mantenimento di conoscenza strutturata in domini aperti a partire da dati non organizzati. Pertanto, l'*argumentation mining* potrebbe contribuire in maniera significativa allo sviluppo scientifico in questa direzione. Nello specifico, potrebbe gettare le basi per una nuova generazione di sistemi di intelligenza artificiale, capaci di combinare *framework* statici e di inferenza logica, in grado quindi di estrarre argomenti a partire da testi non strutturati e scritti da utenti umani e, seguendo un procedimento logico, produrre nuovi argomenti, a partire da quella conoscenza.

1.5 Modelli argomentativi

La disciplina dell'argomentazione ha origine antiche, e si trovava sottoforma di branca della conoscenza nella dialettica e nella filosofia, ed era dedicata allo studio e all'analisi di come dichiarazioni e asserzioni venivano proposti e discussi e come i conflitti tra opinioni divergenti venivano risolti. Data questa tradizione di lunga data, l'argomentazione, nel corso dei secoli, ha permeato diverse aree di conoscenza oltre la filosofia, come il linguaggio e la comunicazione, la logica, la retorica, la legge e l'informatica. Non dovrebbe quindi sorprendere che anche la letteratura è ricca di modelli di rappresentazione degli argomenti. Uno dei modelli più noti è dovuto a Toulmin, che propose che la microstruttura logica dell'argomentazione umana e il ragionamento, sono composte da sei categorie:

- **Datum:** incontrovertibile, costituisce la base per avanzare un *claim* (soggettiva, probabilmente controversa)
- **Claim:** affermazione soggettiva e spesso ambito di discussione
- **Warrant:** regola di inferenza che collega il *datum* con il *claim*
- **Qualifier:** elemento di supporto che rappresenta il grado di confidenza di un dato *claim*
- **Rebuttal:** elemento che definisce le condizioni per delineare il *claim*
- **Backing:** elemento che fornisce giustificazione al *warrant*

Nonostante il modello di Tolmin sia stato largamente utilizzato, nella pratica, applicazioni diverse richiedono diverse strutture argomentative, e l'utilizzo dell'adattamento rappresentativo del modello di Tolmin in campi come politica, legge, design, scienze, filosofia e contenuti generati dagli utenti, è ancora materia di discussione.

I modelli dell'argomentazione si sono diffusi anche nell'area dell'intelligenza artificiale, specialmente in relazione alla rappresentazione della conoscenza, al ragionamento non monotico e alla ricerca di sistemi multi-agente, dando origine ad un nuovo campo chiamato *computational argumentation.* Sono stati quindi sviluppati diversi modelli, secondo tre categorie:

- **Rethorical:** viene posta enfasi sulla capacità di persuasione dell'interlocutore nell'attirare l'attenzione di un pubblico di ascoltatori
- **Dialogical:** descrivono il modo in cui gli argomenti sono connessi tra loro tramite strutture dialogiche
- **Monological:** l'attenzione è posta sulla struttura dell'argomento stesso (per esempio sulle connessioni tra differenti componenti di uno specifico argomento considerato)

Per esempio, la figura 1(a) e la 1(b), seguono un modello *monological*, dove i componenti dell'argomento sono *claim* ed *evidence*, collegate tra loro. La figura 1(c) mette in relazione due argomenti con un terzo, seguendo un modello *dialogical,* basato su una relazione di attacco.

Un'altra classificazione ben nota nel *computational argumentation* è la dicotomia tra argomentazione astratta e argomentazione strutturata. La prima considera ogni argomento come entità atomica, sprovvista di struttura interna. Rappresenta quindi un modello *dialogical*, che permette una modellazione e un'analisi delle relazioni di attacco tra argomenti, come mostrato in figura 1(c), o su una serie di argomenti, che possono, o meno, essere giustificati da alcune semantiche. L'argomentazione strutturata, invece, propone una struttura interna per ogni argomento, che potrebbe essere quella adottata in figura 1(a) e 1(b). La definizione di una struttura risulta cruciale quando l'obbiettivo è l'estrazione di porzioni di argomenti derivanti dal linguaggio naturale. Pertanto, l'*argumentation mining* utilizza, prevalentemente, i modelli di argomentazione strutturata. Il modello proposto rientra nella categoria *monological*, ma può essere ricondotto alla categoria *dialogical.*

Poiché ci sono molte proposte significative riguardo l'argomentazione strutturata, è impossibile dare una singola definizione formale universalmente accettata di argomento strutturato. Una definizione intuitiva sull'argomento è stata proposta da Walton, descrivendo quest'ultimo come una serie di dichiarazioni articolate in tre parti:

- Una o più premesse
- Una conclusione
- Un'inferenza lega le premesse alla conclusione

Nella letteratura, le conclusioni sono talvolta indicate come *claim*, le premesse sono spesso chiamate *evidence* (o *datum* secondo il modello di Toulmin), mentre l'inferenza viene identificato nell'argomento stesso (*warrant*).

Il concetto di *argumentation* è stato storicamente riferito al processo di costruzione di argomenti e, fino all'avvento del *computational argumentation*, al processo di determinazione di un insieme di conclusioni inerenti ad un insieme di argomenti dati. Tuttavia, i termini *argumentation mining* e *argument mining*

sono spesso utilizzati in modo intercambiabile e in senso lato, poiché il campo di ricerca conserva ancora un forte elemento di esplorazione concettuale.

Il processo atto a rilevare le premesse e la conclusione di un dato argomento, così come si trova nel testo di un discorso, viene generalmente indicato come *identification* o *extraction*, mentre compiti più specifici sono il rilevamento dei *claim* (*claim detection*) e delle *evidence* (*evidence detection*). Altri processi sono l'*attribution*, che riguarda l'attribuzione della paternità di un argomento, *completion*, il quale obiettivo è di inferire componenti impliciti di un argomento, come l'entimema e il tacito presupposto relativo al ragionamento del buon senso, ed infine il *relation prediction*, che mira a identificare relazioni tra argomenti o all'interno di uno stesso argomento.

Essendo l'*argumentation mining* un dominio di ricerca giovane, sia la sua definizione sia i suoi approcci ed obbiettivi variano ampiamente. Alcune ricerche puntano ad estrarre argomenti da generici documenti non strutturati, step fondamentale per le applicazioni pratiche, mentre altre ricerche partono da un dato insieme di argomenti e si focalizzano su aspetti quali l'identificazione di relazioni di attacco/supporto che sussistono tra loro.

1.6 Struttura di un sistema di *argumentation mining*

Qualsiasi sistema di *argumentation mining* richiede di gestire una grande quantità di problematiche ad esso strettamente correlate, al fine estrarre informazioni e nozioni inerenti alla dimensione argomentativa dell'espressione umana, a partire dall'elaborazione di testi non strutturati. I sistemi sviluppati fino ad ora rispecchiano l'architettura rappresentata da *pipeline* mostrata in Figura 2. All'interno di questa architettura, è possibile individuare tre fasi principali: *argumentative sentence detection, argument component boundary detection* ed infine *argument structure prediction*.

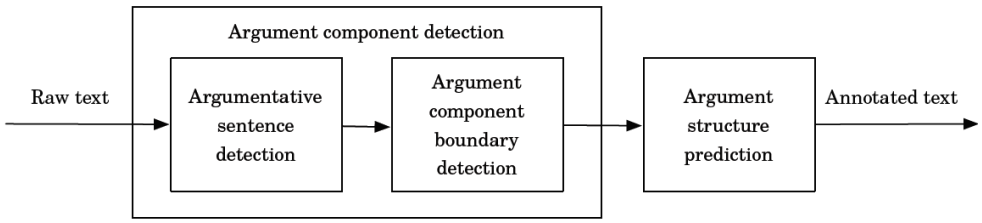

Figura 2) Architettura a pipeline di un sistema di argumentation mining

Tuttavia, prima di descrivere nel dettaglio le fasi appena elencate, è necessario analizzare i molteplici fattori che le interessano, al fine di evidenziarne problematiche e punti aperti di studio. In particolare, le difficoltà che si riscontrano in un sistema di *argumentation mining* possono essere riassunte secondo cinque dimensioni ortogonali:

- **Granularità dell'input:** indica il livello di dettaglio all'interno del quale argomenti o i loro componenti vengono ricercati. Alcuni approcci considerano porzioni di testo a livello di paragrafo. D'altro canto, la maggior parte della ricerca attuale si concentra sulle singole frasi, nonostante alcuni autori suggeriscano una ricerca dei confini sintattici dei componenti ad un livello di dettaglio maggiore
- **Tipologia dell'input:** definisce il tipo dei dati in input, come ambito legale, discussioni online, news, saggi, ecc. Fino ad ora, gli approcci esistenti si sono concentrati principalmente su di una singola tipologia di dati. Ogni tipologia ha le proprie specifiche peculiarità. Questa caratteristica è di particolare importanza in determinati scenari, dove è necessario considerare le informazioni specifiche legate al contesto
- **Modello di definizione dell'argomento:** ogni sistema di *argumentation mining* si riferisce ad un modello di argomento specifico. Finora, la maggioranza dei sistemi implementati utilizza un modello *claim* e *premise*
- **Granularità del *target* di interesse:** il *target* del processo di *mining*, ovvero di estrazione, varia anche in termini di granularità. Alcuni lavori

hanno come *target* specifici componenti di un argomento, come per esempio il *claim*; altri lavori hanno come *target* l'intero argomento

- **Obiettivo dell'analisi:** varia in base ad uno spettro di possibilità che possono essere utilizzate per classificare approcci esistenti. Gli obiettivi più comuni sono: rilevazione (*detection*), classificazione (*classification*), previsione delle relazioni (*relation prediction*), attribuzione (*attribution*) e completamento (*complation*)

1.7 Argument component detection

L'obiettivo più comune del primo stadio di un sistema di *argumentation mining* è rilevare argomenti (o componenti di un argomento, a seconda della granularità del *target* desiderata) all'interno di un testo dato in input. Le entità recuperate rappresentano quindi i nodi (o parte di essi) del grafico finale dell'argomento. Nella maggior parte dei sistemi esistenti, questo problema viene risolto mediante la divisione in due distinti sotto-problemi: l'estrazione di frasi argomentative e l'individuazione dei confini dei componenti. Non tutti i sistemi però seguono necessariamente questa pipeline a due fasi: alcuni sistemi partono dal presupposto che i confini dell'argomento siano stati precedentemente rilevati tramite altri mezzi, limitando quindi il loro obiettivo al singolo task di classificazione.

1.7.1 Argumentative sentece detection

Durante il primo step per rilevare i componenti degli argomenti solitamente viene affrontato il compito di estrarre, dal documento di input, quelle frasi che contengono un argomento (o parte di esso). Il problema può essere facilmente formulato come compito di classificazione (*classification*), che potrebbe essere risolto, in linea di principio, da qualsiasi classificatore che utilizza *machine learning*. Eppure, anche per un compito così semplice, molte soluzioni diverse possono essere concepite, anche a seconda del modello di argomento utilizzato e

sull'obiettivo ultimo del sistema di *argumentation mining*. In generale esistono tre opzioni:

1. Un classificatore binario viene addestrato per distinguere frasi argomentative da quelle non-argomentative. Il task di identificazione del tipo di argomento (*claim* o *premise*) viene delegato ad un secondo stadio
2. Un classificatore multi-classe viene addestrato per distinguere tutti i componenti dell'argomento, secondo il modello di argomento adottato. Ciò presuppone che una frase possa contenere al massimo un componente di un argomento
3. Un set di classificatori binari viene addestrato, uno per ogni componente di argomento esistente nel modello considerato. In questo modo, una stessa frase può contenere più di un componente dell'argomento

Indipendentemente dall'opzione scelta, bisogna selezionare un tipo di classificatore, oltre alle tecniche (*feature*) da utilizzare. I sistemi esistenti hanno utilizzato, fino ad ora, un'ampia varietà di classici algoritmi di *machine learning*. I classificatori vengono addestrati in ambiente supervisionato, facendo quindi sempre riferimento ad un insieme di esempi etichettati. Per ogni esempio, viene data una rappresentazione del testo da classificare (i.e. sottoforma di un vettore di funzione), insieme alla relativa classe (etichetta). La fase di addestramento produce un modello che può quindi essere utilizzato per eseguire predizioni su del nuovo testo mai analizzato prima.

Sebbene, nella letteratura, si abbia cercato di confrontare alcuni di questi approcci, non ci sono prove evidenti per poter affermare quale classificatore dovrebbe essere preferito.

Anche per la scelta delle *feature*, molti lavori condividono diverse analogie, in quanto impiegano tecniche classiche per la rappresentazione del testo. Tra le tecniche più utilizzate, una scelta molto comune, seppur ingenua, ricade sull'utilizzo della rappresentazione a *Bag-of-Words* (*BoW*), secondo la quale una frase *F* viene rappresentata da un vettore di valori binari, avente come lunghezza

il numero di parole contenute nel dizionario di termini preso come riferimento. Un elemento *i-esimo* del vettore ha valore uno se la parola *i-esima* del dizionario è presente nella frase *F*. Questo modello è stato studiato ed esteso, per esempio, utilizzando la variante *TF-IDF*, che rappresenta anche la frequenza di una parola in una frase (TF – Term Frequency) e la rarità della parola nel vocabolario (IDF – Inverse Document Frequency). Un'ulteriore estensione possibile consiste nel considerare anche *bag-of-bigrams* o *trigrams*, che consiste nel costruire dizionari aggiuntivi composti da tutte le possibili coppie o terzine di termini.
Nonostante la sua popolarità, l'approccio *BoW* presenta due limiti importanti:

1. L'ordine delle parole nella frase viene ignorato
2. La somiglianza semantica tra i termini non viene presa in considerazione

Per questo, tecniche più avanzate sono state sviluppate, per sopperire a queste limitazioni. Per esempio, può essere utile incorporare conoscenza derivante da ontologie, tesauri e database lessicali.
Un'altra categoria di tecniche adatte a questi classificatori si basa sulle informazioni grammaticali, in modo da indicare la categoria grammaticale (verbo, sostantivo, aggettivo, ecc.) a cui appartiene un determinato termine di una frase. Frequentemente si utilizzano anche tecniche basate sulla punteggiatura e sui tempi verbali.
Si potrebbe rivelare cruciale la scelta di utilizzare, nei classificatori, informazioni di contesto e non solo quelle ottenibili dal documento. Questa tecnica trova applicazione nell'ambito legale, oppure per i dibattiti, dove potrebbe essere utile specificare il *topic* della discussione. Quanto descritto viene riferito con i termini *context-dependent claim detection* (CDCD) e *context-dependent evidence detection* (CDED).

1.7.2 Argument component boundary detection

L'obiettivo del secondo stadio della *pipeline* è il rilevamento dei limiti esatti di ciascun componente dell'argomento, anche conosciuto come *argumentative*

discourse unit o *argument unit*. Il problema alla base di questo rilevamento è la delimitazione dell'inizio e della fine di un componente, per ogni frase considerata argomentativa, poiché l'intera frase potrebbe non corrispondere esattamente ad un singolo componente dell'argomento. Dal punto di vista della granularità dell'input, occorre considerare tre casi non mutuamente esclusivi:

1. Una parte della frase (possibilmente l'intera frase) corrisponde ad un componente di un argomento
2. Due o più componenti di un argomento possono essere presenti all'interno della stessa frase
3. Un componente può spaziare su più frasi (come per esempio l'EVIDENCE 4 nell'esempio di Figura (1)

La maggior parte dei metodi esistenti presuppone solo una delle possibilità sopra illustrate. Chiaramente, il problema del rilevamento dei limiti dipende fortemente dal modello di argomento adottato, poiché diversi tipi di componenti di un argomento possono avere caratteristiche specifiche. Alcuni lavori addirittura ignorano il problema di delimitare i confini dei componenti.

1.8 Argument structure prediction

La fase finale della *pipeline* è sicuramente la più complessa, in quanto mira a prevedere i collegamenti tra i vari argomenti (o loro componenti), a seconda della granularità del *target*. Questa fase richiede di comprendere le connessioni e le relazioni tra gli argomenti rilevati, necessitando quindi di una rappresentazione di alto livello e di una conoscenza dei processi cognitivi. Il punto cruciale del problema è solitamente ricondotto alla fase di *prediction* (predizione) piuttosto che alla fase di *detection* (rilevazione), poiché l'attenzione non è solo su una parte specifica, ma piuttosto sulle connessioni tra le porzioni del documento di input. L'output di questo stage è un grafo che collega gli argomenti riconosciuti (o parte di essi). Più precisamente, i collegamenti nel grafo possono rappresentare diverse tipologie di legame, come l'implicazione logica, il supporto o il conflitto. Per

quanto riguarda il mondo dei social media e dei documenti Web, tale grafo rappresenterebbe uno strumento di inestimabile valore. Le applicazioni di questo strumento sarebbero molteplici: analisi delle dinamiche dei dibattiti sociali sul Web e studiare la diffusione di un'influenza nei social network. Inoltre, l'impatto su altre discipline, come scienze sociali, linguistica computazionale e argomentazione formale sarebbe drammatica.

1.9 Corpora

Qualsiasi approccio nell'ambito di *argumentation mining*, per mezzo di tecniche quali il *machine learning* e l'intelligenza artificiale, richiede una raccolta di documenti annotati (*corpus*), da utilizzare come *training set* per qualsiasi tipo di predittore. La costruzione di corpora annotati è, generalmente, complesso e dispendioso in termini di tempo e di risorse richieste, come, per esempio, l'impiego un gruppo di esperti, al fine di ottenere annotazioni omogenee e coerenti. Ciò è dovuto al dominio di studio, in quanto l'identificazione dei componenti di un argomento, dei loro confini precisi e delle relazioni che sussistono tra loro, possono essere attività complicate (e controverse) persino per gli esseri umani.

1.10 Scenari di interesse

Una forte limitazione delle ricerche riguardanti l'*argumentation mining* è data divisione netta degli ambiti di applicazione. In particolar modo, è possibile notare che i corpora definiti negli anni, fanno riferimento ad un unico dominio applicativo. Si possono quindi delineare i seguenti scenari di interesse:

- **Dominio legale:** questo è stato il dominio applicativo pioneristico per l'*argumentation mining* e più di successo. Questo studio rappresenta, fino ad ora, uno dei pochi sistemi che mira a implementare una *pipeline* di *argumentation mining* completa, sebbene altamente specializzata per un solo genere

- **Medicina e biologia:** lo sviluppo di set annotati riguardanti la medicina e la biologia è una nuova tendenza che sta attirando crescente attenzione. Potrebbe rivelarsi estremamente importante per costruire una base di conoscenza atta a legare tra loro sintomi e malattie, oppure geni e malattie, o ancora per aiutare nella prescrizione di medicine
- **Scienze sociali:** saggi retorici, filosofici e persuasivi costituiscono un campo interessante per l'*argumentation mining*. Gli argomenti trattati sono molto eterogenei. A causa della natura dei dati, solo poche frasi nel corpus sono non-argomentative
- **Contenuti web:** i social media e il Web in generale, offrono una molteplicità di documenti eterogenei contenenti un numero incalcolabile di argomenti. Nonostante i risultati ottenuti in questo campo siano ancora pochi, questo dominio apre la strada a numerosi ed interessanti ambiti di applicazione. In questo senso, l'*argumentation mining* potrebbe essere il punto chiave per una nuova tecnologia capace di estrapolare nuova conoscenza da un'immensità di contenuti disorganizzati e non strutturati

Capitolo 2: Implementazione del chatbot

2.1 Obiettivi

Questa tesi nasce con l'intento di sfruttare l'*argumentation mining* per studiare un caso di applicazione reale. Infatti, l'obiettivo di questo lavoro è costruire un chatbot capace di interagire con gli utenti, al fine di proporre argomenti simili ad un dato argomento in input. L'idea di base è riuscire a fornire all'utente degli argomenti a supporto di una sua tesi o in contrasto con essa.

Un caso di applicazione reale sarebbe l'ambito dei dibattiti, ambiente già altamente studiato da IBM. Infatti, IBM ha sviluppato un'intelligenza artificiale, chiamata *IBM Debater* [4], già in grado di, appunto, fronteggiare un dibattito con esseri umani riguardo a *topic* complessi. Pertanto, per l'implementazione del chatbot, ho sfruttato i documenti messi a disposizione da IBM per costruire l'argomento in output da fornire all'utente. In particolare, IBM mette a disposizione un file Excel suddiviso in colonne (tra le più importanti *topic*, *evidence* e *claim* a cui si riferiscono le *evidence*), dal quale ho estratto numerose *evidence* a tema "violenza e videogiochi".

2.2 Strumenti e ambienti utilizzati

Per la costruzione del chatbot, ho scelto il linguaggio di programmazione Python, per la sua potenza espressiva, portabilità e semplicità d'uso, ma soprattutto per l'integrazione con librerie fondamentali, che semplificano la programmazione. Infatti, la libreria *numpy* si è dimostrata fondamentale per questo progetto, in quanto fornisce numerosi metodi per lavorare con array, matrici e dizionari, che gestiscono automaticamente operazioni su di esse.

Per quanto riguarda il *core* dell'applicazione, ovvero il riconoscimento effettivo di argomenti (o componenti di essi) all'interno dell'input dell'utente, viene utilizzata una versione *portable* di MARGOT, per gentile concessione di Marco Lippi e Paolo Torroni. MARGOT è essenziale per il riconoscimento degli

argomenti, punto cardine di questo chatbot argomentativo. La versione *portable* si basa sullo stesso modello della versione web (descritta nei paragrafi successivi), ma cambia l'output e l'ambiente di esecuzione. Infatti, questa versione di MARGOT si basa su librerie Java ed eseguibile tramite comandi bash. La forte integrazione di bash con sistemi operativi basati su GNU/Linux, ha indirizzato la mia decisione di scrivere il chatbot per sistemi Linux. L'output (un esempio è riportato in Figura 3) della versione *portable* di MARGOT è salvato su file di diverse estensioni (txt, xml, JSON).

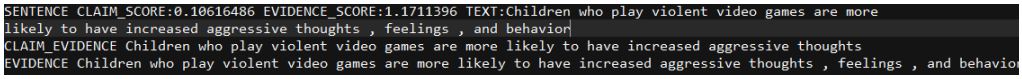

Figura 3) Esempio di file di output txt della versione portable di MARGOT

2.3 Mining ARGument frOm Text (MARGOT)

MARGOT (Mining ARGument frOm Text) è uno strumento specifico di *argument detection*, ideato ed introdotto da Lippi e Torroni [1]. Questo strumento permette di individuare *claim* ed *evidence* sulla base di informazioni non strettamente legate dal contesto specifico.

2.3.1 Definizione e struttura

MARGOT è un sistema web che nasce con l'intento di fornire uno strumento per l'estrazione di argomenti, a partire da testi non strutturati, che permetta ad un utente generico, anche non appartenente al contesto scientifico, di poter interagire con informazioni proprie dell'*argumentation mining* in maniera semplice e diretta.

MARGOT si basa sul lavoro di Lippi e Torroni [2] riguardo il *context-indipendent claim detection*, estendendolo con l'introduzione di ulteriori attività quali *context-indipendent premise detection* e *argument component boundary*

detection, basandosi su evidenze sperimentali secondo le quali molte frasi argomentative sono caratterizzate da strutture sintattiche comuni.

Il modello alla base di MARGOT è il modello argomentativo proposto da Walton. Di conseguenza, le fasi principali di esecuzione di MARGOT sono le seguenti:

1. **Individuazione di frasi argomentative** (*argumentative sentence detection* in Figura 4), ovvero che contengano almeno un componente (i.e. *claim* o *evidence*)
2. **Determinazione dei confini sintattici di ciascun componente** (*argumentative component boundaries detection* in Figura 4)

La fase 1 di questa *pipeline* ricerca componenti senza conoscere a priori il *topic* di interesse.

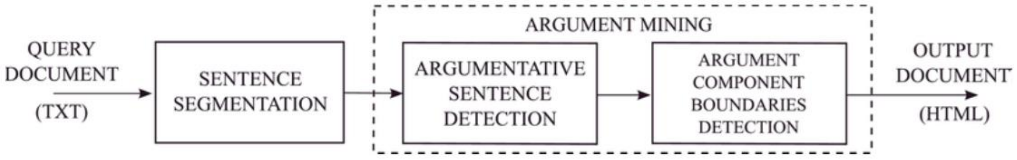

Figura 4) Modello di elaborazione di un testo tramite MARGOT

2.3.2 Funzionamento di MARGOT

MARGOT prende in input file di testo e produce come output un documento testuale contenente il risultato dell'analisi di ogni frase individuata nel testo. MARGOT, oltre a riconoscere *claim* ed *evidence*, può etichettare una determinata frase come *claim_evidence*, considerandola nella fase successiva di costruzione delle coppie sia come *claim* che come *evidence.*

Di seguito verrà descritto in dettaglio il funzionamento operazionale di MARGOT.

1. Al web server viene fornito in input un documento testuale
2. Il documento viene elaborato da parte di uno strumento di analisi (*Stanford parser*) in grado, tra le sue molteplici funzioni, di isolare le

varie frasi e di costruire, per ognuna di esse, un albero di analisi delle dipendenze sintattiche

3. Due classificatori analizzano ogni frase individuata al fine di discernere quelle contenenti *claim* o *evidence* dalle rimanenti. Per la rappresentazione dell'input, i due classificatori si avvalgono del *costituency parse tree* e dei BoW, in modo da assegnare un punteggio ad ogni frase analizzata, che indica il grado di confidenza sulla presenza o meno di un *claim* o una *evidence*
4. Le frasi vengono analizzate dal modulo relativo all'attività di *argument component boundary*, soltanto se sono state riconosciute come argomentative. Durante questa fase, le frasi sono processate secondo il *sequence labeling*, in modo tale da identificare i confini sintattici di tutti i componenti, ovvero *claim* ed *evidence*
5. I risultati sono mostrati come output all'utente sotto forma di pagina HTML (Figura 5), distinguendo graficamente i *claim* e le *evidence* facendo uso rispettivamente del grassetto e del corsivo (entrambi gli stili sono usati per i frammenti di testo identificati come *claim_evidence*)

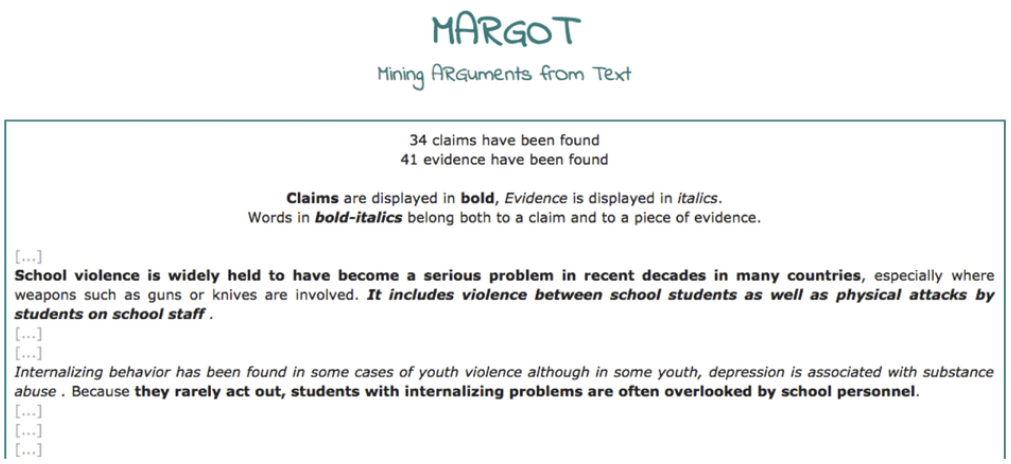

Figura 5) Output prodotto da MARGOT

2.3.3 Definizione e Framework del chatbot

Un chatbot è un software progettato con il fine di conversare con un utente umano, simulandone il comportamento. I chatbot sono molto usati per le interazioni con gli acquirenti di un sito online, per fini di marketing sui social network e quindi in tutti i contesti dove l'elevato numero di utenza non permetterebbe una rapida risposta a tutti i richiedenti. Generalmente un chatbot è costituito da due elementi fondamentali: una zona dedicata all'input dell'utente (solitamente una casella di testo) e una zona di output, dove il chatbot mostra le sue risposte.

Poiché il punto cruciale della tesi non era lo sviluppo di un interfaccia *user-friendly* con cui l'utente potesse interagire, ho deciso di utilizzare un *framework* preesistente per l'interfaccia e i comandi di base. Inoltre, per dare la sensazione di dialogo con un essere "senziente" e non una macchina, uno dei requisiti per il framework da implementare richiedeva il riconoscimento di frasi di base (saluti, ringraziamenti, poter chiedere cosa il chatbot sa effettivamente fare, …), come mostrato in Figura 6.
Il framework che ho trovato, sul quale si basa unicamente l'interfaccia utente del chatbot, si basa su *NLTK* (Natural Language ToolKit, libreria che si basa su oltre cinquanta corpora per lavorare con dati provenienti dal linguaggio umano) e *Keras*, libreria che fornisce API per minimizzare le azioni degli utenti nei casi comuni.

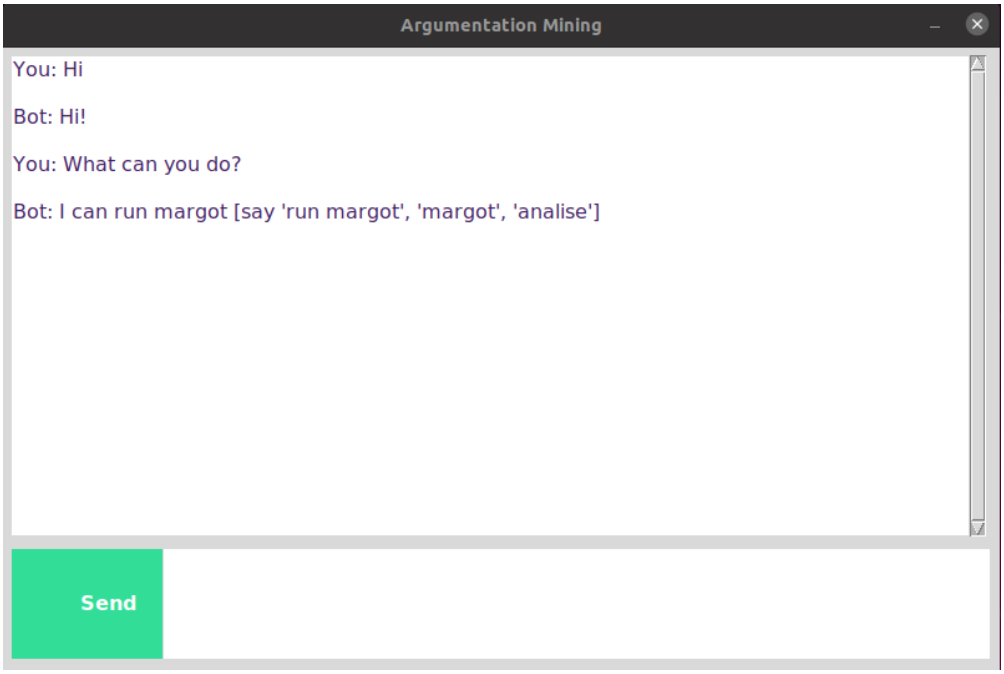

Figura 6) Esempio di interazione base con il chatbot

2.4 Prima versione del chatbot ("statico")

La prima versione del chatbot prevedeva che l'utente inserisse una frase in input all'interno di un'apposita casella e venissero prodotte in output tre frasi, scelte dal dataset fornito da IBM, con il grado di confidenza più alto stimato dalla logica del chatbot.

Il *framework* permette di essere configurato, per poter adattare il chatbot a diversi ambienti. Infatti, permette di definire tutti i contesti di conversazione desiderati nella propria applicazione. Alcuni saranno comuni a tutti gli ambiti, per esempio, per dare la sensazione all'utente di stare interagendo con un'entità "senziente", risulta necessario definire contesti come il "*greetings*", che definisce alcuni possibili *pattern* standard inseriti dall'utente (quindi "*Good morning", "Hello",* ... da cui sono poi inferiti e riconosciuti altri tipi di saluti) e le risposte da dare ("*Hi!", "How are you", ...*).

Per quanto riguarda l'applicativo vero e proprio, dopo l'avvio l'utente si troverà ad interagire con il chatbot nella finestra mostrata in Figura 6. Qui è possibile

porre domande sul funzionamento, risolvere i convenevoli per una maggiore immersione nella conversazione oppure chiedere direttamente il riconoscimento di una propria frase (inserendo la parola chiave “MARGOT”). A questo punto si aprirà una casella per l’input di testo (Figura 7), dove l’utente digiterà una frase e confermerà l’inserimento.

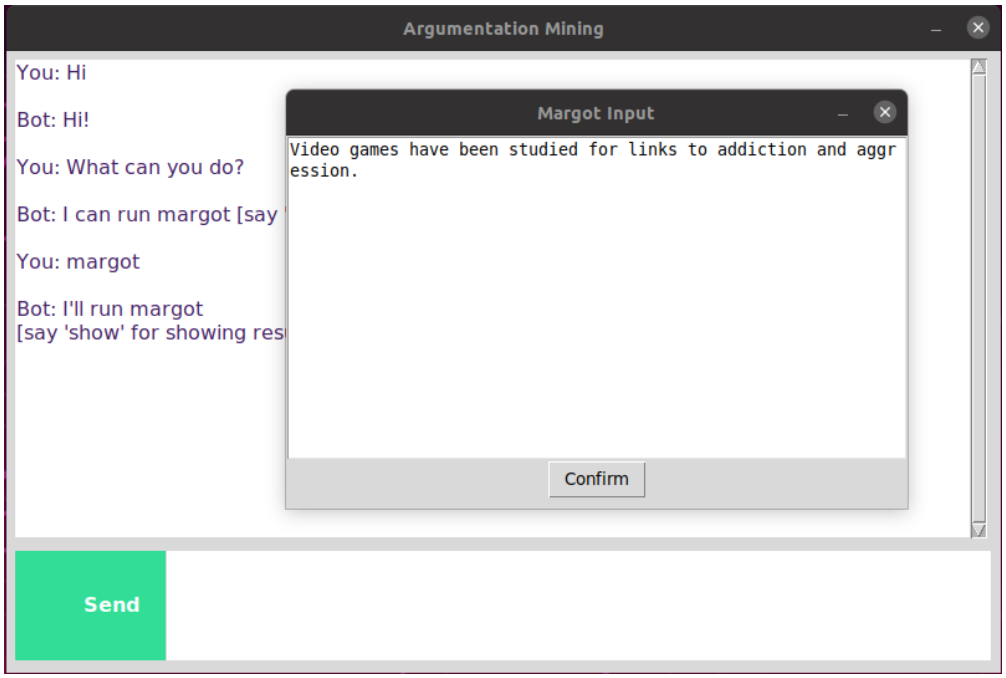

Figura 7) Casella per l’input dell’utente

MARGOT analizzerà l’input e, per semplicità, soltanto la prima *evidence* riconosciuta sarà soggetta alle fasi successive e quindi l’output finale sarà riferito soltanto ad essa. Nel caso in cui nessuna *evidence* venisse riconosciuta nell’input, un messaggio di errore viene mostrato all’utente, a cui viene richiesto di inserire un nuovo testo. Per poter comunque permettere una futura estensione di questo programma, tutti i componenti riconosciuti da MARGOT sono salvati in memoria temporanea.

A questo punto, il dataset di IBM viene caricato e salvato in memoria persistente come matrice. Il dataset rappresenta tutte le possibili risposte che possono essere date all’utente, selezionate sulla base di un criterio di similarità descritto nella

fase seguente. Il *topic* del chatbot è quindi definito dal dataset a disposizione, in questo caso, per semplicità, è stato adottato un solo *topic*, in particolare relativo alla relazione tra violenza e videogiochi. Pertanto, se l'utente inserisce un input non inerente a questo *topic*, MARGOT, come scritto prima, è capace comunque di analizzare il testo indipendentemente dal contesto, allo stesso modo la similarità riesce ad operare, ma sicuramente l'output proposto non potrà essere pertinente.

Per l'operazione di similarità ho scelto di utilizzare, in primo luogo, la similarità coseno, avvalendomi della tecnica già descritta di BoW (*Bag-of-Words*). Prima di tutto, quindi, è fondamentale definire un dizionario di parole per poter utilizzare BoW. Per questo, alla prima esecuzione, viene generato un vocabolario usando tutte le frasi contenute nel dataset di IBM, mentre alle iterazioni successive, viene importato il vocabolario così costruito ed opportunamente salvato (sfruttando la libreria *numpy*).

Successivamente, la prima *evidence* rilevata nell'input dell'utente viene trasformata in vettore binario (grazie alla libreria *sklearn.CountVectorizer*), sfruttando il dizionario precedentemente costruito. Questo vettore viene confrontato con tutte le frasi del dataset (opportunamente convertite anch'esse) tramite la similarità coseno. La similarità coseno rappresenta, appunto, il coseno dell'angolo che si forma tra due vettori A e B. In altre parole, per descrivere il nostro caso, rappresenta di quanto un vettore B si discosta dal vettore A preso come modello. La formula è la seguente:

$$similarità\ coseno = \cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||}$$

Più il valore si avvicina ad 1, più i vettori sono vicini, e quindi le frasi, da essi rappresentati, simili. Nonostante sia una tecnica semplice, con forti limitazioni, come il non considerare l'ordine delle parole, la coniugazione dei verbi e altri fattori, si è rivelata efficace e leggera per questo tipo di chatbot.

Grazie alla similarità coseno, viene calcolata, per ogni frase del dataset rispetto all'*evidence* dell'utente, la confidenza e la coppia (frase, confidenza) viene

salvata in una lista. Questa lista viene ordinata, in ordine decrescente, rispetto al valore della confidenza, in modo da avere alle prime posizioni le frasi più simili. In questo modo, all'utente verranno mostrati soltanto i risultati con almeno qualche parola in comune rispetto alla sua *evidence*. Per evitare un sovraccarico di informazioni mostrate in output, soltanto tre risultati vengono mostrati all'utente.

Infine, l'utente legge a schermo le tre frasi che il sistema ha deciso essere più simili al suo input (Figura 8) e può scegliere se continuare con l'inserimento di un nuovo testo oppure con la chiusura del programma.

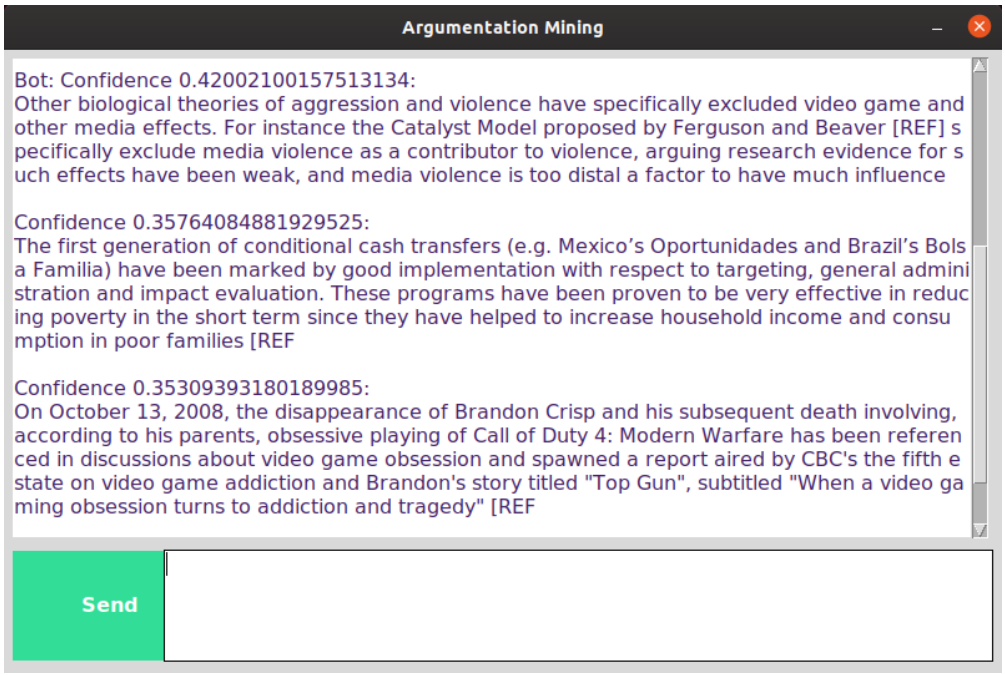

Figura 8) Output prodotto dal chatbot in relazione a un dato input

La seconda versione del chatbot, introdotta per sopperire ai limiti della prima versione, si basa principalmente sul *reinforcement learning*, e in particolare sulla tecnica chiamata *k-armed bandits*. Pertanto, è opportuno introdurre e definire questi concetti.

2.5 Reinforcement Learning

Il *reinforcement learning* è una tecnica di *machine learning* che ha come obiettivo la realizzazione di agenti autonomi in grado di decidere le azioni da intraprendere per il conseguimento di un determinato *task* sfruttando interazioni con l'ambiente in cui sono immersi.

Per ottenere ciò, il *reinforcement learning*, durante l'addestramento di un sistema, utilizza le informazioni che **valutano** le azioni intraprese dal sistema stesso, piuttosto che **fornire** direttamente le azioni corrette. Pertanto, il primo metodo è dipendente dall'azione intrapresa, mentre il secondo è totalmente indipendente. Alla base dell'apprendimento supervisionato, troviamo quindi il feedback istruttivo.

In questo capitolo considereremo uno scenario di apprendimento basilare, che non prevede di insegnare al sistema ad agire in più di una situazione. Questa impostazione viene chiamata non-associativa. I contenuti e la struttura del capitolo si basano sul libro "*Reinforcement Learning: An Introduction"* [3].

2.5.1 k-armed bandits

Il problema relativo ai *k-armed bandits* può essere formulato nel seguente modo: ad ogni step ci sono k azioni od opzioni che si possono scegliere. Dopo ogni scelta, un voto numerico viene attribuito, basandosi su una distribuzione di probabilità statica (non sempre è stazionaria, ma è comunque una buona ipotesi semplificativa) che dipende dalle azioni intraprese. L'obbiettivo è di massimizzare il voto totale previsto dopo un certo periodo di tempo.

Ogni k-esima azione ha un voto previsto corrispondente. Si può esprimere questa quantità come:

$$q_*(a) = \mathrm{E}\,[R_t | A_t = a]$$

Che semplicemente mostra che il valore $q_*(a)$ rispetto all'azione a è la valutazione media data per quell'azione.

Naturalmente, se si conoscessero a priori tutti i valori di $q_*(a)$ si potrebbe facilmente risolvere il problema selezionando il *bandit* con il più alto $q_*(a)$ per ogni fase. Tuttavia, si ha accesso soltanto alle stime di $Q_t(a)$. Pertanto, l'obiettivo è riuscire ad ottenere stime il più vicino possibile ai valori reali.
La selezione di azioni basate sulle stime correnti porta ad una situazione di *trade-off* ben conosciuta: *exploration-exploitation*. In breve, ad ogni fase si può scegliere l'azione con il più alto $Q_t(a)$: questa opzione viene chiamata *greedy action*; ma il caso base prevede la scelta randomica di un'azione, con ricompensa immediata inferiore, ma che potenzialmente potrebbe portare ad un valutazione totale maggiore.

Esistono molte soluzioni per bilanciare il *trade-off exploration-exploitation*, ma la maggior parte di esse partono da forti assunzioni e conoscenza pregressa che vengono però violate o sono impossibili da verificare in un'applicazione e nel problema completo riguardo il *reinforcement learning.*

2.5.2 Metodi di action-value

Considerando che $q_*(a)$ rappresenta la valutazione media di un'azione *a*, l'attenzione del problema si sposta sul calcolo di $Q_t(a)$. Il metodo più semplice per definirlo è come la media delle valutazioni ricevute:

$$Q_t(a) = \frac{somma\ delle\ valutazione\ riferite\ ad\ a\ prima\ di\ un\ istante\ t}{numero\ di\ volte\ in\ cui\ a\ \text{è}\ stato\ scelto\ prima\ di\ un\ istante\ t}$$

Se il numeratore è zero, si può stimare $Q_t(a)$ con un valore di default (0). Nel caso in cui il denominatore tendesse ad infinito, per la legge dei grandi numeri, $Q_t(a)$ convergerebbe a $q_*(a)$. Questo metodo viene chiamato *sample-average method*.
A questo punto, l'azione da scegliere, seguendo il metodo *greedy*, è la seguente:

$$A_t = argmax_a Q_t(a)$$

Altrimenti si può optare per la soluzione che considera l'opzione di *exploration*.

Una semplice variante dell'approccio *greedy*, che sfrutta però anche l'*exploration,* è il metodo *ε-greedy*. L'idea di base è semplice: la probabilità di selezionare un'azione random è ε, mentre la probabilità di selezionare un'azione *greedy* è 1- ε. Il vantaggio immediato di questo metodo è che all'aumentare delle fasi, le azioni che possono essere intraprese vengono esplorate tutte un numero infinito di volte, assicurandosi così che tutti i possibili $Q_t(a)$ convergeranno a $q_*(a)$. Ciò implica che la probabilità di selezionare le azioni migliori convergerà ad un valore maggiore rispetto a 1- ε, ma tutte queste affermazioni sono valide solamente asintoticamente.

Intuitivamente, il vantaggio dell'approccio *ε-greedy* è strettamente dipendente dal *task* da conseguire. Per esempio, se la varianza delle valutazioni è larga, il sistema impiegherà molto tempo per trovare l'azione migliore e quindi l'approccio *ε-greedy* dovrebbe funzionare meglio rispetto all'approccio *greedy*. D'altra parte, se la varianza è zero, allora il metodo *greedy* imparerà il valore corretto di ogni azione dopo soltanto un tentativo. In questo caso, l'approccio *greedy* potrebbe ottenere prestazioni migliori poiché sarebbe in grado di trovare l'azione corretta subito e non esplorare mai. Tuttavia, l'esplorazione può rivelarsi utile per determinati scenari. Per esempio, si supponga che la distribuzione delle valutazioni non sia stazionaria, ovvero che i valori corretti delle azioni cambino nel tempo. In questo caso, c'è necessità di esplorare per essere sicuri che le azioni non-*greedy* non siano diventate migliori rispetto alle opzioni *greedy*.

2.6 Seconda versione del chatbot (“dinamico”)

Le limitazioni della prima versione del chatbot sono evidenti, a partire da un unico *topic* disponibile fino ad arrivare all’analisi di una sola *evidence* all’interno dell’input dell’utente, che potrebbe invece aver bisogno di un analisi dettagliata di un documento corposo. Ma la limitazione a mio parere maggiore, che quindi è stata corretta con questa seconda versione, è la mancanza di un sistema di *feedback* attribuibili dall’utente. La similarità coseno screma i risultati in base alle parole uguali a quelle all’interno della frase dell’utente, ma non sempre questo porta a risultati ottimali. Per questo motivo, ho inserito la tecnica di *reinforcement learning* dei *k-armed bandits* per addestrare il chatbot a rispondere correttamente. Quindi, questa versione del chatbot ha una *pipeline* di funzionamento simile alla prima versione, ma prima di mostrare i risultati all’utente viene applicata la tecnica del *reinforcement learning*.

Con l’introduzione di questa tecnica, la similarità coseno non è più il punto cruciale nella decisione dell’output finale, ma serve come filtro iniziale per una scrematura del dataset. L’utilizzo soltanto della similarità coseno è limitante, anche perché viene utilizzato un *encoding* con BoW che semplifica troppo l’input, privandolo di proprietà semantiche, grammaticali e di contesto. Per esempio, viene perso l’ordine delle parole ed eventuali errori grammaticali rendono due parole elementi distinti nell’array.

Nonostante i problemi sopra descritti, la similarità coseno viene comunque utilizzata per una prima scrematura di dieci risultati, che vengono inseriti nella lista formata dalle coppie (frase, confidenza). Questa lista attraversa un’ulteriore filtro, dove viene applicato il metodo dei *k-armed bandits*. La scelta di utilizzare soltanto i primi dieci risultati è stata presa nell’ottica di non mostrare mai all’utente i risultati con confidenza tendente allo zero, che risulterebbero quasi mai pertinenti. Nel *reinforcement learning* è fondamentale tenere traccia delle decisioni e dei premi ricevuti nelle iterazioni passate, per questo una struttura dati

apposita viene creata e salvata in memoria persistente. Questa struttura dati è un dizionario formato da chiavi e valori corrispondenti, seguendo questo schema:

$$chiave\text{: }[array\ di\ lunghezza\ n]$$

Dove *n* è il numero di frasi contenute all'interno del dataset IBM e la chiave è la *evidence* estrapolata dall'input dell'utente. Per poter sfruttare al meglio questa tecnica di *reinforcement* e per non creare un dizionario di dimensioni troppo elevate, ho deciso di uniformare i vari input degli utenti e accorpare input simili all'interno della stessa chiave ("*Video games are too violent for children*" e "*Video games are too violent for kids"* sono considerati quindi lo stesso input). Uniformare gli input non è un punto cruciale, per questo ho deciso di riutilizzare la similarità coseno per valutarne la similitudine. L'array di ogni chiave è inizialmente popolato con soli zeri e l'indice di ogni elemento rappresenta l'indice di una frase contenuta nel dataset IBM. Ogni volta che una frase viene "votata" (il metodo di votazione è descritto in seguito), il valore dell'elemento corrispondente alla frase nel dizionario, viene incrementato di 1. Più precisamente, quando viene lasciato un *feedback* riguardo alle frasi in output, la distribuzione dei voti è:

$$\begin{cases} 1\ se\ la\ frase\ viene\ selezionata \\ 0\ altrimenti \end{cases}$$

Ricapitolando, dopo aver filtrato ed ottenuto dieci risultati dalla similarità coseno, l'*evidence* dell'utente soggetta ad analisi viene ricercata come chiave all'interno del dizionario: se è presente o è presente una chiave "molto" simile (non esiste un valore oggettivo per definire "molto", per questo ho scelto il valore arbitrario 0.7, a seguito di opportuni test) si passa allo *step* successivo, altrimenti l'*evidence* viene aggiunta come chiave al dizionario e un array viene inizializzato e associato alla nuova chiave. Il dizionario, come detto prima, è persistente e viene caricato ad ogni avvio e salvato ad ogni "votazione" da parte dell'utente.

A questo punto entra in gioco il metodo dei *k-armed bandits* ed in particolare il metodo *ε-greedy*. Pertanto, si ha bisogno di due elementi fondamentali: un valore ε e una variabile *iter* che memorizzi il numero di iterazioni già compiute, in modo

da decrementare il valore di ε opportunamente. La variabile ε ha valore iniziale 1 in modo tale da favorire l'esplorazione, secondo il modello ottimistico. Quando la variabile *iter* supera un valore *rangeNum* prestabilito (nel mio caso $\frac{100}{3}$), la variabile ε viene decrementata ad ogni iterazione, per favorire sempre di più la scelta della *greedy action*.

La classe che opera il *reinforcement learning* deve restituire tre risultati da mostrare all'utente in output; per ognuno dei risultati decide casualmente se adottare il metodo esplorativo oppure scegliere la *greedy action*. Una variabile *rand* assume ad ogni iterazione un valore randomico compreso tra 0 e 1: se *rand* è minore di ε allora si esplora, scegliendo una frase casuale tra le dieci risultanti dalla fase precedente; se *rand* invece è maggiore di ε, allora viene restituita la *greedy action*, ovvero l'elemento dell'array del dizionario per quella chiave con valore maggiore, il cui indice rappresenta una determinata frase del dataset di IBM.

I tre risultati prodotti da questa fase vengono presentati all'utente in maniera analoga alla prima versione del chatbot, ma con l'aggiunta di quattro bottoni (Figura 9) per poter dare la possibilità di votare la frase che l'utente ritiene sia più pertinente al suo input.

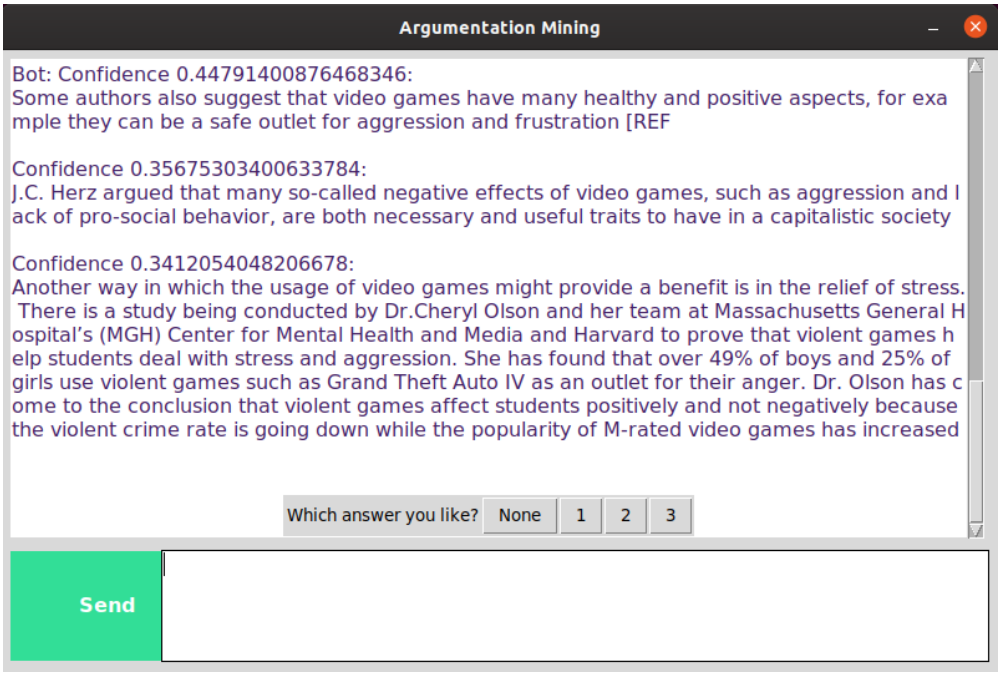

Figura 9) Sistema di votazione della risposta più attinente

Tre bottoni sono dedicati alla votazione delle frasi in output, mentre il quarto serve a non esprimere un giudizio, nel caso in cui nessuna delle proposte sia ritenuta pertinente.

2.7 Confronto tra la prima e la seconda versione del chatbot

La versione con l'aggiunta di *reinforcement learning* apporta notevoli miglioramenti rispetto alla prima versione del chatbot, nonostante sia bastato aggiungere uno *step* alla *pipeline* di funzionamento dello stesso. Mentre la prima versione del chatbot rappresentava un sistema idempotente, ovvero a parità di input l'output era sempre lo stesso, in questa seconda versione l'utente stesso ha la possibilità indiretta di modificare l'output non solo per lui, ma anche per altri utilizzatori. Mentre, quindi, nella prima versione in caso di output non coerente con l'input, l'utente non poteva in alcun modo agire sul sistema ed otteneva sempre gli stessi risultati, nella seconda versione, in caso di output non inerente, l'utente può inserire nuovamente l'input, sapendo che l'output prodotto sarà diverso. In caso l'output soddisfi le richieste dell'utilizzatore, esso può fare in

modo che lo stesso output sia prodotto in iterazioni successive, sia per la stessa frase, sia per frasi simili alla prima inserita.

In termini di prestazioni, l'introduzione del *reinforcement learning* non porta alcun *overhead* in termini di tempo di esecuzione. Però, viene introdotto un tempo di addestramento per il sistema, entro il quale non sono visibili i benefici portati dal *reinforcement*.

Le due versioni del chatbot sono state provate da uno stretto bacino di utenza e di seguito sono riportati alcuni commenti con immagini di confronto.

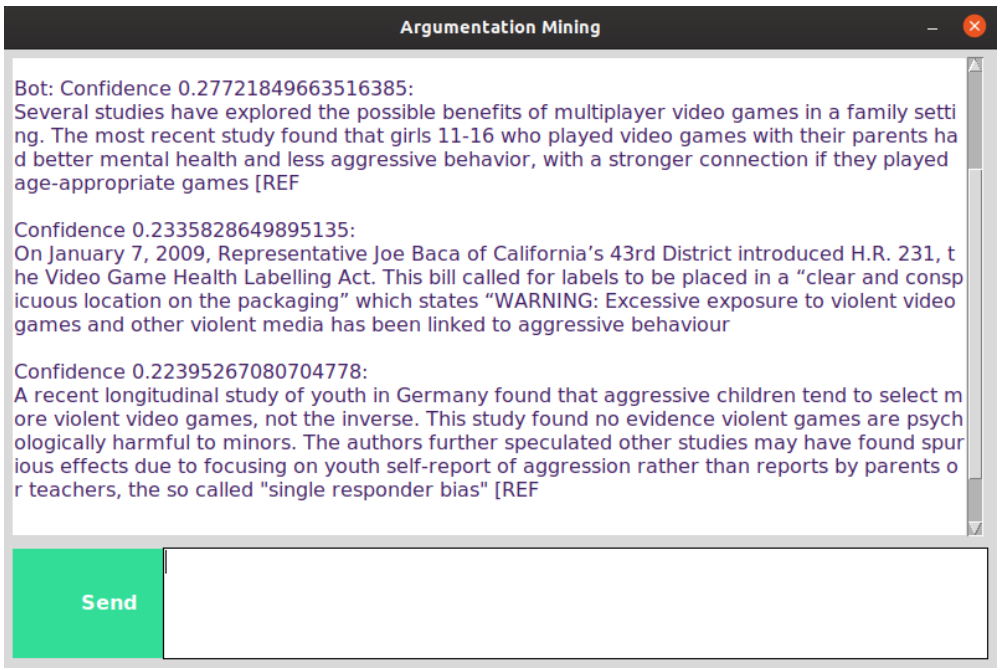

*Figura 10) Nessun risultato attinente all'input "*Video games may have bad effects on some children's health, including obesity, video-induced seizures"

Nella Figura 10 viene presentato un esempio di come, indipendentemente dalla versione del chatbot utilizzata, alcuni argomenti in input, nonostante siano attinenti al *topic*, non trovino un corrispettivo output adeguato. Ciò è dovuto alla limitata dimensione del dataset e di conseguenza del carente numero di argomenti disponibili da presentare in output.

Le seguenti immagini (Figura 11, 12 e 13) fanno riferimento allo stesso input "*Children who play violent video games are more likely to have increased aggressive thoughts, feelings, and behaviors."*. Risulta necessario sottolineare che la "correttezza" dell'output è soggettiva e può variare da utente a utente. Una metrica oggettiva per poter confrontare diversi output è difficile, se non impossibile, da definire. Si possono escludere alcuni output se il *topic* non è lo stesso rispetto a quello dell'input (per esempio in Figura 11 l'input ha come *topic* gli effetti dei videogiochi violenti sui bambini, mentre una delle risposte fa riferimento agli effetti della religione sugli individui).

La prima versione del chatbot, essendo "statico", mostra sempre l'output di Figura 11. L'utente che ha testato questa frase ritiene ci siano due argomenti attinenti al suo input e uno completamente errato: infatti il terzo argomento non è attinente al *topic* di interesse. Grazie a questa immagine risulta evidente che la similarità coseno non può essere l'unico filtro, in quanto attribuisce una confidenza abbastanza elevata (elevata in relazione alle numerose prove eseguite con il chatbot) ad un argomento fuori dal contesto. Per quanto riguarda, invece, i due argomenti rimanenti, l'utente ha indicato il secondo come più pertinente, dimostrando che una metrica oggettiva di confronto non può esistere. A sostegno di ciò, un secondo utente ha reputato più attinente il primo argomento proposto.

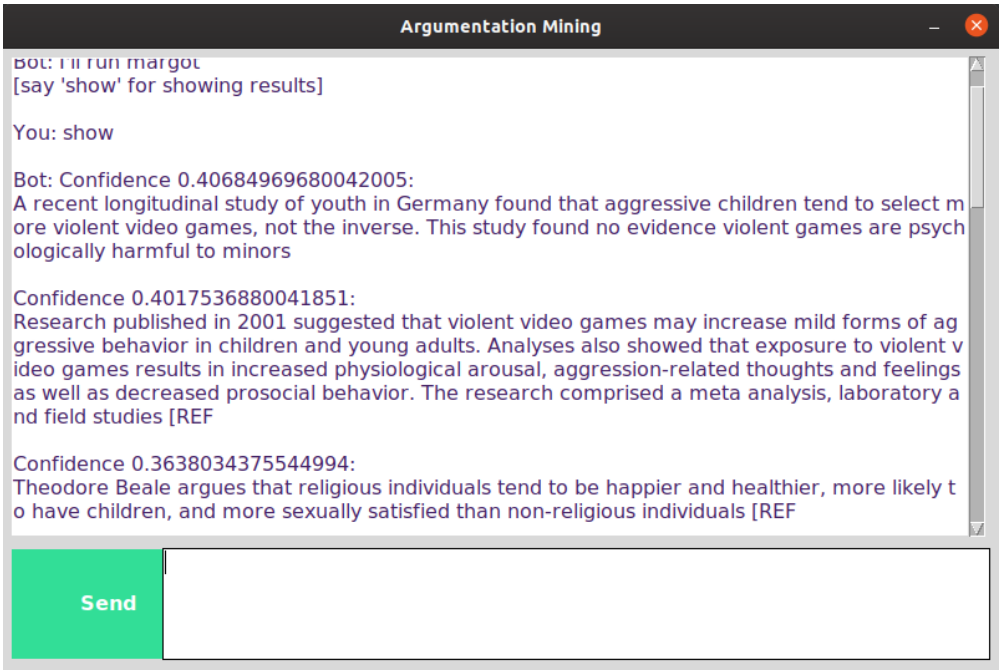

Figura 11) Output della prima versione del chatbot. La terza opzione è "sbagliata", in quanto non attinente al topic prestabilito

Lo stesso input è stato provato con la seconda versione del chatbot. Alla prima esecuzione l'output è risultato lo stesso di Figura 11. L'utente ha quindi votato la frase che riteneva più corretta in relazione al suo input (in questo caso la seconda). Dopo alcune interazioni, in cui l'output cambiava in maniera randomica, l'output proposto è stato quello di Figura 12. Il primo risultato si è dimostrato essere quello più votato dall'utente, nonostante sia ancora presente una frase non pertinente. L'utente ha quindi deciso di non votare più la prima risposta (che il sistema aveva ormai capito essere la più adatta per quell'utilizzatore), ma di scegliere la terza.

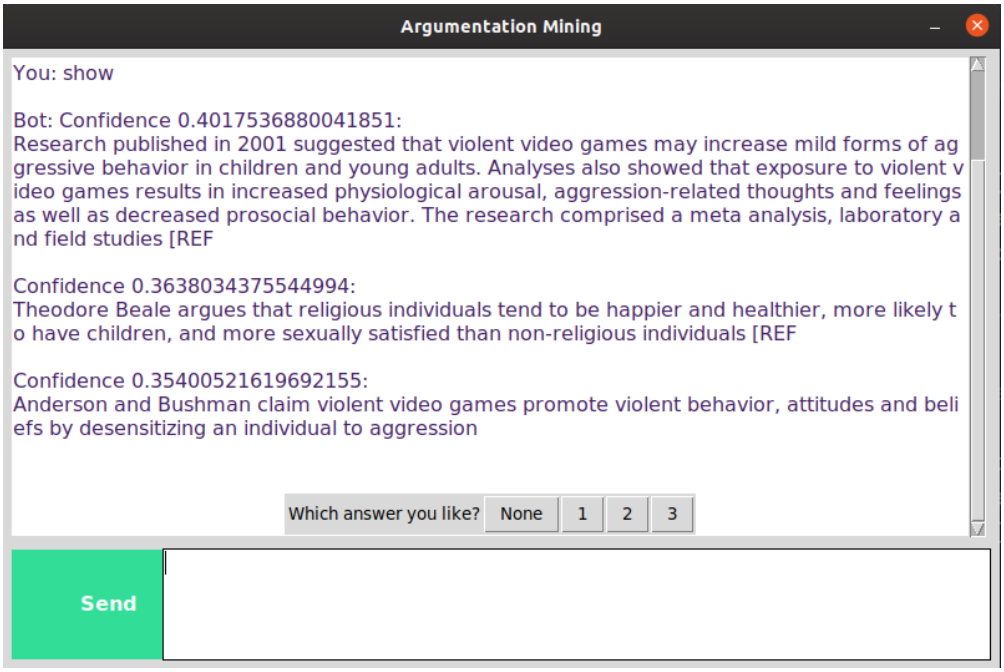

Figura 12) Output dopo alcune interazioni

Dopo trenta interazioni, il sistema esplora meno e sceglie, quasi sempre, l'azione *greedy*. Infatti, il sistema è impostato in modo che entro trenta interazioni l'ε tenda a zero. Pertanto, in Figura 13 non sorprendono i primi due risultati, in quanto sono quelli più votati dall'utente. Finalmente non è più presente l'output non pertinente riguardante la religione. Tutte e tre le argomentazioni proposte all'utente lo soddisfano, pertanto ha concluso che la versione "dinamica" del chatbot è più efficiente ed utile rispetto alla versione "statica".

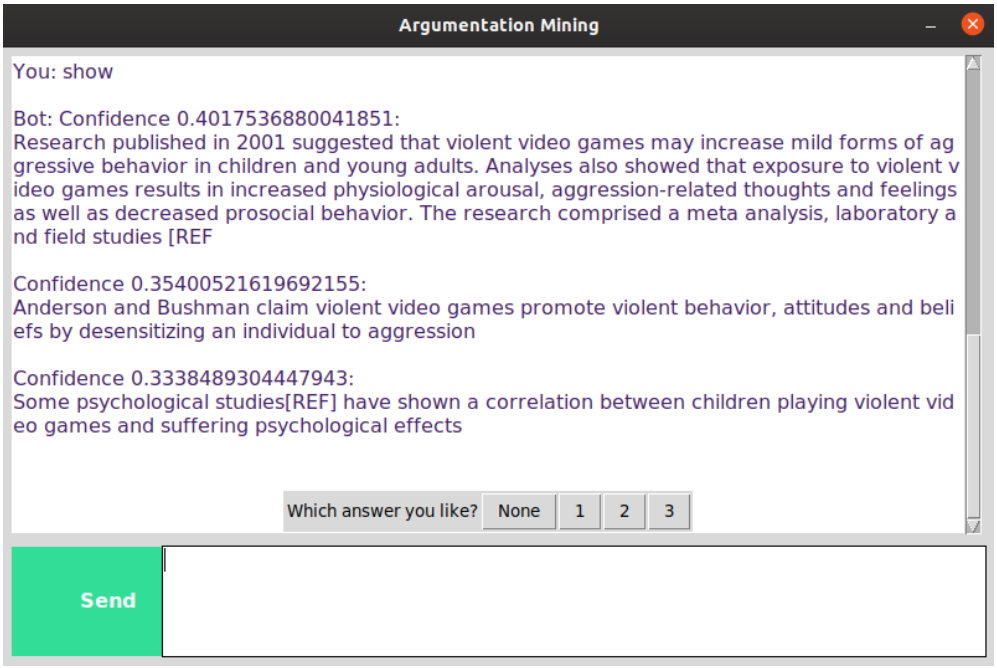

Figura 13) Output dopo 30 interazioni

Conclusioni

I prototipi descritti sono una prima esplorazione e si approcciano al problema di riconoscere argomenti simili tra input dell'utente e database in maniera semplice. Le possibilità di estensione sono quindi molteplici e verranno, in parte, descritte in questa sezione.

Prima di tutto, poiché la attinenza degli argomenti in output è delegata quasi interamente alla soggettività dell'utente stesso, è cruciale adattare l'output alla preferenza dei singoli utenti. Quindi, nonostante la seconda versione (ovvero il chatbot "dinamico") definisca un netto miglioramento in questo senso rispetto alla prima versione, ci sono ancora grosse limitazioni dovute all'utilizzo di tecniche approssimative per la rappresentazione del testo. Infatti, BoW, seppur efficace in questi prototipi, non tiene conto di molti fattori che caratterizzano un testo, come per esempio i tempi verbali o parole contestualmente simili. Pertanto, un eventuale sviluppo futuro, potrebbe essere la sostituzione della tecnica BoW con strumenti più avanzati di *deep learning*, oppure con tecniche BoW più complesse.

Risulta, però, complicato preferire una tecnica ad un'altra, in quanto è difficile decretare la correttezza di un argomento in output. Pertanto, definire una metrica di confronto è poco significativo, o addirittura impossibile, in quanto ogni individuo impone la sua soggettività nella scelta del risultato. Una possibile misurazione oggettiva potrebbe essere fatta sul numero medio di interazioni con ogni versione del chatbot che ogni utente deve compiere prima che il sistema produca un output adeguato alla soggettività dell'utilizzatore.

In qualsiasi caso, è necessaria una fase di *testing* con un bacino di utenza esteso, per rilevare il grado di soddisfazione di ogni utilizzatore nei confronti di varie versioni del chatbot e rendere questi risultati significativi.

Tornando alle questioni più a basso livello, la decisione di analizzare soltanto una *evidence* nel testo inserito come input dall'utente, nonostante abbia semplificato la progettazione del chatbot, è molto limitativa, in quanto il testo da esaminare

potrebbe essere molto lungo e ciò richiederebbe all'utente di inserire frase per frase, aspettando il completamento ogni volta dell'esecuzione.

Il codice del chatbot è già predisposto ad una estensione, in quanto è possibile richiedere tutti i componenti di una frase argomentativa (*claim, evidence, claim_evidence*) analizzati da MARGOT, sottoforma di array. L'analisi di tutti i componenti invece delle *evidence* solamente, permetterebbe di fornire un'esperienza più completa all'utilizzatore finale.

Per permettere un'applicazione del chatbot in diversi ambiti è necessario soprattutto definire *topic* differenti. Semplicemente, basta fornire un file Excel con la stessa struttura di quello IBM utilizzato, per ottenere immediatamente nuovi possibili output.

Infine, potrebbe rivelarsi utile salvare su file l'output prodotto dal chatbot, in modo tale da avere un documento persistente da poter consultare, in quanto l'output mostrato a schermo è volatile, viene cancellato ad ogni chiusura del programma.

Appendice

Soltanto un utente è riuscito a fornirmi la lista delle frasi testate nella seconda versione del chatbot, le quali verranno di seguito riportate. Tra parentesi graffe è riportato l'inizio dell'output selezionato, altrimenti [Nessuna risposta adatta], in caso di nessun voto.

FRASI RICONOSCIUTE

1) Children who play violent video games are more likely to have increased aggressive thoughts, feelings, and behaviors. [Research oublished in 2001...]
2) Video games may have bad effects on some children's health, including obesity, video-induced seizures. [Nessuna risposta adatta]
3) Too much video game playing makes your kid socially isolated. [Some phycological studies…]
4) People that play violent video games have decreased prosocial helping [When one combines all...]
5) People who watch a lot of simulated violence, such as those in video games, can become immune to it, they are more inclined to act violently themselves, and are less likely to behave emphatically. [Nessuna risposta adatta]
6) Chronic exposure to video games is associated with lower empathy and emotional callousness. [A 2001 study found...]
7) Excessive use of video games does not lead to long-term desensitization and lack of empathy. [Another way in which...]

8) Playing video games have a connection to aggressive thoughts and behaviour
   [Anderson and Bushman...]
9) Some experts who believe that there is a connection between video games and violence blame the games' interactive nature.
   [The court's decision...]
10) The act of violence is done repeatedly in video games.
   [Nessuna risposta adatta]
11) Video games can be addictive for kids, and that the kids' addiction to video games increases their depression and anxiety levels.
   [Another way in which…]
12) Kids addicted to video games see their school performance suffer.
   [Because violence...]
13) Addictive video games can have a similar effect on kids' brains as drugs and alcohol.
   [Incidece as such as Coulombine…]
14) The impulsive part of the brain, known as the amygdala-striatal system was more sensitive and smaller in excessive game players.
   [Nessuna risposta adatta]
15) Players who play games that requires players to navigate using spatial strategies like the 3D Super Mario games have increased grey matter in the hippocampus.
   [Nessuna risposta adatta]
16) Some video games teach kids the wrong values as violent behavior, vengeance and aggression are rewarded.
   [39% of the voters...]
17) Gamers who play violent games may feel guilty about their behavior in the virtual world and this may make them be more sensitive to the moral issues they violated during game play.
   [Nessuna risposta adatta]

18) Academic achievement may be negatively related to over-all time spent playing video games.
[After conducting...]

19) The more time a kid spends playing video games, the poorer is his performance in school.
[Nessuna risposta adatta]

20) Many game players routinely skip their homework to play games, and many students admitted that their video game habits are often responsible for poor school grades.
[Another way…]

21) Kids spending too much time playing video games may exhibit impulsive behavior and have attention problems.
[After conducting a two year…]

NON RICONOSCIUTE

1) Games can confuse reality and fantasy.
Viene riconosciuta se cambiata in:
Experts say that games can confuse reality and fantasy.
[There is a study being conducted...]

2) Video games may have bad effects on postural, muscular and skeletal disorders, such as tendonitis, nerve compression, carpal tunnel syndrome.
Viene riconosciuta se cambiata in:
Doctors observe that video games may have bad effects on postural, muscular and skeletal disorders, such as tendonitis, nerve compression, carpal tunnel syndrome.
[Nessuna risposta adatta]

3) Kid playing online can pick up bad language and behavior from other people, and may make it vulnerable to online dangers.

4) Most teenage players are able to leave the emotional effects of the game behind when the game is over.

5) In many games, kids are rewarded for being more violent
6) Gamers usually do not replace their offline social lives with online game playing, but rather it expands them.
7) Negotiating and other nonviolent solutions in video games are often not options.

   Viene riconosciuta se cambiata in:

   Spurio says that negotiating and other nonviolent solutions in video games are often not options. [Californian senator….]
8) Women are often portrayed in video games as weaker characters that are helpless or sexually provocative.
9) Childs playing games train the brain to come up with creative ways to solve puzzles and other problems in short bursts
10) In shooting games, the character may be running and shooting at the same time. This requires the real-world player to keep track of the position of the character, where he/she is heading, his speed, where the gun is aiming, if the gunfire is hitting the enemy, and so on.

Bibliografia

[1] Marco Lippi and Paolo Torroni. Argumentation mining: State of the art and emerging trends. ACM Trans. Internet Techn.

[2] Marco Lippi and Paolo Torroni. Margot: A web server form argumentation mining. *Expert Syst. Appl.*

[3] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. Second Edition, in progress

[4] IBM *Debater*: https://www.research.ibm.com/artificial-intelligence/project-debater/

[5] Ruggeri, Federico (2018) Predizione della struttura di un argomento con feature di stance classification. [Laurea magistrale], Università di Bologna, Corso di Studio in Ingegneria informatica [LM-DM270] <http://amslaurea.unibo.it/view/cds/CDS0937/>