Università degli studi di Torino

SCUOLA DI SCIENZE DELLA NATURA

Corso di Laurea Magistrale
Intelligenza Artificiale e Sistemi Informatici 'Pietro Torasso'

Tesi di Laurea Magistrale

Interazione vocale per la navigazione di espressioni matematiche

Primo Relatore:
Chiar.mo Prof. Alessandro Mazzei

Secondo Relatore:
Chiar.mo Prof. Luca Anselma

Controrelatore:
Chiar.ma Prof.ssa Rossana Damiano

Candidato:
Pier Felice Balestrucci

Anno Accademico 2020/2021

Vorrei ringraziare i professori
Luca Anselma, Alessandro Mazzei e
la professoressa Rossana Damiano
per la fiducia e la pazienza
che hanno avuto nei miei riguardi.

Un ringraziamento speciale
va al dott. Cristian Bernareggi per
i suoi preziosi consigli e l'aiuto
durante tutto il lavoro di tesi.

Infine, ringrazio la mia
famiglia e i miei amici
che mi hanno accompagnato
lungo tutto questo percorso.

Prefazione

Le tecnologie assistive sono quelle tecnologie che permettono di rendere accessibili e usabili prodotti informatici, hardware e software, anche a persone disabili.
Lo scopo di questa tesi è rendere più facile e comprensibile l'ascolto di formule matematiche a persone ipovedenti e cieche. Le formule matematiche sono ricche di simboli difficilmente leggibili dai lettori di schermo, ossia applicazioni software che identificano ed interpretano il testo mostrato sullo schermo del computer.
Generalmente chi ha una disabilità visiva per leggere le formule usa una rappresentazione LaTeX, la quale risulta non solo molto verbosa e lenta, ma costituisce una barriera per chi non conosce questo linguaggio.
L'obiettivo principale di questa tesi è la realizzazione di uno strumento che possa portare diversi vantaggi e semplificazioni a supporto di queste categorie utente. Questo strumento prevede sia la traduzione delle formule matematiche in *frasi matematiche*, ossia frasi in linguaggio naturale convertite con tecniche di Natural Language Generation, che l'introduzione di un sistema di dialogo per navigare ed esplorare la formula.
Questo progetto di tesi affonda le sue radici sul precedente lavoro svolto dal dott. Michele Monticone che affrontava la realizzazione di un sistema di generazione di frasi matematiche e la loro sintesi vocale in italiano.
Il mio lavoro espande quello del collega aprendosi ad un pubblico più ampio con la produzione di frasi matematiche in inglese e lo migliora con l'introduzione di un assistente vocale che possa aiutare l'utente a navigare all'interno della frase rispondendo a specifiche necessità come la ripetizione di sue parti per comprenderla meglio.

"Dichiaro di essere responsabile del contenuto dell'elaborato che presento al fine del conseguimento del titolo, di non avere plagiato in tutto o in parte il lavoro prodotto da altri e di aver citato le fonti originali in modo congruente alle normative vigenti in materia di plagio e di diritto d'autore.
Sono inoltre consapevole che nel caso la mia dichiarazione risultasse mendace, potrei incorrere nelle sanzioni previste dalla legge e la mia ammissione alla prova finale potrebbe essere negata."

Indice

Capitolo 1

Introduzione

Durante la lettura di testi scientifici si incontrano spesso formule e grafici che arricchiscono il testo, forniscono esempi e approfondiscono argomenti. Alla stregua del loro valore chiarificatore ed esemplificativo, spesso questi risultano essere totalmente o poco accessibili costituendo una vera e propria barriera per chi ha una disabilità visiva: non solo risultano essere poco comprensibili, ma ostacolano l'ascolto del testo.
A supporto di queste categorie utente esistono comunque vari strumenti come i lettori di schermo che identificano ed interpretano il testo mostrato sullo schermo del computer.
Tuttavia, ci sono numerosi margini di miglioramento dettati dall'avanzamento tecnologico degli ultimi anni che possono abbattere queste barriere.
Il problema non è banale e risulta complesso rispetto ai diversi possibili elementi che si trovano in un testo. Con questo lavoro si è andato incontro all'esigenza di rendere accessibili le formule matematiche creando un sistema che non solo generi una frase in linguaggio naturale che le rappresenti, chiamata *frase matematica*, per la lingua inglese, ma che introduca anche un sistema di dialogo con il quale l'utente può interagire e chiarificare i dubbi sulla formula.
La generazione di frasi matematiche risulta essere un processo articolato in diverse fasi, partendo dallo studio dei simboli e operatori matematici, passando alle relazioni che intercorrono fra questi, fino alla sintesi di una frase di senso compiuto che li aggrega e li ordina.
Il punto di partenza è stato il lavoro del dott. Monticone che affrontava la generazione di frasi matematiche e sintesi vocale in italiano. Questo però, mancava di diversi elementi che potevano conferirgli un ulteriore valore aggiunto. Ad esempio, l'uso della lingua italiana risulta limitare la sua fruizione ad un pubblico molto ristretto, così come la mancanza di strumenti per esplorare la formula e navigarla. Partendo da questo e raccogliendo spunti e idee dialogando direttamente con la categoria utente interessata, sono stati definiti nuovi requisiti ed è stato progettato un sistema espanso e migliorato.
Si è sviluppato quindi, un software costituito di due parti: il primo per la generazione di frasi matematiche, il secondo di dialogo, ciascuno dei quali mirato a rispondere alle esigenze descritte sopra.
Infine, è stata svolta una sperimentazione per ciascuna atta a testarli e valutarli.

1.1 Obiettivi

Gli obiettivi prefissati per questo progetto sono stati:

- Ricostruire il lavoro di generazione di frasi matematiche per la lingua inglese e valutarlo con una sperimentazione utente. Il sistema deve essere capace di tradurre una formula matematica in una frase matematica in inglese.
- Introdurre la proprietà di "navigabilità" nel sistema: l'utente può chiedere al sistema di ripetere parti dell'espressione matematica. Ad esempio, data l'espressione $a + b + c$ può chiedere di ripetere la frase da b.
- Ampliare il sistema di generazione con un sistema di dialogo che tiene conto della struttura ad albero della formula: il sistema deve generare diverse frasi che descrivono la formula nei suoi diversi livelli dell'albero.

1.2 Panoramica del sistema

Nella seguente sezione si traccia una panoramica del sistema descrivendo come è costituito, specificando le diverse scelte di progetto fatte e commentando l'architettura e i moduli con cui è formata.

1.2.1 Sistema di generazione di frasi matematiche

Il lavoro di tesi si compone di due sistemi distinti che comunicano fra loro: il primo è di generazione di frasi matematiche, il secondo di dialogo.
Il sistema di generazione di frasi matematiche qui descritto è un sistema che partendo dalla rappresentazione LaTeX della formula permette di ottenere una sua traduzione in linguaggio naturale. Infatti, la formula in questa specifica rappresentazione viene convertita prima in un linguaggio, chiamato Content Math Markup Language, che ne esplicita la semantica, e poi sfruttando un'apposita libreria Java, chiamata SimpleNLG [1], e i risultati di un'analisi fatta sugli elementi matematici, in linguaggio naturale.

Rappresentazione delle formule matematiche

Una parentesi necessaria al fine di comprendere la struttura dell'applicazione è quella di raccontare le basi su cui si fonda, ossia come rappresentare le formule.
Il formalismo usato per scriverle è il LaTeX, standard di riferimento nella scrittura di testi matematici e formule chimiche.

Si guardi al seguente esempio:

- Espressione della formula in LaTeX:

$$\sqrt{x} - \frac{x+1}{x}=0$$

- La sua rappresentazione:

$$\sqrt{x} - \frac{x+1}{x} = 0$$

Esempio 1: Rappresentazione formula in LaTeX

La matematica è meno intuitiva da rappresentare rispetto ad un semplice testo perché le formule matematiche devono essere contrassegnate da una stringa unidimensionale di caratteri seppur siano in realtà, costrutti bidimensionali: lo spazio della pagina è sfruttato continuamente con simboli disposti su più livelli (esponenti, pedici, frazioni, matrici) oltre che con diagrammi, grafici e disegni di vario genere.
Tuttavia, anche la formula più complicata è scomponibile in elementi più semplici.
Il LaTeX mette a disposizione un insieme di simboli matematici che permettono in maniera costruttiva di rappresentare una formula [2].
Nell'esempio 1, i simboli *\sqrt* e *\frac* sono usati per rappresentare rispettivamente la radice quadrata e la frazione, mentre nelle parentesi graffe sono presenti i loro argomenti.
Accanto alla rappresentazione tipografica della formula è necessario definirne anche una semantica.
Il formalismo in questo caso usato è il Content Math Markup Language (CMML).
Il CMML è un formato XML che permette di rappresentare la semantica con l'uso di un insieme chiuso di tag.
Questo permette ad esempio, ad un simbolo di funzione di venire descritto da un tag specifico; al contrario, non è possibile rappresentare la semantica di quei simboli che non sono definiti nella definizione del linguaggio [3].
Ad esempio, la formula $a + b$ viene rappresentata così:

<apply> <plus/> <ci> a </ci> <ci> b </ci> </apply>

Esempio 2: Formula in CMML

Laddove il tag *<ci>* identifica una variabile, il tag *<plus/>* la somma tra gli elementi presenti all'interno del binding di *<apply>*.

1.2.2 Sistema di dialogo

Parte centrale di questo lavoro di tesi è stata la realizzazione del sistema di dialogo.
Lo scopo è di fornire assistenza all'utente permettendogli di ripetere parti della frase e analizzarle nel dettaglio. Il sistema deve pertanto poter interrompere la pronuncia

della frase in qualsiasi momento e ascoltare l'utente.
Ricevuta la richiesta, i passi successivi sono decodificarla, produrre il risultato atteso ed infine, fornirlo in output.
Il sistema, quindi, riprenderà la dettatura dell'espressione prima interrotta.

Principali problematiche

Un sistema così pensato, tuttavia, presenta diversi problemi:

1. **Come sintetizzare la frase matematica?**
   Il sistema deve essere capace di pronunciare tutta l'espressione matematica e potersi interrompere in un qualsiasi momento. Non è possibile però, sapere quando sarà fermato, e quindi si potrebbe immaginare di avere un insieme di sotto-frasi che vengono invece pronunciate.

2. **Come evitare che il sistema confonda le richieste utente?**
   Ad esempio, il problema sorge quando, nonostante il sintetizzatore stia pronunciando la frase, deve anche mettersi in ascolto e non deve confondere quello che sta dicendo con la richiesta dell'utente.

3. **Quali richieste permettere di fare all'utente?**
   Un generico utente potrebbe far qualsiasi domanda inerente a quello che sta ascoltando. Tuttavia, essere a conoscenza a priori di come è costituita l'espressione matematica ovviamente permette di predire quali domande o quali parti della frase ripetere.

La risoluzione a questi problemi sarà ovviamente spiegata nei successivi capitoli.

1.2.3 Architettura dei sistemi

L'architettura dei sistemi si articolata in diversi moduli e viene riassunta dal seguente grafico:

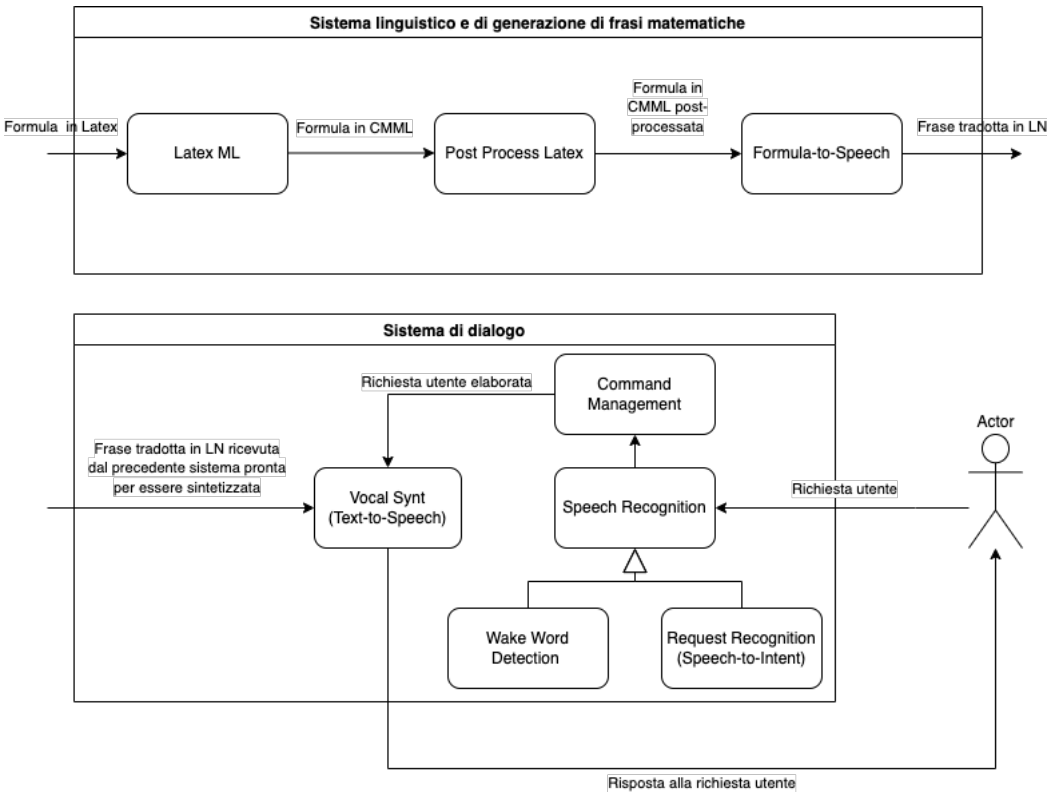

Fig. 1.1: Architettura del sistema

I primi tre moduli in alto sono quelli che si occupano di tradurre la formula dalla sua rappresentazione in LaTeX in una espressione inglese corretta. In particolare:

- **LatexML** permette la traduzione della formula dal LaTeX al CMML;
- **Post process Latex** corregge e post-processa il CMML prodotto nel modulo precedente;
- **Formula-to-Speech**, sfruttando la libreria java SimpleNLG, permette di generare il corrispondente testo;

Tale testo è dato in input al sistema di dialogo che è composto sostanzialmente da altri tre moduli:

- **Vocal Synt** si occupa di vocalizzare il testo;
- **Speech Recognition** permette di riconoscere eventuali richieste dell'utente;

- **Command Management** di analizzare la richiesta ricevuta;

In particolare, il modulo di Speech Recognition è composto da una parte che si occupa di riconoscere la *wake up word* e da una seconda che permette di riconoscere le richieste utente.
Una volta che viene riconosciuta una richiesta utente, questa viene inviata al modulo Command Management che analizza la richiesta, la traduce e la elabora. Trovata una risposta che la soddisfi, questa viene inviata al Vocal Synt per poter essere sintetizzata vocalmente.

1.3 Struttura della tesi

La tesi è articolata in quattro capitoli:

1. Nel capitolo 1 si affrontano i motivi per cui tale progetto nasce e quindi, a quali esigenze risponde. Inoltre, si delinea sommariamente la struttura del sistema.

2. Il capitolo 2 tratta la rappresentazione sintattica e semantica delle espressioni matematiche e come da esse si riescano a generare frasi in linguaggio naturale.

3. Nel capitolo 3 si tracciano le caratteristiche del sistema di dialogo quindi, come è stato pensato e realizzato per essere un valore aggiunto. Si analizzano le varie tecnologie a supporto e infine, come è stato implementato.

4. Nell'ultimo capitolo si discute dei risultati ottenuti, i pro e i contro, ponendo l'accento su ulteriori sviluppi futuri.

Capitolo 2

Analisi e generazione di espressioni matematiche

Il sistema per come è stato pensato e poi progettato getta le sue fondamenta sulla generazione di espressioni matematiche in linguaggio naturale a partire da una loro rappresentazione in LaTeX.
In questo capitolo si esaminerà l'iter della generazione analizzando prima come in letteratura si affronta il problema della generazione di frasi del linguaggio naturale e successivamente ponendo il focus sulle espressioni matematiche.

2.1 Linguaggio naturale

La Treccani definisce il linguaggio naturale come: "[...] il linguaggio usato nella comunicazione fra individui di un gruppo sociale che lo condivide; presenta una sua ricchezza espressiva, ma anche sfumature e ambiguità, per cui matematica e logica tendono a ridurne l'utilizzo. [...]
Il settore di studi denominato "intelligenza artificiale" ha fra i suoi obiettivi quello di mettere un automa in condizione di esprimersi tramite il linguaggio naturale, ma per raggiungere tale obiettivo è necessario scrivere programmi che siano in grado di comprendere e produrre discorsi in linguaggio naturale" [4].
La Natural Language Generation è un processo software che permette di automatizzare la produzione di linguaggio naturale sotto determinate premesse.
Nei seguenti paragrafi si esamineranno in breve varie modalità di rappresentazione della struttura sintattica del linguaggio tramite alberi a costituenti e alberi a dipendenze note in letteratura, per poi descrivere come per questo lavoro è stata rappresentata tramite la libreria scritta in Java, SimpleNLG.

2.1.1 Alberi a costituenti

Una rappresentazione del genere è possibile assumendo che il linguaggio naturale sia generato da una grammatica Context-Free, ossia da un insieme di regole o produzioni, ciascuna delle quali esprime i modi con cui i simboli del linguaggio possono essere raggruppati e ordinati insieme, e un lessico di parole e simboli.

Un esempio di grammatica Context-Free, tratto dal testo *Speech and Language Processing* [5], è il seguente:

S	→	NP VP	I + want a morning flight
NP	→	Pronoun	I
	→	Proper-Noun	Los Angeles
	→	Det Nominal	a + flight
Nominal	→	Nominal Noun	morning + flight
	→	Noun	flights
VP	→	Verb	do
	→	Verb NP	want + a flight
	→	Verb NP PP	leave + Boston + in the morning
	→	Verb PP	leaving + on Thursday
PP	→	Preposition NP	from + Los Angeles

Tab. 2.1: Grammatica Contex-Free

In particolare, nella tabella 2.1 i simboli presenti indicano:

1. **S** (Sentence): una frase nella sua interezza;
2. **NP** (Noun Phrase): una frase che ha un nome o pronome come testa o svolge la stessa funziona grammaticale di un nome. Es. "many people";
3. **VP** (Verbal Phrase): unità sintattica composta almeno da almeno un verbo e i suoi dipendenti;
4. **PP** (Prepositional Phrase): hanno una preposizione come testa della frase. Es. "She walked *to his desk*";
5. **Det**: indica un articolo;
6. **Proper-Noun**: indica un nome proprio;
7. **Noun**: indica un nome;
8. **Verb**: indica un verbo;
9. **Preposition**: indica una preposizione;
10. **Nominal**: segue l'articolo e contiene ogni pre- e post- modificatore della testa dei nomi;

Fissata questa ipotesi e formalizzando sia una grammatica come quella nell'esempio precedente che definendo un determinato lessico è possibile costruire un albero a costituenti per la frase "I prefer a morning flight":

Noun	→	flights | breeze | trip | morning
Verb	→	is | prefer | like | need | want | fly
Adjective	→	cheapest | non-stop | first | latest | other | direct
Proper-Noun	→	Alaska | Baltimore | Los Angeles | Chicago |United | American
Pronoun	→	me | I | you | it
Determiner	→	the | a | an | this | these | that
Preposition	→	from | to | on | near
Conjunction	→	and | or| but

Tab. 2.2: Lessico per linguaggio L_0 [6]

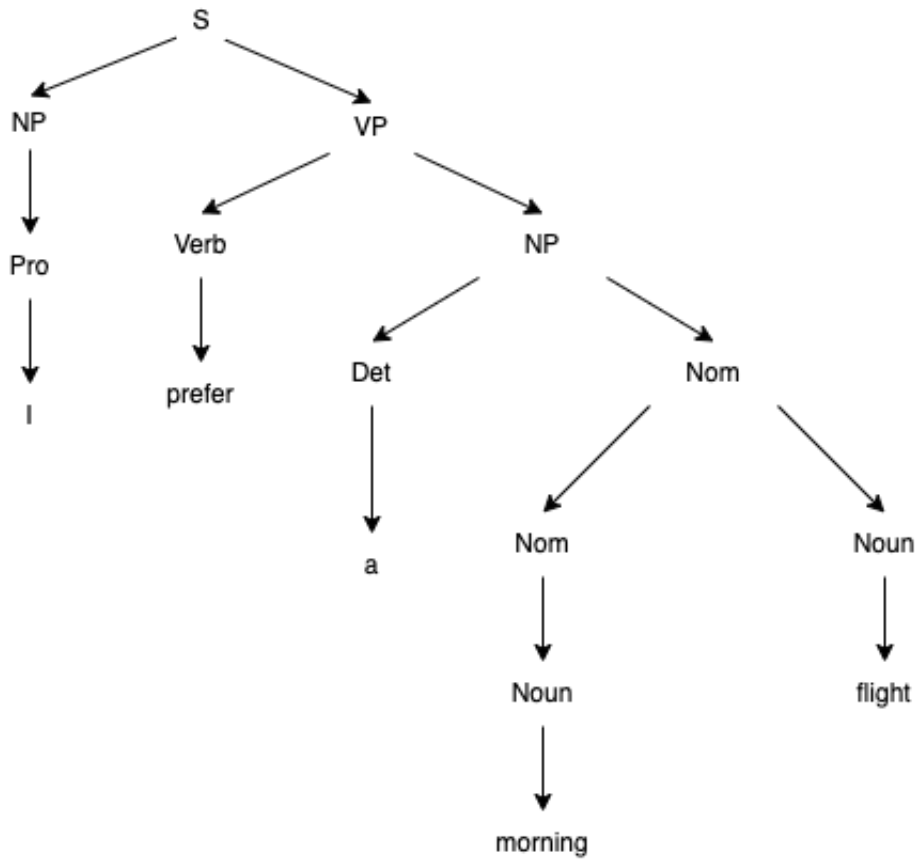

Fig. 2.1: Albero a costituenti [7]

2.1.2 Alberi a dipendenze

Un altro tipo di formalismo grammaticale è quello a dipendenze, il quale sta diventando sempre più importante nell'ambito dello speech e languange processing.

Per questa classe di grammatiche la struttura sintattica di una frase è descritta puramente in termini di parole e per relazioni binarie sintattiche tra queste parole.
L'immagine qui di seguito mostra l'esempio di un albero le cui dipendenze sono tipizzate, per la frase "They hid the letter on the shelf" [8]:

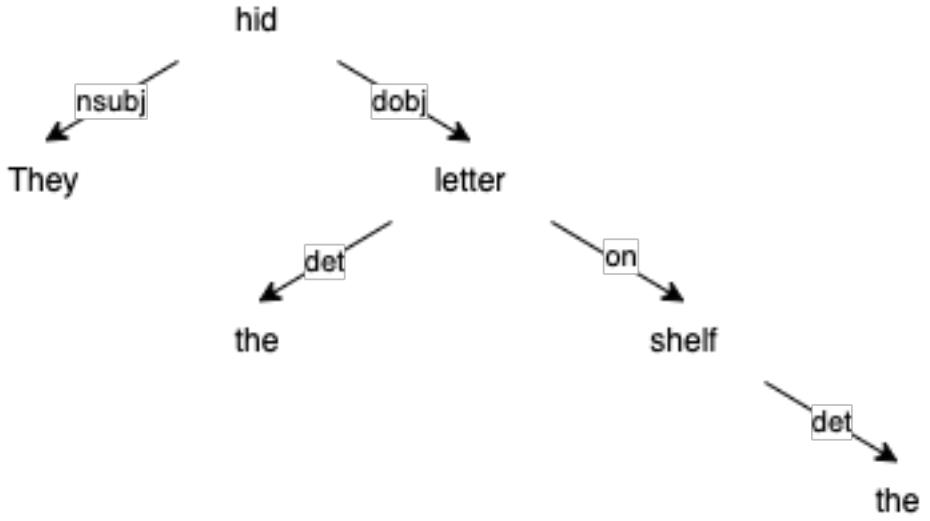

Fig. 2.2: Albero a dipendenze

Laddove le principali dipendeze fra gli elementi sono:

1. **subj**: soggetto della frase;
2. **obj**: complemento oggetto della frase;
3. **indirect-obj**: oggetto indiretto della frase, tipico dei verbi ditransitivi;
4. **pre-modifier**: pre modificatore di una parola, come un aggettivo;
5. **post-modifier**: post modificatore di una parola, come un avverbio;
6. **complement**: generico complemento;
7. **coordinatee**: coordinata fra due parole o frasi;
8. **det**: articolo;
9. **prep**: preposizione;
10. **conj**: congiunzione;

2.2 SimpleNLG

La definizione e costruzione delle frasi per questo lavoro è stata svolta con SimpleNLG. SimpleNLG è una libreria sviluppata in Java che offre il controllo diretto sul processo con cui le frasi sono costruite, sulle operazioni morfologiche e sulla linearizzazione. Inoltre, definisce un insieme di tipi lessicali e frasali che corrispondono alle categorie grammaticali e mette a disposizione delle impostazioni per modificare diverse feature. La costruzione della struttura sintattica e linearizzazione del testo avviene nei seguenti passi:

1. Si inizializzano i costituenti di base con gli elementi lessicali appropriati;
2. Si definiscono l'insieme delle feature dei costituenti (ad esempio la coniugazione per un verbo);
3. Si combinano i costituenti in strutture più grandi;
4. La struttura ottenuta viene passata al lineariser che applica le inflessioni corrette e definisce l'ordine delle parole sulla base delle feature;

Si veda il seguente esempio per la costruzione dell'espressione "the boys left the house", inizializzando al frase con il verbo "leave" e impostando la feature "Tense".

```
1. Phrase s1 = new SPhraseSpec(‘leave’);
2. s1.setTense(PAST);
3. s1.setObject( new NPPhraseSpec(‘the’, ‘house’));
4. Phrase s2 = new StringPhraseSpec(‘the boys’);
5. s1.setSubject(s2);
```

Nella prima riga di codice si definisce l'espressione come una SPhraseSpec che permette di specificare il verbo principale; mentre nella seconda si definisce il tempo verbale, il passato.
Nella terza si definisce il complemento oggetto, ossia una NNPhraseSpec che prevede in ordine un articolo e un nome.
Nella quarta linea di codice si può leggere come il soggetto è definito come una stringa unica, per poi definirlo effettivamente soggetto di s1 [9].

2.3 Espressioni matematiche

La struttura delle espressioni matematiche risulta essere dipendente direttamente dagli operatori che le compongono. Risultano in tal senso, essere differenti da tipiche costruzioni di frasi del linguaggio naturale.
Se volessimo costruire la corrispondente espressione per la formula $2 + 3$ potremmo pensare di considerare l'operatore della somma come un verbo, mentre il due e il tre come rispettivamente soggetto e oggetto.

Nel successivo paragrafo si analizzeranno vari operatori, e più in generale i simboli matematici, raggruppati in categorie sulla base della loro struttura linguistica.
Tale analisi è stata fatta considerando le espressioni presenti nel *Handbook for Spoken Mathemathics* [10], che costituisce uno standard inglese per la cosiddetta matematica parlata. Come già scritto prima infatti, la matematica è principalmente visuale e quando viene presentata oralmente può risultare ambigua. Nel tentativo di mitigare questo problema principalmente per ciechi e ipovedenti, questo manuale propone alcuni modi consistenti e ben definiti di pronunciare le espressioni matematiche affinché gli ascoltatori possano recepire rappresentazioni chiare, non ambigue e correttamente pronunciate.

2.3.1 Elementi matematici

Come scritto in precedenza, in questa sezione si farà una disamina di come sono organizzati i simboli matematici per la costruzione di frasi nel linguaggio naturale. Questi sono divisi in:

- Operatori relazionali
- Operatori algebrici, aritmetici e insiemistici
- Connettivi Logici
- Insieme condizionale
- Coppia
- Ausiliari
- Sequenze
- Funzioni elementari
- Calcolo

Per ciascuna categoria sarà riportata una tabella che descrive l'operatore (cioè corrispondente simbolo), come è identificato in CMML e come viene letto nel linguaggio naturale.

Operatori relazionali

Questi operatori come suggerisce il nome, definiscono la relazione che intercorre tra due o più elementi nelle formule:

a) x is *much greater than* y

b) x *belongs to* A

Esempio 3: Operatori relazionali

Simbolo	**Operatore**	**Forma Inglese**
$>$	gt	is greater than
$\geq$	geq	is greater than or equal to
$\gg$	gg	is much greater than
$<$	lt	is less than
$\leq$	leq	is less than or equal to
$\ll$	ll	is much less than
$=$	eq	is equal to
$\neq$	neq	is not equal to
$\subseteq$	subset	is subset of or equal to
$\subset$	prsubset	is subset of
$\nsubseteq$	notsubset	is not subset of or equal to
$\not\subset$	notprsubset	is not subset of
$\circ$	compose	composed with
$\in$	in	belongs to
$\notin$	notin	not belongs to
$\exists$	exists	exists such that

Tab. 2.3: Operatori relazionali

Si può osservare analizzando la traduzione di questi operatori nella tabella 2.3 che questa è generalmente costituita da un verbo, talvolta un aggettivo e un sostantivo, ed infine una preposizione. Soffermandosi, però, più a fondo è possibile realizzare un'analisi molto più significativa.

Guardando l'esempio 3 si possono analizzare le parti del discorso che costituiscono rispettivamente le due frasi per poi cercare di definire una grammatica Contex-Free che possa generarle.
Nel dettaglio:

a) (x, N) (is, Verb) (much, Adv) (greater, Adj) (than, P) (y, N)

b) (A, N) (belongs, Verb) (to, P) (A, N)

Pertanto una possibile grammatica Context-Free può essere la seguente:

S → NP VP
VP → V AdjP PP | V PP
AdjP → Adj | Coord | Adv AdjP
Coord → Adj Conj Adj
NP → Det N | N
PP → P NP

Tab. 2.4: Grammatica Context-Free per operatori relazionali

Dal lato implementativo, la sua realizzazione con SimpleNLG prevede quindi, la creazione di una frase verbale per cui si definiscono sia soggetto che oggetto e in cui si compone una parte preposizionale per cui si specifica in caso un modificatore (es. nel caso di "much greater").

```
304         ["rel-op"]
305         (let [vp (. nlg-factory createVerbPhrase (operator :verb))
306               obj (generate-object operator)
307               op1 (generate-operand operand1-node aggregation-method)
308               op2 (generate-operand operand2-node aggregation-method)
309               pp (generate-prepositional-phrase (operator :preposition) op2)
310               s (. nlg-factory createClause op1 vp obj)]
311          (doto s
312            (. setFeature Feature/NEGATED (operator :is-negate))
313            (. setFeature Feature/SUPRESSED_COMPLEMENTISER true)
314            (. addPostModifier pp))
315          s)
```

Fig. 2.3: Codice per operatori relazionali

Come è possibile vedere dal codice scritto in Clojure della figura 2.3 si definiscono:

1. alla riga 305 la frase verbale che ha come parametro il verbo definito dall'operatore;
2. alla riga 306 l'oggetto della frase, ossia l'operatore stesso;
3. dalla riga 307 alla 308 la generazione degli operandi in base alla specifica del CMML;
4. alla riga 309 la frase preposizionale relativa al secondo operando;
5. dalla 312 alla 314 le feature della frase come ad esempio l'aggiunta di un post modificatore;

Operatori algebrici, aritmetici e insiemistici

Simbolo	Operatore	Forma Inglese
$+$	plus	plus
$-$	minus	minus
$/$	divide	over
$*$	times	times
$[...]^{[..]}$	power	to
$\setminus$	settdiff	minus
$\times$	cartesianproduct	cross
$\cap$	intersect	the intersection of
$\cup$	union	the union of ... and ...

Tab. 2.5: Operatori algebrici, aritmetici e insiemistici

Questa categoria di operatori corrisponde alla famiglia di operazioni algebriche, aritmetiche e insiemistiche. Nello specifico è possibile determinare due tipi di forme: unarie

e binarie. Ad esempio, l'operatore "minus" può definire un'espressione del tipo "a - b" oppure può svolgere il ruolo di aggettivo definendo un valore negativo, "-5".
Per tale motivo, in base ai nodi che compongono l'operatore la realizzazione è stata resa differente.

a) minus 5

b) a times b

Esempio 4: Operatori algebrici

Come per la precedente categoria, si analizzeranno di seguito le parti del discorso per l'esempio 4 e si proporrà una grammatica Context-Free per queste:

a) (minus, Adj) (5, N)

b) (a N) (times, V) (b, N)

S → NP VP | NP
VP → V NP
NP → N | Adj N
PP → P NP

Tab. 2.6: Grammatica Context-Free per operatori algebrici, aritmetici e insiemistici

Dal punto di vista realizzativo, bisogna fare una distinzione netta fra le due forme possibili per questa categoria di operatori.

1. **Forma unaria**: per questo caso viene creata una cosiddetta "frase aggettivo" per cui l'operatore svolge il ruolo di aggettivo dell'elemento (minus 5);
2. **Forma binaria**: si costruisce una frase che ha come nodo padre l'operatore stesso e come figli i due elementi della formula. Inoltre, come si può notare della tabella 2.3 ci sono casi in cui compaiono anche delle preposizioni: le si aggregano alla frase ponendole dopo l'operatore;

Connettivi logici

I connettivi logici sono operatori che connettono sotto formule logiche.

Simbolo	Operatore	Forma Inglese
$\wedge$	and	and
$\vee$	or	or
$\neg$	not	not
$\Longrightarrow$	implies	if ... than ...
$\Longleftrightarrow$	if and only if	if and only if

Tab. 2.7: Connettivi logici

a) a and b

b) if a than b

Esempio 5: Connettivi logici

Anche per questo caso, analizzando l'esempio 5 otteniamo:

a) (a, N) (and, Conj) (b, N)

b) (if, Conj) (a, N) (than, Conj) (b, N)

Questo suggerisce che una possibile grammatica possa essere:

Coord → S Conj S | Conj S Conj S
S → NP
NP → N

Tab. 2.8: Grammatica Context-Free per connettivi logici

La realizzazione di questi operatori prevede la creazione di una frase preposizionale per cui sono specificati due operandi (nell'esempio 5, "a" e "b"). Inoltre, se l'operatore lo prevede, viene definita una frase coordinata, come nel caso dell'implicazione che lega le sotto frasi.

Insieme condizionale

In questa categoria rientra un unico operatore che permette di descrivere l'insieme condizionale.

Simbolo	**Operatore**	**Forma Inglese**
$[vars][cond]$	conditional-set	the set of ... such that ...

Tab. 2.9: Insieme condizionale

Esempio:

a) The set of x such that x is greater than 4

Esempio 6: Insieme condizionale

Le parti del discorso che contraddistinguono la frase dell'esempio 6 sono:

a) (The, Det) (set, N) (of, P) (x, N) (such that, P) (x, N) (is, V) (greater, Adj) (than, P), (4, N)

Mentre una grammatica Context-Free può essere:

S → NP VP | ...
NP → Det N | NP PP |...
VP → V PP |...
PP → P NP | P S |...

Tab. 2.10: Grammatica Context-Free per l'insieme condizionale

Tramite SimpleNLG è possibile costruire frasi che prevedono una prima parte che rappresenta gli elementi dell'insieme ed una seconda che determina le loro caratteristiche.

Coppia

Come per il precedente paragrafo anche in questa categoria troviamo un unico operatore che definisce l'accoppiamento fra due elementi.

Simbolo	**Operatore**	**Forma Inglese**
([..], [...])	pair	.. and ... pair

Tab. 2.11: Coppia

Esempio:

a) x and y pair

Esempio 7: Coppia

Le parti del discorso sono per l'esempio 7:

a) (x, N) (and, Conj) (y, N) (pair, N)

Invece, una possibile grammatica generativa è:

NP → N Conj NP | N

Tab. 2.12: Grammatica Context-Free per la coppia

Le frasi matematiche di questa categoria sono contraddistinte dai due operatori che compongono la coppia e che fungono da aggettivo per il sostantivo "pair".

Ausiliari

Nella suddetta categoria rientrano le unità lessicali di completamento usati da altri operatori.

Simbolo	Operatore	Forma Inglese
$\int^{\cdots}, \prod^{x=\ldots}, \sum^{x=\ldots}$	uplimit	to
$\int_{\ldots}, \prod_{x=\ldots}, \sum_{x=\ldots}, \lim_{x\to\ldots}$	lowlimit	to
$x_{\ldots}$	subscription	with

Tab. 2.13: Ausiliari

Esempio:

a) a to 0

b) x with 0

Esempio 8: Ausiliari

Le parti del discorso sono per l'esempio 8:

a) (a, N) (to, Prep) (0, N)

b) (x, N) (with, Prep) (0, N)

Laddove una grammatica Context-Free può essere:

NP $\rightarrow$ NP PP | N
PP $\rightarrow$ P N

Tab. 2.14: Grammatica Context-Free per gli ausiliari

La realizzazione di questi operatori si distingue dagli altri per la sua semplicità. L'idea è quella di legare il primo operando con il secondo tramite la preposizione inerente.

Sequenze

Con questa etichetta si identificano quegli operatori che rappresentano sequenze di valori o di operazioni.

Simbolo	**Operatore**	**Forma Inglese**
$\sum$	sum	the summation of ... from ... equal
$\prod$	prod	the product of ... for ...
lim	limit	the limit of ... as ... approach

Tab. 2.15: Sequenza

Esempio:

a) The product of I for x

b) The limit of x as 0 approaching to n

c) The summation from x equals a to b of sine x

Esempio 9: Sequenze

Dall'analisi delle parti del discorso risulta che:

a) (The, Det) (product, N) (of, P) (I, N) (for, P) (x, N)

b) (The, Det) (limit, N) (of, P) (x, N) (as, Adv) (0, N) (approaching, V) (to, P) (n, N)

c) (The, Det) (summation, N) (from, P) (x, N) (equals, V) (a, N) (to, P) (b, N) (of, P) (sine, N) (x, N)

Mentre una grammatica che può generare questa categoria di operatori è:

S $\rightarrow$ NP | NP VP
NP $\rightarrow$ Det N | NP PP | N NP | N
VP $\rightarrow$ V NP
PP $\rightarrow$ P NP

Tab. 2.16: Grammatica Context-Free per le sequenze

Come per gli operatori descritti nella tabella 2.5, anche per questi è necessario fare una distinzione in quanto la loro costruzione dipende dalla loro arietà. Distinguiamo tre casi:

1. **Forma unaria**: ad esempio la prima espressione dell'esempio 9 appartiene a tale caso. In particolare, la sua realizzazione prevede di legare l'operatore al suo argomento tramite una preposizione;

2. **Forma ternaria**: la costruzione si complica perché si introducono più argomenti per l'operatore. Guardando alla seconda frase dell'esempio 9, ci sono tre operandi che svolgono il ruolo rispettivamente di limite inferiore, limite superiore e argomento. A questi poi è necessario associare la corretta preposizione e il modificatore. Si definisce infatti una frase che è costituita da un soggetto (il limite) ed una sotto-frase verbale di cui il verbo è "approaching", il soggetto è "x", mentre l'oggetto è "0". Infine, si aggiunge una frase preposizionale che compone l'ultima parte.

3. **Forma quaternaria**: si guardi all'ultima frase dell'esempio già citato. Rispetto alla precedente, viene considerato anche l'argomento dell'operatore. La costruzione quindi, è simile al precedente con l'accortezza di aggregare alla frase un complemento che rappresenti il suddetto argomento.

Funzioni elementari

Le funzioni elementari identificano sia gli operatori trigonometrici che le funzioni matematiche.

Simbolo	**Operatore**	**Forma Inglese**
sin	sine	sine
cos	cosine	cosine
tan	tangent	tangent
arcsin	arcsine	arcsine
arccos	arcosine	arcosine
arctan	arctangent	arctangent
$\|...\|$	abs	the absolute value of ...
$\sqrt[...]{x}$	the root	...-ed root of x
$N!$	factorial	the factorial of
f^{-1}	inverse	the inverse of

Tab. 2.17: Funzioni elementari

Si prendano come esempio le seguenti espressioni matematiche:

a) The inverse of x

b) Sine x

Esempio 10: Funzioni elementari

Laddove da un'analisi delle parti del discorso per l'esempio 10 risulta:

a) (The, Det) (inverse, N) (of, P) (x, N)

b) (Sine, N) (x, N)

Mentre, una grammatica Context-Free può essere:

NP → Det NP | NP PP | Adj NP | N PP → P NP

Tab. 2.18: Grammatica Context-Free per le funzioni elementari

La realizzazione per questi operatori si sviluppa definendo una *noun phrase* a cui viene legato l'operando tramite la corretta preposizione e pre-modificatore se richiesti.

Calcolo

L'ultima categoria è costituita da integrali e derivate.

Simbolo	**Operatore**	**Forma Inglese**
$\int$	integral	the integral of ... dx
$f'(x)$	diff	the derivative of x with respect to

Tab. 2.19: Calcolo

Si considerino i seguenti esempi:

a) The derivative of sine x with respect to x

b) The integral from b to c of d d x

Esempio 11: Calcolo

Per le quali le parti del discorso sono:

a) (The, Det) (derivative, N) (of, P) (sine, N) (x, N) (with respect to, P) (x, N)

b) (The, Det) (integral, N) (from, P) (b, N) (to, P) (c, N) (of, P) (d, N) (d, Adj) (x, N)

Una grammatica per questa categoria può essere:

NP → Det N | NP PP | NP Adj NP | N PP → P NP

Tab. 2.20: Grammatica Context-Free per calcolo

Infine, la realizzazione di questi due operatori si può sviluppare in due forme distinte:

1. **Forma binaria**: gli elementi che vengono considerati sono l'operatore e il suo argomento, a cui è necessario legare una serie di preposizioni.

2. **Forma quaternaria**: per quest'ultima si considerano invece l'operatore, il suo argomento e, il limite inferiore e superiore. Si definisce quindi una frase nominale a cui si aggrega l'argomento come oggetto e in ordine limite inferiore e superiore legati tramite la corretta preposizione.

2.3.2 Caratterizzazione delle espressioni matematiche per il parlato

L'ultimo passo legato alla sintesi delle espressioni matematiche è quella di caratterizzazione per adattarle al parlato. La lingua parlata differisce notevolmente da quella scritta, la quale identifica un insieme di caratteristiche strutturali e funzionali. Infatti, subentra il sistema fonico-uditivo e determina un insieme di proprietà quali:

1. L'organizzazione temporale del segnale;
2. La non-ripetibilità e permanenza del segnale;
3. La contemporaneità tra produzione e ricezione del segnale;

Ad esempio, nel sistema fonico-uditivo i suoni si susseguono nel tempo invece di essere simultanei nello spazio e i parametri connessi ai fenomeni temporali (ritmo, velocità e durata) permeano l'intero processo di produzione e ricezione della parola, sia a livello segmentale che prosodico [11].
Al fine di caratterizzare l'espressione matematica e renderla può vicina ad una frase parlata si è utilizzato lo Speech Synthesis Markup Language.

Speech Synthesis Markup Language

Lo Speech Synthesis Markup Language (SSML) è un linguaggio di markup basato su XML specificato dal gruppo di studio "Voice Browser" del W3C. I tag definiti in questo linguaggio permettono di controllare la resa di un sistema di sintesi durante tutte e sei le fasi del processo tipico di elaborazione:

1. Scansione dell'XML;
2. Analisi della struttura;
3. Normalizzazione del testo;
4. Conversione testo-fonema;
5. Analisi della prosodia;
6. Generazione dell'audio;

Un esempio di una frase arricchita da questo linguaggio è il seguente:

```
<speak>
    <prosody rate="slow">
       A plus <break time="500ms"/> three over two.
    </prosody>
</speak>
```

Laddove la prosodia lenta indica che il sintetizzatore pronuncerà lentamente la frase; mentre il "break time" indica la presenza di una pausa di 500ms tra "plus" e "three".

2.4 Implementazione del sistema di generazione di frasi matematiche

Nella seguente sezione si andrà ad illustrare l'implementazione del sistema generativo delle espressioni matematiche, partendo dalla conversione del LaTeX in CMML. L'IDE con cui è stata sviluppata questa parte del sistema è stato IntelliJ IDEA e i linguaggi di programmazione usati sono stati Clojure [12] (linguaggio funzionale compilato ed eseguito dalla Java Virtual Machine (JVM)) e Java [13].
È possibile reperire il codice al seguente link Github: https://github.com/PierBale/MathFormulaVerbalization.

2.4.1 Dal LaTeX al Content Math Markup Language

Come introdotto nella sezione 1.2.1 il CMML è un linguaggio di markup utile per la rappresentazione della semantica di una espressione matematica.
È possibile convertire automaticamente l'espressione dal LaTeX al CMML tramite uno strumento chiamato LatexML.
Accanto a questo strumento, si è vista la necessita di fare del post-processing per ripulire e correggere eventuali errori che tale software può inaspettatamente causare.
Ad esempio, si consideri l'insieme condizionale: $A = \{x|x > 0\}$.

L'output ottenuto dal LatexML è il seguente:

```
<m:apply>
    <m:eq/>
    <m:ci>A</m:ci>
    <m:apply>
       <m:csymbol cd="latexml">conditional-set</m:csymbol>
       <m:ci>x</m:ci>
       <m:apply>
           <m:gt/>
           <m:ci>x</m:ci>
           <m:cn type="integer">0</m:cn>
       </m:apply>
    </m:apply>
</m:apply>
```

Esempio 12: Conversione dal LaTeXin CMML

Dopo la fase di post-processing il risultato ottenuto è:

```
<apply>
    <eq/>
    <ci>A</ci>
    <apply>
       <conditional-set/>
       <ci>x</ci>
       <apply>
           <gt/>
           <ci>x</ci>
           <cn type="integer">0</cn>
       </apply>
    </apply>
</apply>
```

Esempio 13: Post-processing del CMML

Come è possibile vedere il codice dopo la fase di post-processing è privato del prefisso "*m:*" che precedeva i tag e viene eliminato la specifica del tag "*csymbol*" per rendere omogenea la lettura degli operatori.

2.4.2 Generazione dell'albero linguistico

Una volta ottenuta la rappresentazione della formula in CMML, si procede alla costruzione dell'albero linguistico corrispondente tramite la libreria java SimpleNLG.

Si guardi alla seguente formula $a - \frac{b}{c} \cdot d$:

La sua rappresentazione in CMML è:

```
<apply>
    <minus/>
        <ci>a</ci>
        <apply>
            <times/>
                <apply>
                    <divide/>
                        <ci>b</ci>
                        <ci>c</ci>
                </apply>
                <ci>d</ci>
        </apply>
</apply>
```

Esempio 14: Formula CMML per $a - \frac{b}{c} * d$

L'albero generato corrispondente dell'esempio 14 è:

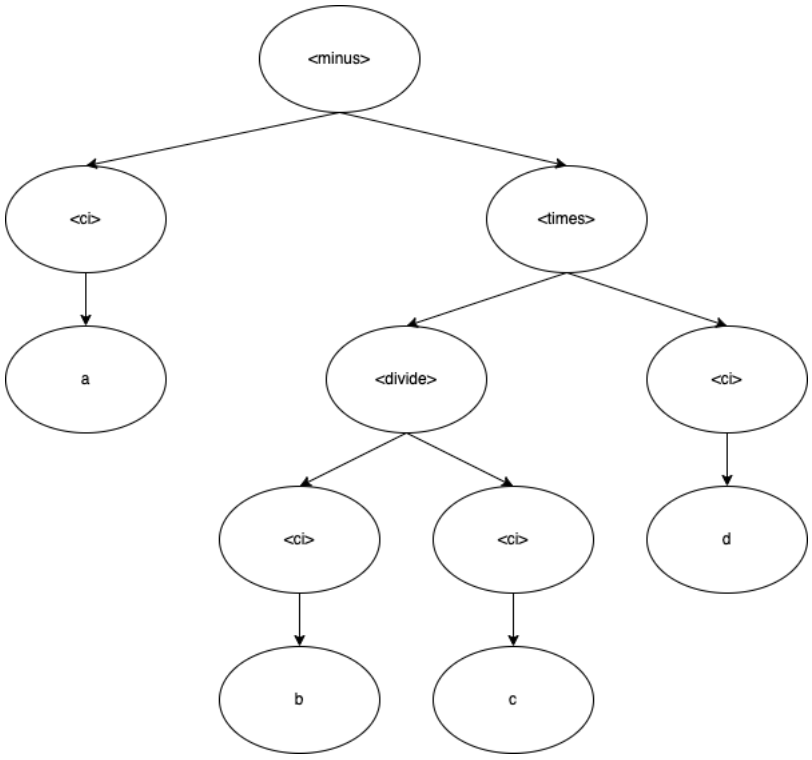

Fig. 2.4: Albero derivato dal CMML per una formula esemplificativa

Come visto già nella sezione 2.3.1 ogni operatore è categorizzato e possiede determinate proprietà linguistiche.

Ai fini di racchiudere tutte le sue caratteristiche si è scelto di costruire dei dizionari, scritti in JSON, da consultare durante la fase generativa.
Si guardi al seguente esempio:

```
{
 "gt": {
    "determiner": false,
    "name": "greater",
    "coordination-conj": "",
    "coordinated-name": "",
    "preposition": "than",
    "pre-modifier": "",
    "post-modifier": "",
    "verb": "be",
    "is-negate": false,
    "is-plural": false,
    "category": "rel-op",
    "label": "major"
}
```

Esempio 15: Dizionario operatore in Jason

Nella prima riga è presente il nome in CMML dell'operatore in questione, "greater". Nelle successive invece, si determinano le sue caratteristiche, nello specifico:

- **Determiner**: indica se l'operatore ha bisogno dell'articolo determinativo. I suoi valori sono *True*, in caso affermativo, *False* altrimenti.
- **Noun**: indica il nome principale dell'operatore. Può essere ad esempio *greater*, *absolute value*, *function*, etc...
- **Coordination-conj**: indica quale congiunzione usare in caso vi sia una coordinazione tra elementi. Ad esempio "or" in "is greather than *or* equal to".
- **Coordinated-name**: indica un eventuale nome coordinato al nome principale.
- **Preposition**: indica la preposizione da legare all'operatore. In questo caso è *than*.
- **Pre-modifier**: modifica le parole che lo seguono nella frase. Generalmente sono aggettivi che sono posti prima dei nomi. Inoltre, anche gli avverbi spesso possono fungere da pre-modificatori.
- **Post-modifier**: modifica le parole che lo precedono nella frase. Generalmente sono avverbi posti dopo i verbi, ma ci sono casi in cui gli aggettivi sono posizionati dopo i nomi.
- **Verb**: indica il verbo che regge l'operatore.

- **Is-negate**: è una feature del verbo che specifica se il verbo deve essere negato o meno.
- **Is-plural**: similmente alla precedente caratteristica, indica se il verbo è coniugato al singolare o al plurale.
- **Category**: specifica la categoria in cui fa parte l'operatore (vd. 2.3.1).
- **Label**: rappresenta il nome dell'operazione legata all'operatore. Ad esempio, nel caso della somma, l'operatore è *plus* mentre l'operazione è *sum*.

2.4.3 Realizzazione del parlato

Questa ultima fase prevede la costruzione di una frase di senso compiuto partendo dall'albero generato in CMML. In particolare, SimpleNLG svolge i seguenti compiti:

1. Ordina i vari sintagmi dell'albero sulla base delle loro dipendenze. Ad esempio, in inglese l'aggettivo precede generalmente un sostantivo;
2. Coniuga i verbi al tempo specificato;
3. Nega i verbi se è specificato;
4. Declina gli aggettivi rispetto al nome a cui sono associati;
5. Declina i nomi rispetto al genere e al numero;
6. Determina gli articoli corretti da usare;

La maggior parte di queste operazioni avvengono autonomamente, tuttavia esistono casi in cui il dizionario su cui si basa la libreria non è in grado ad esempio di effettuare correttamente delle variazioni morfologiche come quando i lessemi prevedono delle forme irregolari. In questi casi, vengono specificate manualmente.

2.5 Sperimentazione

In questa sezione si discute della sperimentazione per il sistema linguistico e di generazione di frasi matematiche. Essendo il lavoro del dott. Monticone speculare a questo sistema per l'italiano, è stato pensato di riproporre una sperimentazione simile, laddove un'analisi dei risultati fra le due può portare all'elicitazione di nuovi elementi.
Per verificare l'efficacia di tale sistema è stato redatto un questionario con Google Form[1] (totalmente accessibile per le persone ipovedenti e cieche) per raccogliere informazioni sulla percepibilità delle formule sintetizzate. Il questionario è diviso in due parti:

- Nella prima vengono poste domande di profilazione quali:
  1. Qual è l'eta;
  2. Se si è madrelingua inglesi;
  3. Se si ha una disabilità visiva;
  4. Quanto bene si conosce la matematica e l'analisi matematica;
  5. Qual è il grado di istruzione;
  6. Se si ha un diploma di laurea, specifica se è relativo a discipline matematiche;
- Nella seconda parte invece, viene chiesto di ascoltare 10 formule, recuperate dal libro di analisi I di Pandolfi [14], sintetizzate secondo delle strategie di parentesizzazione diverse e per ciascuna rispondere a delle domande. Più nello specifico viene chiesto di:
  1. Riscrivere la formula ascoltata usando una notazione non ambigua;
  2. Rispondere a una domanda a scelta multipla in cui si chiede di valutare la facilità di comprensione della formula su una scala di Likert a sette valori;

  Le 10 formule appartengono a due categorie principali: facili e difficili. La difficoltà di una formula è legata al numero di nodi generato dal corrispondente albero in CMML e dal numero di livelli di parentesi presenti (parentesizzazione).

[1] Il questionario è consultabile tramite il seguente link: Speech synthesis of math formulas - https://forms.gle/Eccy4deKeHYtrYg56

Formula LaTeX	**Nodi**
$A \times B = \{(x, y) \mid x \in A, y \in B\}$	15
$g^{-1}(y) = f^{-1}\left((y - b)/a\right)$	13
$\int_b^c a \, \mathrm{d}x = a(c - b)$	14
$x > b \Longrightarrow \lvert f(x)\rvert < M$	10
$\sqrt[n]{x} = x^{1/n}$	10

Tab. 2.21: Formule facili - 1° Sperimentazione

Formula LaTeX	**Nodi**
$\lim\limits_{x \to x_0} \left\{ \dfrac{f(x) - f(x_0)}{x - x_0} - f'(x_0) \right\} = 0$	31
$y = f(a) + \dfrac{f(b) - f(a)}{b - a}(x - a)$	21
$\int \dfrac{1}{\sqrt{m^2 - x^2}} \mathrm{d}x = \arcsin \dfrac{x}{m} + c$	20
$\sum\limits_{k=0}^{n} \dfrac{f^{(k)}(x_0)}{k!}(x - x_0)^k$	28
$\lim \left(1 + \dfrac{1}{n}\right)^n = e$	10

Tab. 2.22: Formule difficili - 1° Sperimentazione

Più in particolare, le formule che non necessitano parentesi e che hanno un numero di nodi inferiore a 15 sono classificate come facili. Il numero "15" è il valore di riferimento che corrisponde al numero di nodi della definizione del prodotto cartesiano relativa alla prima formula della tabella 2.21.
Le formule che invece, hanno un alto numero di nodi e necessitano di parentesizzazione vengono classificate come difficili, eccezion fatta per l'ultima formula della tabella 2.22 che però, prevede due livelli annidati di parentesizzazione.

2.5.1 Risultati

La fase di sperimentazione è al momento della stesura della tesi ancora aperta.
Come già scritto nella precedente sezione sono state scelte 10 formule, 5 facili e 5 difficili. Le formule sono state proposte con sintetizzatori vocali diversi e aggregate in

modi differenti.
Le strategie di aggregazione delle formule sono in totale 3:

1. **pause**: le parentesi delle formule sono sostituite con delle pause dalla durata di 500ms;

2. **parenthesis**: come da nome, i simboli matematici sono inseriti all'interno di parentesi;

3. **smart**: si basa sulla precedente strategia e si sostituiscono le parentesi più interne con delle pause.

Per ogni differente versione (sintetizzatore e strategia di aggregazione) le formule sono state presentate in modo altrettanto diverso, rinominando i simboli di funzione e parametri (Ad esempio $f(x)$ in una, $g(y)$ nell'altra).
I sintetizzatori usati sono AWS Polly e eSpeak (vedi sez. 3.2).
Per ogni formula si riporta la percentuale di persone che l'hanno trascritta correttamente, mentre in caso contrario sarà indicato il tipo di errore:

- errore di parentesizzazione: è stata percepita una parentesizzazione della formula diversa da quella attesa. Si indica con la lettera P.

- errore simbolico: l'errore riguarda la comprensione dei simboli pronunciati. Si indica con la lettera S.

Utenti tester Una parentesi necessaria per l'analisi e la spiegazione dei risultati è il numero di tester che hanno preso parte alla sperimentazione e il loro profilo.
Come già scritto in precedenza, la fase di sperimentazione è ancora aperta e conta ad oggi due utenti ciechi che hanno effettuato il test.
Entrambi i tester sono di madrelingua italiana e hanno riportato di conoscere bene la matematica e di avere una laurea affine alle discipline scientifiche.

Risultati per le formule facili

Di seguito si riportano delle tabelle che mostrano i risultati della sperimentazione in cui si sono fatte ascoltare le formule facili con due sintetizzatori diversi ma, con lo stesso metodo di aggregazione. In particolare, rispetto alle tabelle mostrate: la colonna "Aggr." indica la strategia di aggregazione usata, "Sint." indica il tipo di sintetizzatore usato, "Corr. %" indica la percentuale di persone che ha compreso correttamente la formula e "Tipo Err." indica il tipo di errore commesso: S errore simbolico, P errore di parentesizzazione.

Formula LaTeX	**Aggr.**	**Sint.**	**Corr. %**	**Tipo Err.**
$A \times B = \{(x, y) \mid x \in A, y \in B$	smart	polly	100	-
$g^{-1}(y) = f^{-1}\left((y - b)/a\right)$	smart	polly	100	-
$\int_b^c d \, \mathrm{d}x = d(c - b)$	smart	polly	100	-
$x > b \Longrightarrow \lvert f(x)\rvert < M$	smart	polly	100	-
$\sqrt[n]{x} = x^{1/n}$	smart	polly	100	-

Tab. 2.23: Risultati per formule facili pt.1 - 1° sperimentazione

Formula LaTeX	**Aggr.**	**Sint.**	**Corr. %**	**Tipo Err.**
$A \times B = \{(x, y) \mid x \in A, y \in B$	smart	espeak	100	-
$g^{-1}(y) = f^{-1}\left((y - b)/a\right)$	smart	espeak	100	-
$\int_b^c d \, \mathrm{d}x = d(c - b)$	smart	espeak	100	-
$x > b \Longrightarrow \lvert f(x)\rvert < M$	smart	espeak	100	-
$\sqrt[n]{x} = x^{1/n}$	smart	espeak	100	-

Tab. 2.24: Risultati per formule facili pt.2 - 1° sperimentazione

Risultati per le formule difficili

In maniera analoga a quanto descritto nel precedente paragrafo vengono mostrati qui i risultati per le formule difficili.

Formula LaTeX	Aggr.	Sint.	Corr. %	Tipo Err.
$\lim_{x\to x_0}\left\{\frac{f(x)-f(x_0)}{x-x_0}-f'(x_0)\right\}=0$	smart	polly	100	-
$y=f(c)+\frac{f(d)-f(c)}{d-c}(x-c)$	smart	polly	100	-
$\int\frac{1}{\sqrt{k^2-x^2}}\mathrm{d}x=\arcsin\frac{x}{k}+c$	smart	polly	100	-
$\sum_{k=0}^{n}\frac{g^{(k)}(x_0)}{k!}(x-x_0)^k$	smart	polly	100	-
$\lim\left(1+\frac{1}{n}\right)^n=e$	smart	polly	50	P

Tab. 2.25: Risultati per formule difficili pt.1 - 1° sperimentazione

Osservazione

- Il 50% degli utenti ha recepito la quinta formula per la tabella 2.25 come:

$$\lim\left(1+\left(\frac{1}{n}\right)^n\right)=e$$

Formula LaTeX	Aggr.	Sint.	Corr. %	Tipo Err.
$\lim_{z\to z_0}\left\{\frac{g(z)-g(z_0)}{z-z_0}-g'(z_0)\right\}=0$	par	polly	100	-
$y=g(b)+\frac{g(c)-g(b)}{c-b}(z-b)$	par	polly	100	-
$\int\frac{1}{\sqrt{s^2-y^2}}\mathrm{d}y=\arcsin\frac{y}{s}+c$	par	polly	100	-
$\sum_{n=0}^{l}\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$	par	polly	100	-
$\lim\left(1+\frac{1}{x}\right)^x=e$	par	polly	50	P

Tab. 2.26: Risultati per formule difficili pt.2 - 1° sperimentazione

Osservazione

- Il 50% degli utenti ha recepito la quinta formula per la tabella 2.25 come:

$$\lim\left(1+\left(\frac{1}{n}\right)^n\right)=e$$

Formula LaTeX	Aggr.	Sint.	Corr. %	Tipo Err.
$\lim_{y \to y_0} \left\{ \frac{h(y) - h(y_0)}{y - y_0} - h'(y_0) \right\} = 0$	pause	polly	100	-
$y = h(s) + \frac{h(t) - h(s)}{t - s}(z - s)$	pause	polly	100	-
$\int \frac{1}{\sqrt{l^2 - x^2}} \mathrm{d}x = \arcsin \frac{x}{l} + c$	pause	polly	100	-
$\sum_{k=0}^{n} \frac{h^{(k)}(y_0)}{k!}(y - y_0)^k$	pause	polly	100	-
$\lim \left(1 + \frac{1}{k}\right)^k = e$	pause	polly	100	-

Tab. 2.27: Risultati per formule difficili pt.3 - 1° sperimentazione

2.5.2 Analisi

I risultati ottenuti mostrano un evidente successo nella comprensione delle frasi matematiche. Si sottolinea come effettivamente i dati ottenuti siano sensibili a variazioni essendo il numero di tester molto piccolo. Inoltre, seppur i tester siano di madre lingua italiana, hanno fatto studi matematici: ciò ha permesso loro di ottenere risultati molto positivi.
Volendo fare un confronto con i risultati ottenuti dal lavoro del dott. Monticone si presentano evidenti differenze. A tal proposito si riportano delle tabelle che raggruppano i dati su entrambe le tipologie di formule e che riportano dei punteggi basati su due diverse metriche.
La prima è Exact Match e restituisce il valore 1 (o 0) se l'albero CMML iniziale e quello percepito sono uguali (o meno).
La seconda è il punteggio SPICE, una misura di somiglianza usata nell'ambito di *automatic caption generation*. Il punteggio SPICE si ottiene calcolando l' F-score della sovrapposizione tra i due alberi CMML: la sovrapposizione si misura decomponendo gli alberi in sotto-strutture elementari tipizzate (operandi, operatori e le loro relazioni). Ad esempio, l'espressione $x - 1$ è scomposta come $1, x, minus, (op : minus, first : x), (op : minus, second : 1)$ [15].

Utente	Tot. formule (25)	F. facili (10)	F. difficili (15)
1	23	10	13
2	25	10	15
Tot:	48(96%)	20(100%)	28(93%)

Tab. 2.28: Exact Match - Risultati

Utente	**Tot. formule (25)**	**F. facili (7)**	**F. difficili (18)**
1	13	5	8
2	18	5	13
3	13	5	9
4	14	5	10
Tot:	58(58%)	20(71%)	38(54%)

Tab. 2.29: Exact Match - Risultati dott. Monticone

Utente	**Tot. formule (25)**	**F. facili (10)**	**F. difficili (15)**
1	0.98	1.0	0.97
2	1.0	1.0	1.0
AVG:	0.99	1.0	0.985

Tab. 2.30: SPICE media - Risultati

Utente	**Tot. formule (25)**	**F. facili (7)**	**F. difficili (18)**
1	0.92	0.95	0.91
2	0.96	0.98	0.97
3	0.93	0.90	0.94
4	0.95	0.97	0.94
AVG:	0.94	0.95	0.94

Tab. 2.31: SPICE media - Risultati lavoro dott. Monticone

Prima di commentare le tabelle sovrastanti si precisa che la modalità con cui è avvenuta la sperimentazione del dott. Monticone, le formule ascoltate dai suoi sperimentatori e le strategie di aggregazione usate sono le medesime. Gli unici fattori di disparità sono da ritrovarsi in uno dei sintetizzatori vocale usati (IBM Watson [16] invece di AWS Polly) e il numero di formule facili e difficili. Su quest'ultimo punto si precisa che seppur il numero sia diverso, le formule presentano la stessa struttura con simboli diversi. Es: $f(x)$ diventa g(y).
I risultati del dott.Monticone mostrano chiaramente come la complessità delle formule sia un fattore determinante la comprensione delle stesse. In minor modo ma pur sempre presente, è possibile tracciare questo parallelismo con i risultati ottenuti da questa sperimentazione.
Il motivo invece, della disparità dei risultati è da ritrovarsi nel campione di tester per i due diversi sistemi: per il lavoro precedente solo un utente su quattro aveva una laurea affine a studi matematici rispetto alla totalità del campione che ha effettuato questo test.

Capitolo 3

Analisi e design di un sistema di dialogo per le espressioni matematiche

Il sistema di dialogo è composto, come già descritto nella sezione 1.2.3, da tre parti distinte: Text-To-Speech, Speech Recognition e Command Management.
In questo capitolo si esamineranno le scelte prese, le tecnologie utilizzate e come è stato realizzato.

3.1 Caratteristiche del sistema di dialogo

L'introduzione di un sistema di dialogo si deve alla volontà di migliorare la recepibilità e la comprensione delle espressioni matematiche. La sua progettazione ha dovuto tenere in considerazione sia gli obiettivi di progetto sia l'esigenza di avere degli standard sufficientemente buoni di *human-computer interaction*.

3.1.1 Caratteristiche funzionali

Tra le caratteristiche funzionali principali del sistema troviamo:

1. Poter essere interrotto in qualsiasi momento tramite input vocale;
2. Capire e analizzare le richieste utente;
3. Rispondere a determinate domande utente;

Si ricorda che il sistema è pensato innanzitutto per un'utenza di persone ipovedenti e cieche che potrebbero trovare in questo strumento un mezzo per comprendere meglio e più velocemente le formule matematiche.
Rispetto ai lettori di schermo generalmente adoperati, come NVDA [17], che prevedono la digitalizzazione di specifiche combinazioni di tasti il sistema permette di interrompersi in qualsiasi punto mediante comandi vocali. Inoltre, mentre i lettori di schermo permettono di muoversi avanti e indietro nel testo, con questo progetto si è cercato di spingere verso nuove funzionalità per l'utente come la possibilità di ricevere

richieste di ripetizione da specifiche parti di espressioni o domandare come è composta la stessa. Logicamente queste funzionalità sono elaborate proprio partendo dai punti 2. e 3. delle caratteristiche del sistema sopra elencate.

3.1.2 Voice Design

Parallelamente a queste è stato tenuto conto anche dell'aspetto interattivo del sistema. In generale, nello sviluppo delle interfacce utente è necessario non sottovalutare lo sforzo che l'utente fa per completare un compito o per rispondere a una richiesta. In termini tecnici questo prende il nome di "sovraccarico cognitivo" [18].
Ad esempio, il progettista del sistema non deve assillare l'utente con informazioni o opzioni non finalizzate al suo obiettivo.
Quando si costruiscono interfacce grafiche, i progettisti devono creare schermate intuitive, facili da usare e che abbiano elementi di riconoscimento (i.e. icone familiari). Nello sviluppo delle interfacce vocali, tuttavia, i metodi tradizionali per evitare il sovraccarico cognitivo dell'utente non sono più validi. Mentre con le prime l'utente rimane concentrato sulla pagina, leggendo il testo o navigando, il pattern di attenzione con le VUI (Vocal User Interface) è differente: l'utente pronuncia un comando e il sistema risponde immediatamente. Questo significa che la persona deve essere concentrata durante l'interazione e che può essere meno attenta in altri momenti [19].

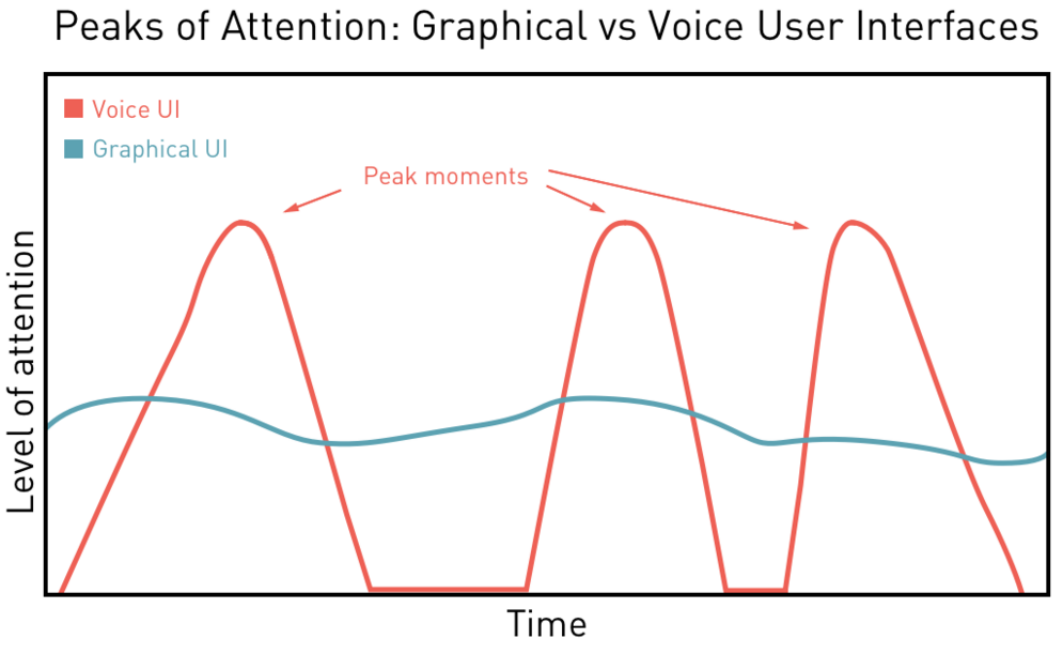

Fig. 3.1: Confronto tra picchi di attenzione tra GUI e VUI

Come è possibile vedere dal grafico, i picchi di attenzione sono brevi e ciò significa che bisogna limitare la lunghezza e la densità delle risposte. Il modo migliore per

gestire ed evitare i cali di attenzione è prevenire il sovraccarico cognitivo rispettando le massime conversazionali di Grice [20]. Le massime sono quattro:

- **The maxim of quantity** (massima della quantità): a una persona viene fornita l'informazione strettamente necessaria;
- **The maxim of quality** (massima della qualità): l'informazione data ad una persona è vera;
- **The maxim of relation** (massima della relazione): a una persona vengono offerte informazioni utili e pertinenti alla discussione;
- **The maxim of manner** (massima del modo): si tenta di essere chiari non risultando ambigui;

L'insieme di queste massime forma il "principio cooperativo", importante per la progettazione delle interfacce vocali.

Paradigmi di Interazione

Nello sviluppo dell'interfaccia per l'interazione con l'utente è stato necessario seguire degli standard per creare un'interfaccia usabile, dove il termine "usabilità" trova una sua definizione nella norma ISO 9241-11: "è il grado in cui un prodotto può essere usato da particolari utenti per raggiungere certi obiettivi con efficacia, efficienza, soddisfazione in uno specifico contesto d'uso" intendendo:

- Efficacia come precisione e completezza con cui gli utenti raggiungono specifici obiettivi;
- Efficienza come risorse impiegate in relazione alla precisione e completezza con cui gli utenti raggiungono specifici obiettivi;
- Soddisfazione come libertà dal disagio e attitudine positiva con cui gli utenti raggiungono specifici obiettivi attraverso l'uso del prodotto.

3.2 Text-to-Speech

Per Text-to-Speech (TTS) si intende una tecnologia che converte un testo in output vocale. Grazie a grandi database contenenti molte espressioni parlate è possibile allenare dei sistemi a riprodurre sonorità simili a quelle umane. Le voci di questi sistemi sono simili a suoni naturali e reagiscono a tono, pronuncia e frequenza.
La sintesi vocale garantisce accessibilità alle persone che non sono in grado di leggere a causa di problemi di alfabetizzazione o di disabilità, offrendo un modo alternativo per ottenere informazioni.

3.2.1 Architettura dei sistemi TTS

I sistemi di Text-to-Speech si basano su una architettura encoder-decoder [1] e si definiscono essere *speaker-dependent*: sono addestrati per avere una voce coerente di unico speaker [21].
Per esempio, il corpus vocale LJ (set di dati di pubblico dominio) consiste di 13.100 brevi clip audio di un singolo relatore che legge brani di 7 libri di cui viene fornita una trascrizione per ogni clip. Le clip variano in lunghezza da 1 a 10 secondi e hanno una durata totale di circa 24 ore [22].

Generalmente il sistema è diviso in due componenti:

1. un modello encoder-deconder per la previsione dello spettrogramma: si mappano stringhe di lettere in spettogrammi mel, ossia uno spettrogramma in cui le frequenze vengono convertite nella scala mel. Partendo dalla stringa: *It's time for lunch!* otteniamo:

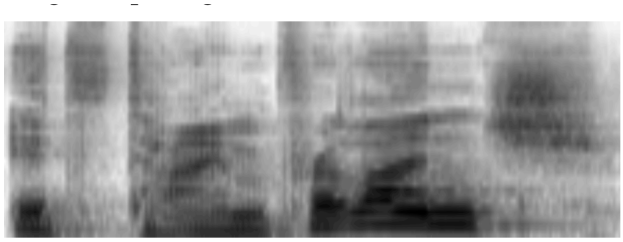

Fig. 3.2: Esempio di spettrogramma mel

2. il secondo, invece, mappa gli spettrogrammi mel in forme d'onda. Questo processo si chiama vocoding; il modulo, vocoder:

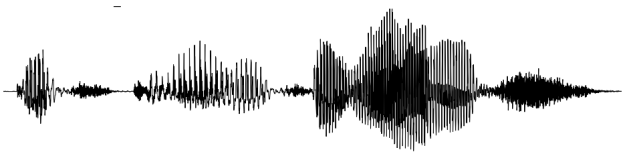

Fig. 3.3: Esempio di waveform

3.2.2 AWS Polly e eSpeak

I sintetizzatori usati per dare voce al sistema sono AWS Polly e eSpeak, due TTS basati su tecnologie diverse:

[1] L'architettura encoder-decoder è usata in molti ambiti, dal Natural Language Processing alla Computer Vision. Si tratta di due reti neurali che operano a cascata: la prima codifica l'input; la seconda cerca di ricostruire l'input partendo dall'output dall'encoder.

- **AWS Polly**: utilizza le tecnologie avanzate di deep learning per sintetizzare discorsi naturali simili a quelli umani. Oltre alle voci standard TTS, Amazon Polly offre voci Text-to-Speech neurali (NTTS) che forniscono miglioramenti avanzati nella qualità del parlato [23];
- **eSpeak**: è un sintetizzatore vocale software open source e a differenza delle tecnologie basate su machine learning, utilizza un metodo di "sintesi formante". Il parlato risulta chiaro e può essere utilizzato ad alta velocità, ma non è così naturale o fluido come i sintetizzatori che si basano su registrazioni vocali umane [24]. Tra le alcune delle sue caratteristiche troviamo:
  - Include diverse voci, le cui caratteristiche possono essere modificate;
  - Può produrre output vocale come file WAV;
  - Supporta l'SSML;
  - Ha delle dimensioni compatte. Il programma e i suoi dati, comprese molte lingue, ammontano a circa 2 Mbyte.

L'utilizzo di eSpeak accanto a quello di AWS Polly, seppur meno fluido e naturale del secondo, è dovuto all'abitudine che le persone cieche e ipovedenti hanno nell'ascoltare voci del genere, essendo solo particolarmente recente la diffusione di tecnologie basate su ML.
L'applicazione nella sua realizzazione permette quindi, di poter scegliere quale sintetizzatore usare per ascoltare il sistema e interagirci.

3.3 Speech-to-Text

Lo Speech-to-Text è una tecnologia di riconoscimento vocale che consente di riconoscere e trasformare la lingua parlata in testo attraverso tecniche di linguistica computazionale. In particolare, un programma del genere si basa su algoritmi linguistici per ordinare i segnali uditivi dalle parole pronunciate e trasferirli nel testo utilizzando caratteri Unicode. La conversione del parlato in testo si basa su un modello di apprendimento automatico complesso che prevede diversi passaggi [25]:

- La tecnologia Speech-to-Text funziona captando le vibrazioni prodotte dai suoni pronunciati e traducendole in un linguaggio digitale attraverso un convertitore (da analogico a digitale);
- Il convertitore analogico-digitale prende i suoni da un file audio, misura le onde in grande dettaglio e le filtra per distinguere i suoni rilevanti;
- I suoni vengono quindi segmentati in centesimi o millesimi di secondo e quindi abbinati a fonemi. Un fonema è un'unità di suono che distingue una parola da un'altra in una determinata lingua. Ad esempio, ci sono circa 40 fonemi in lingua inglese;

- I fonemi vengono poi, processati in una rete basata su un modello matematico che li confronta con parole e frasi ben note;
- L'output viene, infine, presentato come testo o una richiesta sulla base della versione più probabile dell'audio;

3.3.1 Architettura dei sistemi STT

La figura mostrata di seguito delinea l'architettura codificatore-decodificatore standard, comunemente denominata *attention-based encoder decoder* o *AED*, or *listen attend and spell (LAS)*, usata per i sistemi STT [26].

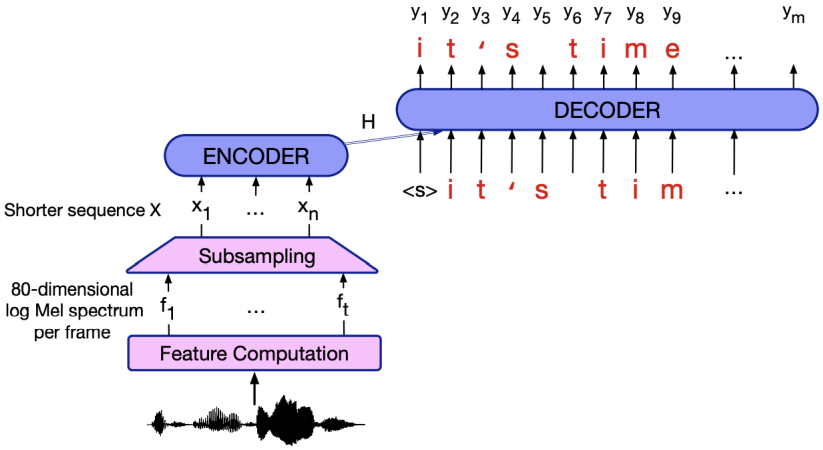

Fig. 3.4: Architettura encoder-decoder per sistemi STT

L'input è una sequenza di t vettori di caratteristiche acustiche $F = f_1, f_2, ..., f_t$, per frame di $10ms$.
L'output può essere costituito da lettere o frammenti di parole (in questo caso lettere). Quindi la sequenza di output sarà $Y = (< SOS >, y_1, ..., y_m < EOS >)$, assumendo che i token speciali di inizio e fine sequenza siano <sos> e <eos>, mentre ogni y_i è un carattere; per l'inglese potremmo scegliere il set:

$$y_i \in \{a, b, c, ..., z, 0, ..., 9, < space >, < comma >, < period >, < apostrophe >, < unk >\}$$

Naturalmente l'architettura encoder-decodificatore è particolarmente appropriata quando le sequenze di input e output hanno forti differenze di lunghezza, come accade per il parlato, con sequenze di caratteristiche acustiche molto lunghe mappate su sequenze di lettere molto più brevi o parole.

3.3.2 AWS Transcribe e AWS S3

Nella prima fase implementativa del progetto è stata usata la tecnologia STT per ascoltare e comprendere le richieste utente. In questa sezione, si tratterà di AWS Transcribe, un servizio cloud messo a disposizione da Amazon che permette la conversione

automatica in testo [27].
Accanto ad AWS Trascribe è stato necessario usare anche AWS S3 [28], ossia uno storage online in cui i dati sono archiviati come oggetti all'interno di risorse chiamate "bucket". L'iter realizzativo per sfruttare questa tecnologia è:

1. Registrare l'audio dell'utente;
2. Salvare l'audio in un file temporaneo sulla macchina;
3. Caricare questo su di un bucket di AWS S3;
4. Importare l'audio dal bucket in AWS Transcribe;
5. Aspettare l'elaborazione della traduzione;
6. Cancellare le copie della registrazione audio;

Si premette che l'intera operazione si basa sul modello asincrono di AWS Transcribe, che al momento della scrittura del codice era gratuito con seppur diverse limitazioni. Tra queste limitazioni, tuttavia, la più restrittiva per l'accessibilità è da ritrovarsi nelle tempistiche di traduzione.
Di seguito è presente una tabella che mostra i suddetti tempi (mediati per 5 iterazioni di successo) per tipiche frasi di interazione con il sistema:

Frase pronunciata	**Tempo di pronuncia della frase**	**Tempo totale**
Hey stop	3.35s	8.96s (5.61s)
Repeat from x	7.68s	15.86s (8.18s)
What is the first integral?	11.16s	22.21s (11.65s)

Tab. 3.1: Tempi di traduzione STT - AWS Polly

La seconda colonna della tabella 3.1 mostra il tempo impiegato per pronunciare la frase nella prima colonna, mentre nella terza sono presenti i secondi totali che comprendono il tempo di pronuncia più il tempo necessario per la traduzione della frase. Nelle parentesi tonde è presente il tempo che intercorre dai tempi parziali della seconda colonna.
Come è possibile evincere le tempistiche per la traduzione sono leggermente superiori a quelli di pronuncia, con dei tempi che vanno via via diminuendo quanto più è lunga la frase pronunciata.
Il vincolare l'utente ad aspettare nel caso di una frase come "Hey stop" per quasi 6s si è mostrato nella fase di testing essere molto snervate e poco efficiente.
Nelle successive sezioni, si presenteranno delle alternative che hanno permesso di abbattere completamente queste tempistiche.

3.4 Wake Word Detection

Le considerazioni fatte nella precedente sezione mostrano ampiamente come in un sistema che deve soddisfare certi standard di accessibilità ed efficienza, usare sistemi STT

asincroni risulta poco efficiente.
È stato pensato di dividere quindi, il problema in due parti. La prima è riuscire a riconoscere una parola, che chiameremo "wake up word" per permettere al sistema di ascoltare direttamente le richieste utente; la seconda è proprio riconoscere determinate richieste con un sistema di Speech-to-Intent (3.5).
Quando si usano sistemi di dialogo moderni come Alexa, Siri o Google Assistant, gli utenti interagiscono innanzitutto con determinate parole di attivazione.
Sono quindi, sistemi di classificazione binaria che permettono di capire se l'utente voglia o meno interagire con l'assistente vocale in quel momento. Rispetto ai sistemi di Keyword spotting (KWS) per lo Speech Processing, i sistemi di *wake word detection* presentano diverse caratteristiche [29]:

- la wake up word è una parola predefinita che viene fissata durante l'addestramento e il rilevamento;
- durante l'ascolto, un sistema di *wake word detection* è eseguito in streaming online, ossia non attendiamo che l'intero flusso audio venga elaborato per determinare la sua presenza, ma un trigger viene attivato e segnala l'inizio di una richiesta;
- l'attività di *wake word detection* consuma maggiori risorse di calcolo e memoria;

3.4.1 Porcupine

Il sistema di *wake word detection* usato è Porcupine, sviluppato da Picovoice. La scelta di questo strumento è stata dettata sia da un primo studio dei benchmark che l'azienda proponeva rispetto ad altri sia da una facilità implementativa.
Di seguito viene riassunto il risultato dell'esecuzione di diversi wake word detector su sei diverse parole chiave. Il grafico seguente mostra il tasso di errore di diversi motori con 1 falso allarme ogni 10 ore. Più basso è il tasso di errore, più preciso è il motore.

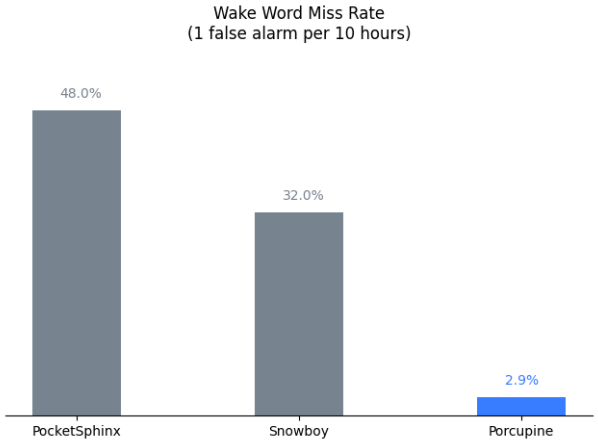

Fig. 3.5: Benchmark per wake word detection [30]

L'accuratezza di questi sistemi è stata calcolata per varie wake up word (alexa, computer, jarvis, smart mirror, snowboy, e view glass) facendo per ciascuna di queste ascoltare un insieme di file audio che potevano o meno includerle. Ad esempio, potevano essere audio con musica, suoni e rumori in cui non venivano pronunciate.

3.5 Speech-to-Intent

Nella ricerca di strumenti più veloci nella traduzione del parlato si è rivolta l'attenzione ai sistemi di Speech-to-Intent.
Un intento categorizza l'intenzione di un utente, laddove una combinazione di intenti permette di definire una conversazione.
Sistemi del genere lavorano su un ristretto vocabolario di termini e il sistema abbina la richiesta utente ad una intenzione. Questa operazione prende il nome di *intent classification*.
I sistemi di Speech-to-Intent sono in genere allenati con tecnologie di AI per classificare l'intento. Rispetto ai motori Speech-to-Text che analizzano "meccanicamente" ciò che viene ascoltato, quelli Speech-to-Intent cercano di carpirne l'intento, ciò cosa si sta cercando di dire.

3.5.1 Rhino

Per questo progetto si è fatto uso di Rhino, un motore di Speech-to-Intent sviluppato da Picovoice [31], il quale permette di dedurre direttamente l'intento dai comandi vocali all'interno di un certo contesto in real time.
La scelta di questo strumento è data sia dalla possibilità di poter esplicitare il contesto matematico sia dai risultati in termini di accuratezza e velocità della traduzione. Si mostrano di seguito dei grafici di confronto di prestazioni tra Rhino e altri sistemi di Natural Language Understanding (NLU) in cloud [32]:

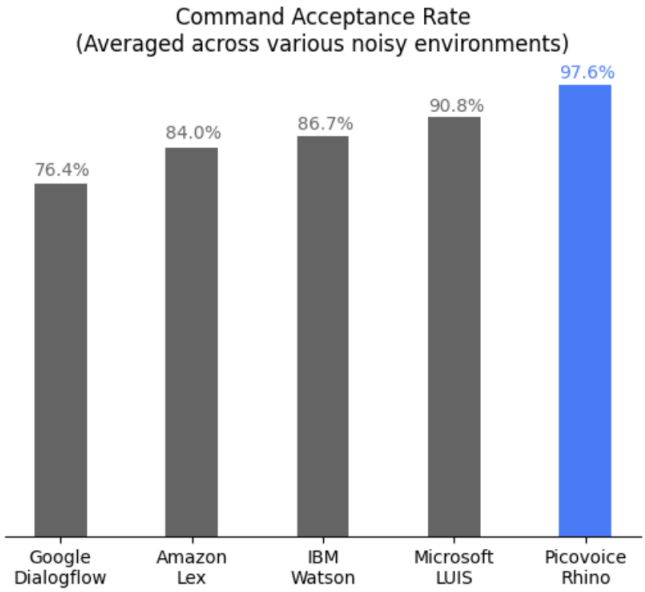

Fig. 3.6: Tasso di command acceptance

Il tasso di command acceptance è la probabilità che un motore capisca correttamente il comando vocale.

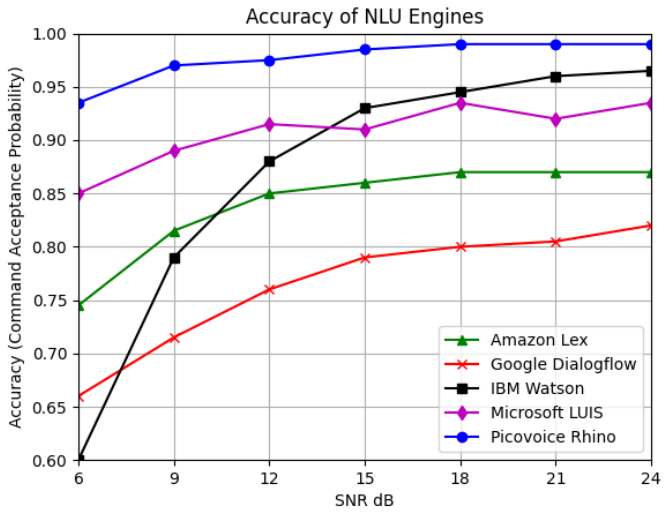

Fig. 3.7: Accuracy dei motori NLU

I dati sul parlato provengono da più di 50 relatori unici. Ognuno ha contribuito con una decina di espressioni diverse e collettivamente ci sono 619 comandi utilizzati in questo benchmark. Inoltre, i motori sono stati testati in condizioni rumorose per simulare situazioni tipiche nel mondo reale [33].

Costruire un modello con Rhino

Il primo passo per costruire un modello STI con Rhino è definire un contesto, ossia un insieme di espressioni, di intenti e gli argomenti degli intenti.
Ad esempio un intento per questa applicazione può essere l'intenzione dell'utente di far ripetere l'espressione matematica da un certo punto della frase. Quindi si crea come mostrato nelle immagini di seguito l'intento "RepetitionFrom" e si aggiungono le varie espressioni che possono definirlo:

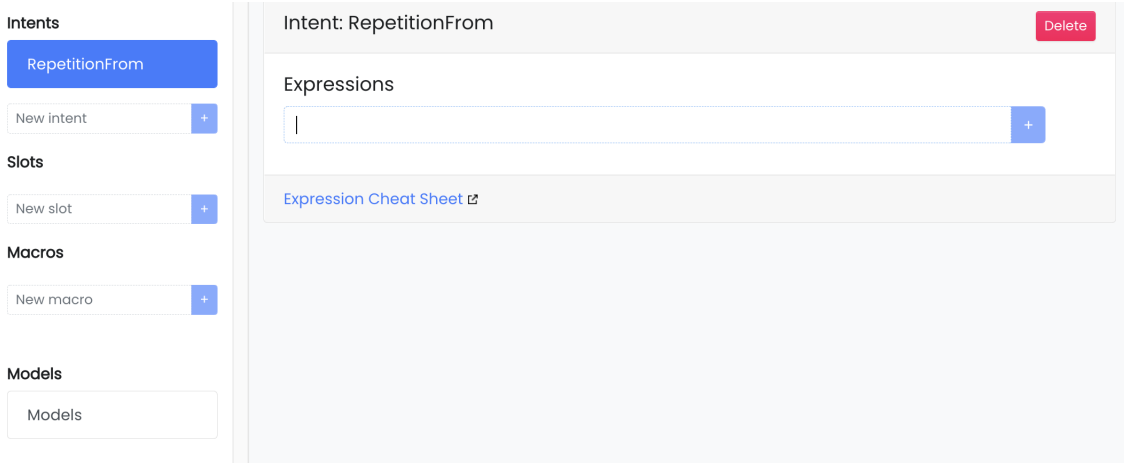

Fig. 3.8: Creazione dell'intent "Repeatition from"

Il passo successivo come già scritto è scrivere delle espressione che possano rappresentare possibili richieste utente:

Fig. 3.9: Caratterizzazione dell'intent "Repeatition from"

Dove:

- $[repeat\ from,\ say\ again\ from]$ esprimono due modi diversi che l'utente può usare per iniziare la richiesta;
- (the) rappresenta una parola opzionale;
- $\$command$ è uno slot dell'intent;

Più nel dettaglio, viene creato uno slot dal nome $*command* che viene definito da degli "elements", ossia variabili che possono trovarsi nell'espressione che stiamo definendo:

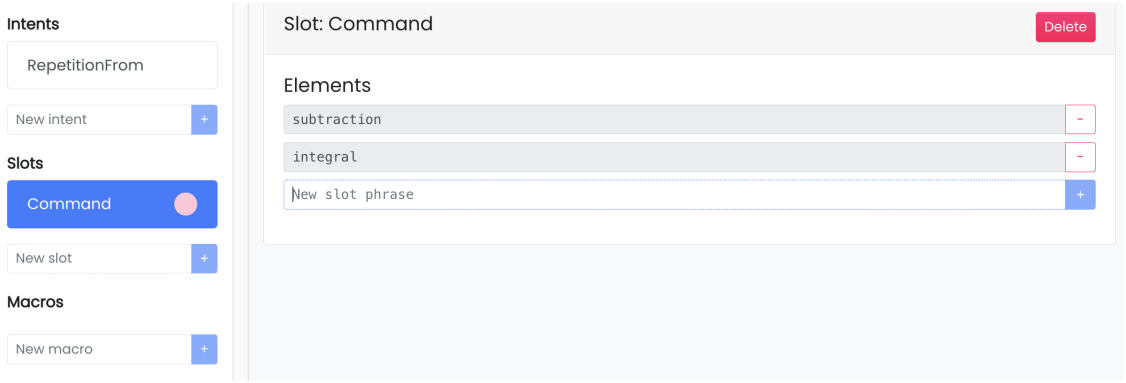

Fig. 3.10: Argomento dell'intent

Nella figura 3.10 lo slot può essere occupato dalla parola *subtraction* o *integral*. L'ultimo passo è l'addestramento del modello che viene fatto in automatico e che produrrà un motore capace di capire espressioni complesse come "*Repeat from the integral*".

3.6 Command Management

L'ultimo tassello che completa il quadro del progetto riguarda la gestione dei comandi che l'utente ha a disposizione. Dal momento in cui il sistema si mette in ascolto delle richieste utente con l'attivazione del modulo di Speech-to-Intent, l'utente può porre due tipi di domande:

1. *Repeat from ...*
2. *What is ...*

Il primo comando permette di ripetere un operatore o una variabile presente nell'espressione matematica e che è stata già pronunciata.
Si guardi al seguente esempio:

y is equal to h of s plus h of t minus h of s over t minus s times z minus s

Esempio 16: Esempio di frase matematica da far ripetere

Si immagini che il sintetizzatore vocale abbia pronunciato fino ad un certo momento la sotto-frase:

y is equal to h of s plus h of t minus h of s over t

È possibile quindi fare richieste come "Repeat from y", laddove il sistema dopo aver analizzato la frase darà in output tutta l'espressione essendo "y" la prima parola della frase, oppure ancora "Repeat from the second s". In quest'ultimo caso, il sistema andrà a ricercare la seconda occorrenza della variabile "s" e risponderà con "s over t". Il secondo comando invece permette di analizzare nel dettaglio parti dell'espressione matematica. Ogni operatore, ricordando la sezione 2.3.1, presenta caratteristiche specifiche dettate dalla sua arietà e dai suoi argomenti, e per ciascuno quindi, è possibile richiedere da quali sia costituito.
Di seguito è presente una tabella che descrive per ogni categoria o operatore specifico quali elementi lo contraddistinguono:

Categoria operatori	**Elementi caratteristici**
operatori relazionali	arg. destro | arg. sinistro
operatori algebrici, aritmetici e insiemistici	arg. destro | arg. sinistro
connettivi logici	arg. destro | arg. sinistro
insieme condizionale	arg. | proprietà
coppia	primo arg. | secondo arg.
ausiliare	ausiliare
sequenze	arg. | l. inf. | l. sup. | variabile
funzioni elementali	arg.
calcolus	arg. | arg. inf. | arg. sup. | funzione

Tab. 3.2: Elementi caratteristici delle categorie degli operatori

Si presentano ora diversi esempi:

the integral from b to c of d d x is equal to d times c minus b

Esempio 17: Esempio di frase matematica da analizzare

Delle possibili domande possono essere:

1. What is the low limit of integral?
2. What is the argument of integral?
3. What is the left argument of minus?
4. What is the left argument of subtraction?

Le cui risposte corrispondenti sono:

1. b
2. d
3. b

4. b

Si noti come la terza e la quarta domanda siano equivalenti. Questo perché l'utente può domandare basandosi o sul nome dell'operazione, in questo caso la sottrazione, o sul simbolo dell'operatore che ha sentito durante la pronuncia, "minus".
Infine l'ultimo comando che ha a disposizione l'utente è quello di ripresa. Nel caso in cui l'utente interrompa la pronuncia dell'espressione matematica, può richiedere di continuarla dicendo "Go on" oppure "Let's resume".

3.7 Implementazione del sistema di dialogo

Nella seguente sezione si discute dell'implementazione del sistema di dialogo. Così come raccontato nella sezione 2.4 anche per questa parte è stato usato lo stesso IDE e gli stessi linguaggi di programmazione, Java e Clojure.
Tuttavia, il modulo di Speech Recognition è stato sviluppato in Python [34]. Per rispettare le tempistiche del progetto, si è preferito costituire il modulo in modo che potesse comunicare con la restante parte dell'applicazione. Esso funziona come un server, sviluppato con il framework Flask [35], che ascolta durante tutta l'esecuzione del programma per riconoscere eventuali comandi utente e li comunica al client.

3.7.1 Algoritmo principale del sistema di dialogo

Si mostra di seguito lo pseudo-codice del sistema di dialogo:

```
dialogue():
1.  Say "I'm starting to say the sentence"
2.  Activate the Wake Word Detection system
3.  For each word within the sentence:
4.      Say the word
5.      If the Wake-up Wor Detection system has detected "Ehy stop":
6.          Say "Ok, I'm listening to you"
7.          Activate the Request Recognition system
8.          Fulfill the request
9.  Say "I've finished reading this sentence, but I'm still here for you"
10. Activate the Request Recognition system
```

Con la riga 1 e 9 si avverte l'utente quando sta incominciando o è terminata la sintesi dell'espressione matematica.
Si precisa che i messaggi di segnalazione come "Ok, I'm listening to you" sono pronunciati con un tono di voce diverso da quello con cui è pronunciata normalmente.
Il motivo è da ritrovarsi nella semplicità di accorgersi di un suono diverso e che evidenzia che l'uno è un messaggio di avviso, l'altro parole matematiche.
Nel caso in cui il sistema venga eseguito con AWS Polly, sono usate due voci femminili diverse; alternativamente, con eSpeak viene modificato il pitch della voce, rendendola più acuta.

Con la riga 2 si attiva il sistema di Wake Word Detection. Più nel dettaglio, si avvia un thread parallelo che esegue una chiamata al server che si occupa di riconoscere la wake up word. Nel momento in cui viene riconosciuta, viene segnalato al sistema che nella riga 6 ferma l'esecuzione della sintesi vocale e si mette in ascolto.
Infatti nella riga 6 viene pronunciato il messaggio con cui si comunica che l'utente può fare delle richieste e con le righe 7 e 8, si ascoltano queste e vengono soddisfatte.

Infine, la riga 10 permette al sistema di ricevere ulteriori richieste al termine della pronuncia della frase matematica, senza che sia pronunciata la wake up word, per dare all'utente ulteriori possibilità di interrogare il sistema.

3.7.2 Algoritmo di gestione delle richieste

La riga 8. dell'algoritmo presente nella sezione 3.7.1 inquadra solo genericamente come vengono soddisfatte le richieste utente.
Di seguito, si mostra lo pseudo-codice per soddisfarle:

```
fulfill_request(intent, request):
1.  Parsify the request
2.  If the intent is "Questioning":
3.      Search for the operator in the request
4.      Match the request with the operator arguments
5.      If the match is successful:
6.          Answer the question
7.          Otherwise, utter an apology message
8.  If the intent is "RepetitionFrom":
9.      Search within the sentence uttered what the user
                                    wants to hear repeated
10.     If the search is successful:
11.         Repeat from that point
12.         Otherwise, utter an apology message
13. If the intent is "Resume":
14.     Resume the synthesis of the mathematical expression
```

La prima operazione che viene fatta è controllare il tipo di intent. Nelle righe 2., 8. e 13. viene fatto questo verificando che l'intenzione sia una tra il domandare, il ripetere o la ripresa della sintesi.

Nel primo caso, "Questioning", viene ricercato l'operatore presente nella richiesta e successivamente una corrispondenza con gli operatori dell'espressione matematica. Inoltre, se nella richiesta utente, accanto all'operatore è presente un argomento, viene ricercato proprio quell'argomento specifico.

Nel secondo caso, "Repetition from", viene ricercato all'interno dell'espressione matematica fino al momento pronunciata la variabile o l'operatore, in caso connotati da un indice di occorrenza nella frase.

Infine, l'ultimo caso è quello che prevede la riprese della sintesi vocale.

3.7.3 Modello di Speech-to-Intent

L'ultimo aspetto da considerare per l'implementazione del sistema di dialogo è la costruzione di un modello che possa riconoscere le richieste utente come raccontato nella sezione 3.5.
Il primo passo è quindi definire degli intent e delle espressioni che le rappresentano tramite la console di Rhino [36]:

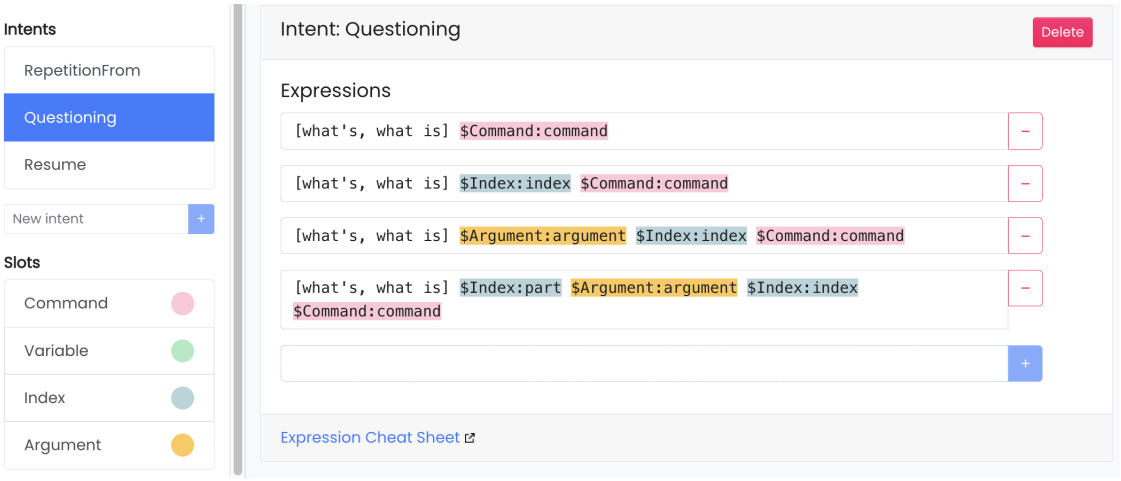

Fig. 3.11: Intent per Questioning

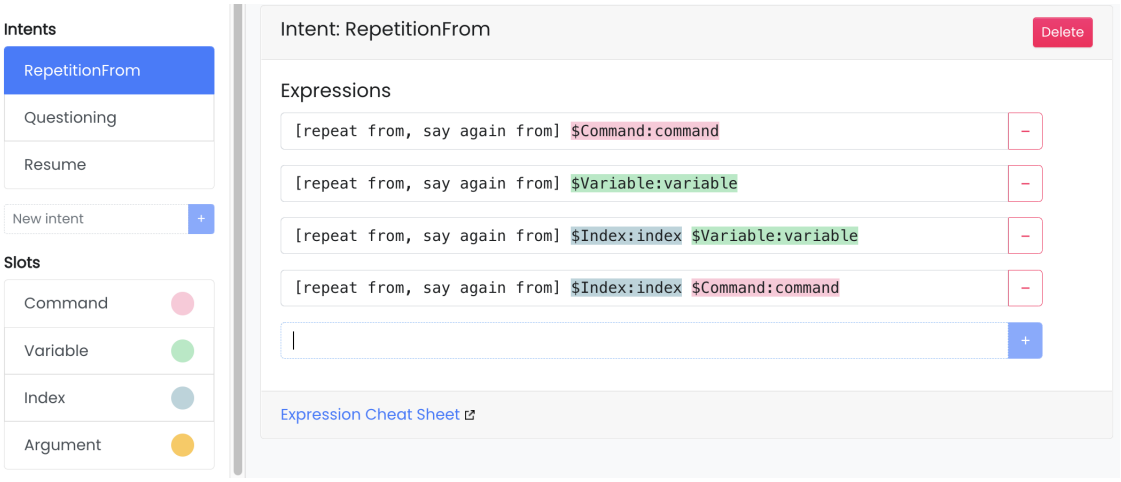

Fig. 3.12: Intent per RepetitionFrom

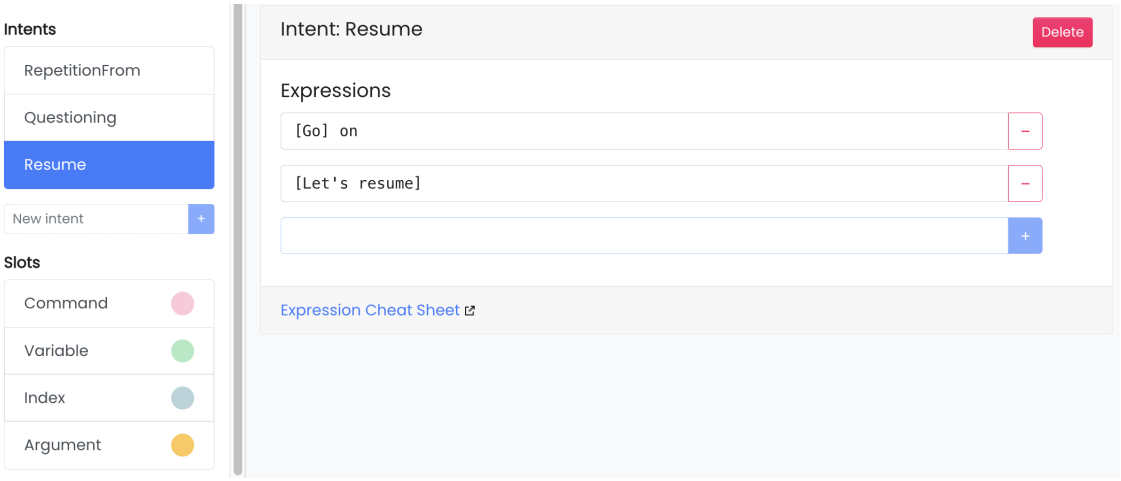

Fig. 3.13: Intent per Resume

Ciò che è presente nelle parentesi quadre per tutte le immagini rappresenta i diversi modi con il quale iniziare la richiesta (ossia si può dire sia "Repeat from x" che "Say again from x" per attivare l'intento), successivamente sono presenti per la figure 3.11 e 3.12 delle combinazioni di slot che delineano varie possibilità di composizioni di domande.
Ad esempio gli slot sono così definiti:

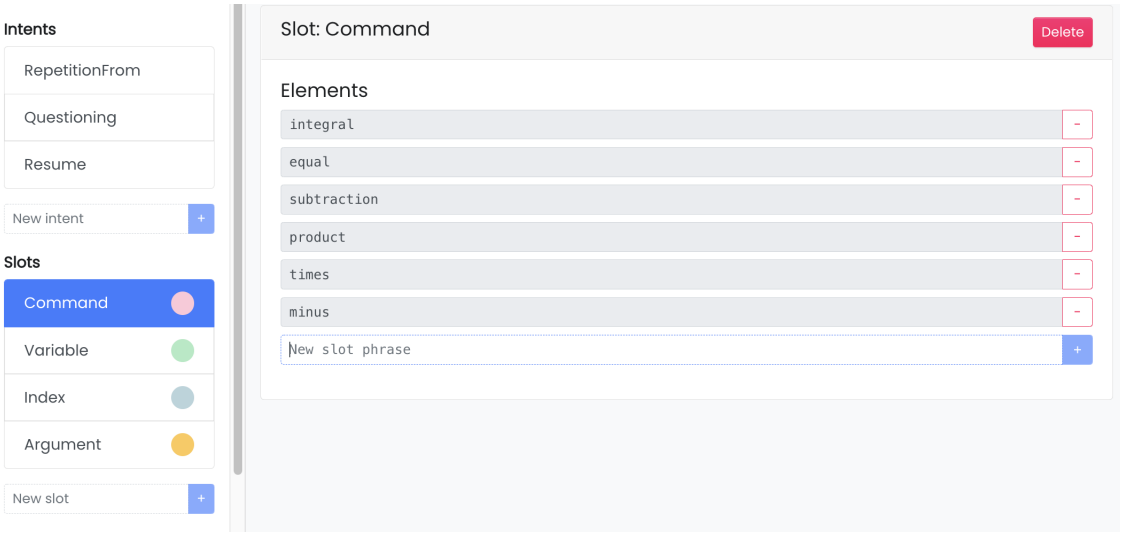

Fig. 3.14: Slot per operatori

Fig. 3.15: Slot per variabili

Fig. 3.16: Slot per indici

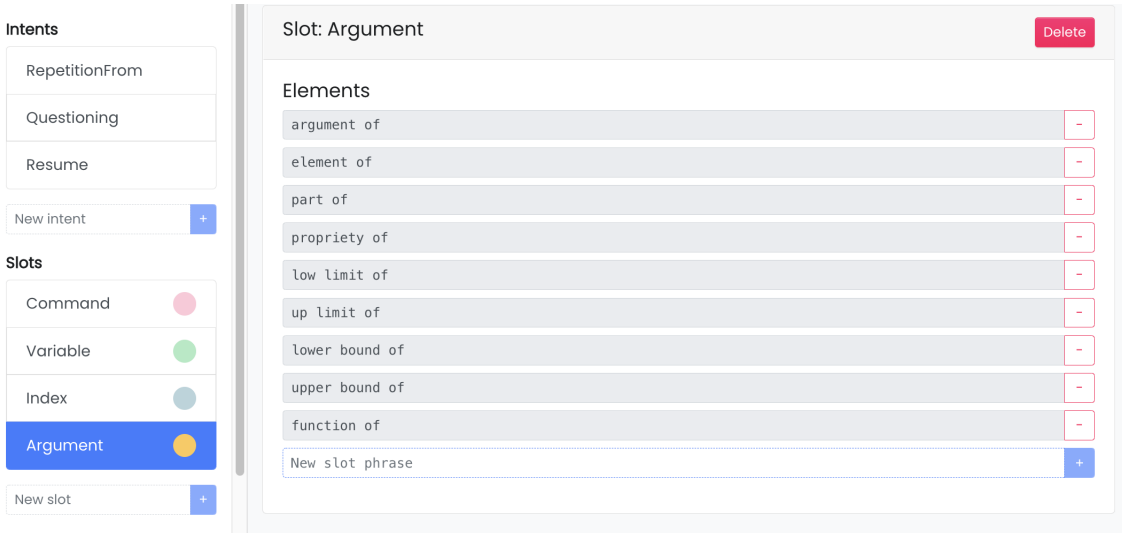

Fig. 3.17: Slot per argomenti

Pertanto delle possibili richieste possono essere:

- Repeat from x
- Say again from the second integral
- What is the first product?
- What is the left argument of the subtraction?
- Go on

Esempio 18: Possibili richieste utente

3.8 Sperimentazione

La seconda parte della sperimentazione verte a testare non solo l'efficacia del sistema di dialogo, ma anche la sua usabilità. Gli utenti tester hanno quindi, risposto sia a delle richieste per verificare l'utilità dello strumento sia a delle domande per valutare la User Experience.
Per questa fase di test sono stati reclutati cinque tester totalmente ciechi e di madrelingua italiana.
Rispetto alla prima parte della sperimentazione che poteva esser svolta in totale autonomia dagli utenti, per questa è stato necessario prima allenare i tester con esempi e poi, fatto sperimentare il sistema.
Il compito di ciascun tester è stato riscrivere in un qualsiasi formalismo privo di ambiguità l'espressione matematica che veniva ascoltata. Naturalmente, essendo una sperimentazione interattiva, potevano interagire con il sistema e fare delle domande per migliorare la comprensione della formula che stava venendo pronunciata.
Prima di discutere delle domande poste agli utenti, è necessario descrivere la fase di training:

1. Per prima cosa è stato fatto ascoltare il seguente testo che descrive il funzionamento dell'applicazione tramite il sintetizzatore vocale AWS Polly affinché l'utente si potesse abituare ad ascoltare sin da subito questa voce che sarebbe stata la stessa utilizzata dal sistema durante tutto il test:

   First of all, thank you for taking part in this experimentation.
   During today's test you will have to interact with a voice assistant who will pronounce 6 mathematical phrases to you. Your job is to write them in any understandable formalism, even in English, and send me your answers by the Google Meet chat or by email.
   The voice assistant is able to receive various commands when interrupted by saying ''Hey stop!''.
   I recommend you spell these words well and use a high volume of voice.
   The commands are:
   1. Repeat from anywhere in the sentence. For example, consider the expression ''a + b'': if ''Say again from plus'' is said, the assistant will repeat ''+ b''. The keyword is ''Say again from'' and you can repeat a variable or an operator.

   2. Specify an operator. For example consider the expression ''a + b + c'' you can ask ''What is the first sum?'' or ''What is the left argument of second sum''. The keyword is ''What is'' and you can ask for information about a variable or an operator.
   Regarding the operator you can ask how you have already heard to repeat the argument in its case, specifying whether left or right, lower or upper.

```
3. In the event that you accidentally interrupted the assistant
or do not remember the question you wanted to ask him,
just say ‘‘Go on’’.

If you try to stop the assistant without success, please repeat
‘‘Hey stop!’’ several times. From the moment it stops you have about
1 minute maximum to make the request.
Let’s take a few examples now, if you don’t have any questions!
```

2. Successivamente l'organizzatore si proponeva prima per dare una dimostrazione all'utente e poi, farlo esercitare su interazioni di prova eventualmente chiarendo i suoi dubbi e rispondendo alle sue domande.

Come per la prima sperimentazione (vedi sez. 2.5), anche in questo caso sono state fatte ascoltare diverse formule tra quelle facili e quelle difficili. Più nel dettaglio, sono state scelte 6 formule (3 facili e 3 difficili): queste sono quelle che hanno ottenuto un risultato peggiore nella comprensione guardando all'analisi dei risultati della prima sperimentazione e del lavoro del dott. Monticone.

Formula LaTeX	**Nodi**
$g^{-1}(y) = f^{-1}\left((y-b)/a\right)$	13
$\int_b^c a \, \mathrm{d}x = a(c-b)$	14
$\sqrt[n]{x} = x^{1/n}$	10

Tab. 3.3: Formule facili - 2° Sperimentazione

Formula LaTeX	**Nodi**
$\lim_{x \to x_0} \left\{ \dfrac{f(x) - f(x_0)}{x - x_0} - f'(x_0) \right\} = 0$	31
$y = f(a) + \dfrac{f(b) - f(a)}{b - a}(x - a)$	21
$\int \dfrac{1}{\sqrt{m^2 - x^2}} \mathrm{d}x = \arcsin \dfrac{x}{m} + c$	20

Tab. 3.4: Formule difficili - 2° Sperimentazione

Infine, al termine della sperimentazione agli utenti sono state poste sia domande di profilazione come le seguenti, sia lo User Exceperience Questionnaire [37].

Domande di profilazione:

1. Qual è l'eta;
2. Se si è madrelingua inglesi;
3. Se si ha una disabilità visiva;
4. Quanto bene si conosce la matematica e l'analisi matematica;
5. Qual è il grado di istruzione;
6. Se si ha un diploma di laurea, specifica se è relativo a discipline matematiche;

	1	2	3	4	5	6	7		
annoying	○	○	○	○	○	○	○	enjoyable	1
not understandable	○	○	○	○	○	○	○	understandable	2
creative	○	○	○	○	○	○	○	dull	3
easy to learn	○	○	○	○	○	○	○	difficult to learn	4
valuable	○	○	○	○	○	○	○	inferior	5
boring	○	○	○	○	○	○	○	exciting	6
not interesting	○	○	○	○	○	○	○	interesting	7
unpredictable	○	○	○	○	○	○	○	predictable	8
fast	○	○	○	○	○	○	○	slow	9
inventive	○	○	○	○	○	○	○	conventional	10
obstructive	○	○	○	○	○	○	○	supportive	11
good	○	○	○	○	○	○	○	bad	12
complicated	○	○	○	○	○	○	○	easy	13
unlikable	○	○	○	○	○	○	○	pleasing	14
usual	○	○	○	○	○	○	○	leading edge	15
unpleasant	○	○	○	○	○	○	○	pleasant	16
secure	○	○	○	○	○	○	○	not secure	17
motivating	○	○	○	○	○	○	○	demotivating	18
meets expectations	○	○	○	○	○	○	○	does not meet expectations	19
inefficient	○	○	○	○	○	○	○	efficient	20
clear	○	○	○	○	○	○	○	confusing	21
impractical	○	○	○	○	○	○	○	practical	22
organized	○	○	○	○	○	○	○	cluttered	23
attractive	○	○	○	○	○	○	○	unattractive	24
friendly	○	○	○	○	○	○	○	unfriendly	25
conservative	○	○	○	○	○	○	○	innovative	26

Fig. 3.18: User Experience Questionnaire

Lo User Experience Questionnaire mostrato nella figura 3.18 permette di analizzare l'impressione generale riguardo l'esperienza utente. Vengono misurati sia gli aspetti

classici di usabilità, come l'efficienza, la perspicuità e l'affidabilità, sia aspetti di UE come originalità e stimolazione.

3.8.1 Risultati

Così come per la prima sperimentazione, anche questa al momento della scrittura della tesi è ancora aperta.
Rispetto alla prima che poteva essere svolta in totale autonomia dagli utenti, questa ha previsto la presenza di una figura che facilitasse e osservasse l'interazione dei tester. Si premette prima di mostrare i risultati, che a causa delle norme vigenti attualmente per il Covid-19, le interazioni sono avvenute in maniera telematica tramite video chiamate su Google Meet e che la figura dell'osservatore e del facilitatore [2] sono interpretate dalla stessa persona.

Utenti tester I tester che hanno preso parte alla sperimentazione sono cinque. Tutti hanno disabilità visiva e sono di madrelingua italiana, inoltre hanno conseguito tutti una laurea triennale e il 60% ha una laurea affine alle discipline scientifiche. Nel dettaglio risulta che:

- un 20% afferma di non conoscere bene la matematica;
- un altro 20% afferma di conoscere perfettamente la matematica;
- il rimanente 60% la conosce bene;

Risultati per le formule facili e difficili

Di seguito si riportano delle tabelle che mostrano i risultati della sperimentazione in cui si sono fatte ascoltare le formule facili e difficili con il sintetizzatore vocale AWS Polly e lo stesso metodo di aggregazione, smart.
Per ogni formula si riporta la percentuale di persone che l'hanno trascritta correttamente, mentre in caso contrario sarà indicato il tipo di errore:

- errore di parentesizzazione: è stata percepita una parentesizzazione della formula diversa da quella attesa. Si indica con la lettera P
- errore simbolico: l'errore riguarda la comprensione dei simboli pronunciati. Si indica con la lettera S.

In particolare, rispetto alle tabelle mostrate: la colonna "Aggr." indica la strategia di aggregazione usata, "Sint." indica il tipo di sintetizzatore usato, "Corr. %" indica la percentuale di persone che ha compreso correttamente la formula e "Tipo Err." indica il tipo di errore commesso: S errore simbolico, P errore di parentesizzazione.

[2] Generalmente un osservatore ha il compito di osservare i comportamenti del tester, le sue difficoltà e le modalità con cui le risolve; il facilitatore ha il compito di facilitare l'interazione con il tester nel caso in cui quest'ultimo si trovi in una situazione di stallo, oppure abbia qualche difficoltà.

Formula LaTeX	**Aggr.**	**Sint.**	**Corr. %**	**Tipo Err.**
$g^{-1}(y) = f^{-1}\left((y-b)/a\right)$	smart	polly	20	S, P
$\int_b^c d\,\mathrm{d}x = d(c-b)$	smart	polly	40	S, P
$\sqrt[n]{x} = x^{1/n}$	smart	polly	60	S, P

Tab. 3.5: Risultati per formule facili - 2° sperimentazione

Osservazioni

- Il 20% degli utenti ha recepito la prima formula per la tabella 3.5 come:

$$1 + y = \frac{1}{f(y-b)} \cdot a$$

- Un altro 20% degli utenti ha percepito la stessa formula come:

$$\frac{\frac{1}{j}}{i} = f(i/j)$$

- il 20% degli utenti ha capito la seconda formula essere:

$$\int_b^c \mathrm{d}x = d \cdot (c-a)$$

Formula LaTeX	**Aggr.**	**Sint.**	**Corr. %**	**Tipo Err.**
$\lim_{x \to x_0} \left\{ \frac{f(x) - f(x_0)}{x - x_0} - f'(x_0) \right\} = 0$	smart	polly	60	S
$y = f(c) + \frac{f(d) - f(c)}{d - c}(x - c)$	smart	polly	40	S, P
$\int \frac{1}{\sqrt{k^2 - x^2}} \mathrm{d}x = \arcsin \frac{x}{k} + c$	smart	polly	60	S, P

Tab. 3.6: Risultati per formule difficili - 2° sperimentazione

Osservazioni

- Il 20% degli utenti ha recepito la seconda formula per la tabella 3.6 come:

$$y = f(s) + h(t) - f(\frac{s}{t}) - s \cdot (z - s)$$

- Il 20% degli utenti ha capito la terza formula essere:

$$\int \frac{1}{\sqrt{k^2 - x^2}} \mathrm{d}y = \arcsin \frac{x}{k + c}$$

Risultati per la UX

Si premette che il questionario somministrato non produce un punteggio complessivo sull'esperienza dell'utente. A causa della sua costruzione non ha senso ottenere un punteggio così globale (ad esempio calcolando la media su tutte le scale), perché questo valore non potrebbe essere interpretato correttamente.
I punteggi quindi, sono relativi ad *attrazione*, *perspicuità*, *efficienza*, *affidabilità* e *stimolazione*.
Inoltre, le risposte proposte in questa sezione tengono conto solo di quelle considerate consistenti, ossia che non siano risultate conflittuali fra loro. Sui cinque tester che hanno sostenuto la sperimentazione, sono state considerate solo le risposte di tre di questi.
Il grafico sotto riportato elenca i valori dei singoli elementi al fine di consentire di rilevare valori anomali nelle valutazioni.
Un elemento che mostra grandi deviazioni rispetto alle valutazioni degli altri elementi della stessa scala, può suggerire che l'articolo è interpretato erroneamente (ad esempio, a causa di un contesto speciale nella valutazione) da un numero maggiore di partecipanti. I valori compresi tra -0,8 e 0,8 rappresentano una valutazione più o meno neutrale della scala corrispondente; i valori $>$ 0,8 rappresentano una valutazione positiva e quelli $<$ 0,8 rappresentano una negativa. L'intervallo delle scale è compreso tra -3 (estremamente negativo) e +3 (estremamente buono).

Item	Mean	Variance	Std. Dev.	No.	Left	Right	Scale
1	0,7	2,3	1,5	3	annoying	enjoyable	Attractiveness
2	0,7	0,3	0,6	3	not understandable	understandable	Perspicuity
3	1,0	1,0	1,0	3	creative	dull	Novelty
4	2,0	1,0	1,0	3	easy to learn	difficult to learn	Perspicuity
5	2,3	0,3	0,6	3	valuable	inferior	Stimulation
6	1,0	1,0	1,0	3	boring	exciting	Stimulation
7	2,3	0,3	0,6	3	not interesting	interesting	Stimulation
8	0,3	1,3	1,2	3	unpredictable	predictable	Dependability
9	0,0	9,0	3,0	3	fast	slow	Efficiency
10	2,0	1,0	1,0	3	inventive	conventional	Novelty
11	2,3	0,3	0,6	3	obstructive	supportive	Dependability
12	2,3	0,3	0,6	3	good	bad	Attractiveness
13	1,0	1,0	1,0	3	complicated	easy	Perspicuity
14	1,7	0,3	0,6	3	unlikable	pleasing	Attractiveness
15	2,0	1,0	1,0	3	usual	leading edge	Novelty
16	2,3	0,3	0,6	3	unpleasant	pleasant	Attractiveness
17	1,3	1,3	1,2	3	secure	not secure	Dependability
18	1,3	4,3	2,1	3	motivating	demotivating	Stimulation
19	1,3	1,3	1,2	3	meets expectations	does not meet expectations	Dependability
20	1,7	0,3	0,6	3	inefficient	efficient	Efficiency
21	1,3	0,3	0,6	3	clear	confusing	Perspicuity
22	2,3	0,3	0,6	3	impractical	practical	Efficiency
23	1,3	0,3	0,6	3	organized	cluttered	Efficiency
24	2,3	0,3	0,6	3	attractive	unattractive	Attractiveness
25	1,7	0,3	0,6	3	friendly	unfriendly	Attractiveness
26	2,7	0,3	0,6	3	conservative	innovative	Novelty

Fig. 3.19: UEQ Risultati

UEQ Scales (Mean and Variance)		
Attractiveness	1,833	0,03
Perspicuity	1,250	0,25
Efficiency	1,333	0,58
Dependability	1,333	0,02
Stimulation	1,750	0,44
Novelty	1,917	0,15

Fig. 3.20: UEQ Scala

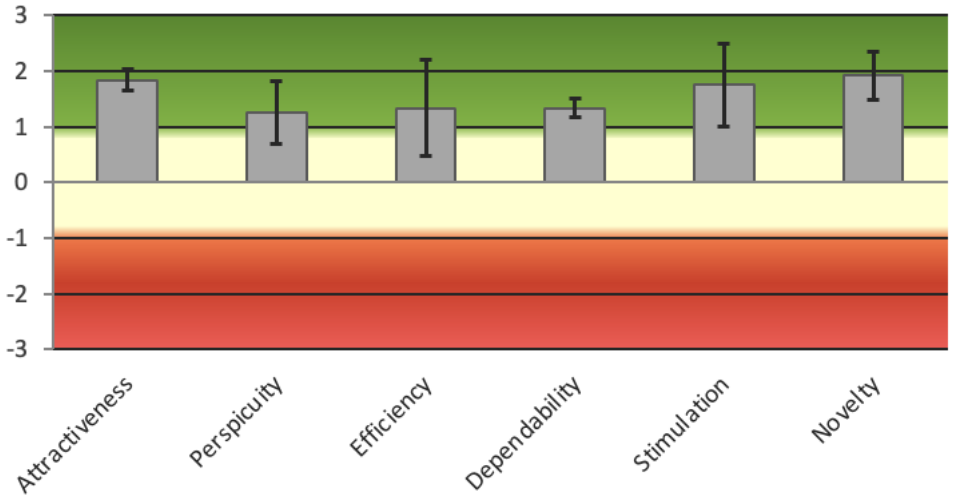

Fig. 3.21: UEQ Scala - Grafico

I valori più a sinistra della figura 3.19 sono indice delle ventisei domande poste nel questionario. Questi sono mediati per ciascuna scala di riferimento e sono presentati numericamente nella figura 3.20 e tramite istogramma nella figura 3.21.
Di seguito invece, si mostra il grafico che raggruppa la media per ogni risposta degli utenti. Il colore per ciascun Item indica una diversa scala di appartenenza.

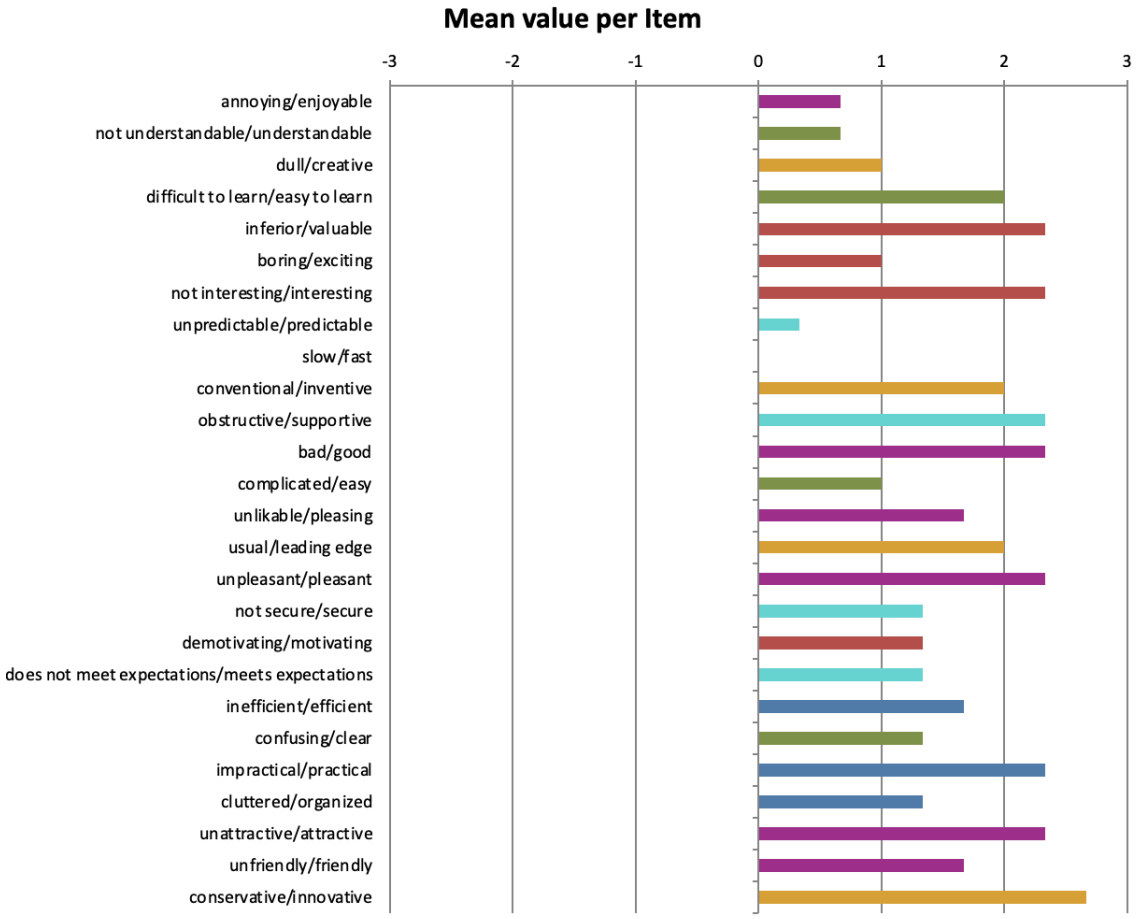

Fig. 3.22: UEQ Valori Medi per Item

Nella figura 3.23 vengono mostrati gli intervalli di confidenza del 5% per le medie della scala e le medie dei singoli elementi.
L'intervallo di confidenza è una misura per la precisione della stima della media della scala. Minore è l'intervallo di confidenza, maggiore è la precisione della stima e più ci si può fidare dei risultati. L'ampiezza dell'intervallo di confidenza dipende dal numero di dati disponibili e dalla coerenza con cui le persone hanno giudicato il prodotto valutato. Più coerente è la loro opinione, minore è l'intervallo di confidenza.

Confidence intervals (p=0.05) per scale						
Scale	**Mean**	**Std. Dev.**	**N**	**Confidence**	**Confidence interval**	
Attractiveness	1,833	0,167	3	0,189	1,645	2,022
Perspicuity	1,250	0,500	3	0,566	0,684	1,816
Efficiency	1,333	0,764	3	0,864	0,469	2,198
Dependability	1,333	0,144	3	0,163	1,170	1,497
Stimulation	1,750	0,661	3	0,748	1,002	2,498
Novelty	1,917	0,382	3	0,432	1,485	2,349

Fig. 3.23: UEQ Intervallo di confidenza

3.8.2 Analisi

I risultati di questa sperimentazione sono complessi da analizzare rispetto alla molteplicità degli elementi che la contraddistinguono: è importante non solo guardare ai dati numerici, ma anche all'incisività dell'assistente vocale e all'esperienza utente. A tal proposito si possono tracciare varie considerazioni prima di analizzare nel dettaglio i risultati:

1. **I profili dei tester**: Il livello di conoscenza della matematica e dell'analisi matematica è un fattore particolarmente determinante: se nella prima sperimentazione tutti i tester avevano fatto degli studi legati alle discipline scientifiche, in questa solo il 60% ne ha fatti.

2. **La famigliarità dello strumento**: dal lavoro del dott. Monticone e dai risultati della prima sperimentazione si è determinato che le formule più facili siano più comprensibili rispetto a quelle più difficili. Tuttavia, in questa sperimentazione risulta che le formule difficili siano state comprese quanto le formule più facili. Subentra in questi termini la familiarità dello strumento usato. Nella prima sperimentazione i tester potevano ascoltare le frasi matematiche tramite un video caricato su Youtube: non solo è possibile riascoltare la traccia audio quante volte lo si desidera, ma è una modalità di fruizione a cui sono abituati. Al contrario, questa applicazione risulta totalmente nuova per loro e usabile solo dopo alcune iterazioni. Infatti, i risultati e le performance migliorano più gli utenti imparano ad usarla.

3. **User Experience**: dalle risposte dello User Experience Questionnarie si evince come lo strumento sia stato ampiamente apprezzato e che costituisca effettivamente un mezzo efficace ed efficiente.

Ed è proprio dalla User Experience che inizia l'analisi dei risultati. A tal proposito si riporta un grafico che riassume il comportamento del sistema rispetto alle sei scale dell'UEQ relative all'attrazione, perspicuità, efficienza, affidabilità e stimolazione.
La figura 3.24 riporta i valori medi misurati posti in relazione a valori di un set di dati di riferimento preesistente. Questo set di dati contiene risultati di 21175 persone provenienti da 468 studi riguardanti diversi prodotti (software aziendale, pagine Web, negozi Web, social network).
Dal confronto dei risultati per il prodotto valutato con i dati del benchmark è possibile trarre conclusioni sulla qualità relativa del prodotto rispetto ad altri.

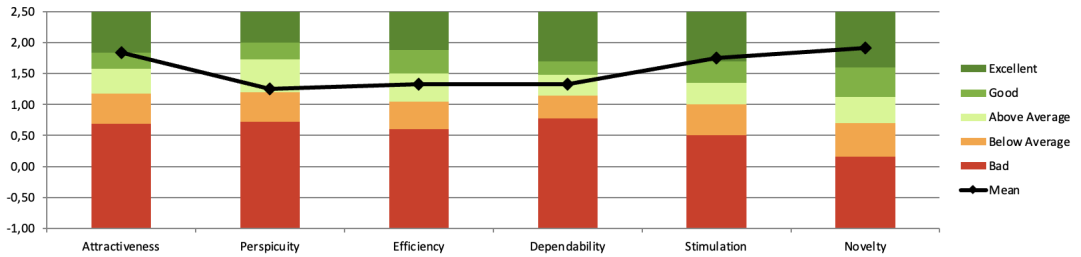

Fig. 3.24: UEQ Benchmark

In particolare, si sono ottenuti punteggi molto alti per quanto riguarda l'attrazione, la stimolazione e la novità del sistema.

Passando all'analisi dei risultati per la comprensione delle formule vengono riportate qui delle tabelle che mostrano i risultati per Exact Match e SPICE (vedi 2.5.2) similmente a quanto fatto per la precedente fase della sperimentazione.

Utente	**Tot. formule (6)**	**F. facili (3)**	**F. difficili (3)**
1	3	2	1
2	6	3	3
3	2	1	2
4	2	0	2
Tot:	13(54.2%)	6(50%)	8(66.7%)

Tab. 3.7: Exact Match - Risultati

Utente	**Tot. formule (6)**	**F. facili (3)**	**F. difficili (3)**
1	0.86	0.89	0.87
2	1.0	1.0	1.0
3	0.66	0.98	0.82
4	0.80	0.95	0.89
AVG:	0.83	0.95	0.89

Tab. 3.8: SPICE media - Risultati

Entrambe le tabelle 3.7 e 3.8 tengono in considerazione le risposte di quattro utenti rispetto i cinque a cui è stato somministrato il test. È stato ritenuto opportuno non considerare le risposte di un tester che non ha riportato correttamente alcuna formula, probabilmente sia a causa della sue conoscenze in materia che di lingua inglese.
Dalle tabelle presenti nella sezione 3.8.1 e dalla tabella 3.7 si evince come i risultati siano peggiori paragonati a quelli della prima fase di sperimentazione. Infatti, i risultati precedenti mostrano dei numeri molto vicini al 100%, indice di un'ottima comprensione delle formule rispetto alle percentuali ottenute in questo caso.
In realtà, guardando la tabella 3.8 ci si rende conto come seppur le formule risultino diverse, gli alberi delle formule originali e percepite siano strutturalmente simili.
Le cause di questa disparità sono da ritrovarsi sicuramente nella famigliarità dello strumento, dai profili utenti coinvolti e dalla possibilità che gli utenti avevano nella prima sperimentazione di riascoltare liberamente le formule.
Infine, le ultime considerazioni sono fatte rispetto al sistema di Speech Recognition. Nel dettaglio, risulta che la maggior parte degli utenti non abbia quasi mai interrotto il sistema di dialogo durante l'interazione.
Un esempio del suo utilizzo è stato il caso in cui è stato pronunciato "Ehy Stop!" per fermare la sintesi vocale della frase a causa di un imprevisto al di fuori della sperimentazione.

Questo si spiega perché durante l'interazione gli utenti sono stati impegnati a riscrivere la frase matematica. Le interazioni sono avvenute tutte al termine della sintesi vocale e la tabella 3.9 riporta nel dettaglio quante volte il sistema è riuscito a comprendere e rispondere alle domande degli utenti.

Utente	**F.- 1**	**F.- 2**	**F.- 3**	**D.- 1**	**D.- 2**	**D.- 3**
1	2(5)	1(1)	3(3)	4(4)	2(3)	4(4)
2	1(5)	7(9)	5(6)	2(3)	3(4)	3(4)
3	1(1)	0(1)	1(2)	0(1)	0(1)	1(1)
4	3(3)	2(2)	2(4)	2(2)	2(2)	4(4)
5	2(2)	1(1)	2(3)	3(3)	2(2)	2(4)

Tab. 3.9: Richieste utente comprese

In particolare, la tabella sovrastante riporta per ciascun utente il numero di richieste totali fatte (il numero tra parentesi), il numero di richieste comprese e quindi, a cui è conseguita una risposta per ciascuna formula facile (F.) e difficile (D.).
In base alle considerazioni precedentemente fatte e a questa tabella è visibile una effettiva curva di apprendimento positiva. Il numero di volte in cui gli utenti fanno una domanda a cui il sistema sa rispondere aumenta nel tempo considerando per l'appunto che sono state fatte ascoltare prima quelle facili e poi le difficili.
Al sistema che sa rispondere alle domande dell'utente corrisponde una comprensione della formula matematica maggiore.

Capitolo 4

Conclusioni

Questo lavoro si colloca nell'ambito delle tecnologie assistive con lo scopo di creare una soluzione per rendere accessibili le formule matematiche sia traducendole in frasi matematiche sia introducendo un sistema di dialogo per navigarle ed esplorarle per utenti ciechi e ipovedenti.
Riassumendo il lavoro svolto in vari punti, questi sono:

1. **Analisi linguistica delle formule matematiche**: si sono studiate le caratteristiche e le relazioni che intercorrono fra i diversi operatori matematici.

2. **Sintesi delle frasi matematiche in inglese**: sulla base delle analisi fatte, si è sviluppato un sistema che possa sintetizzare delle espressioni matematiche in linguaggio naturale che le rappresenti, chiamate frasi matematiche.

3. **Introduzione di un sistema di dialogo**: si è sviluppato un sistema che non solo sintetizzi vocalmente queste frasi, ma che offra uno strumento per interagire con l'utente permettendogli di navigare attivamente la formula, ossia ripetere e analizzare alcune sue parti.

4. **Sperimentazione**: testing del sistema di generazioni delle frasi matematiche e del sistema di dialogo.

I risultati ottenuti mostrano come il sistema sviluppato si comporti particolarmente bene nel soddisfare gli obiettivi prefissati.
In particolare, i risultati prodotti con la prima fase della sperimentazione evidenziano come l'applicazione riesca con successo a trasformare le formule matematiche in frasi in linguaggio naturale in inglese. La comprensione di queste usando clip audio contenenti le rappresentazioni delle formule, è stata particolarmente alta determinando sia la capacità del sistema di sintetizzare frasi matematiche sia di essere chiaro con le modalità di fruizione che riguardano la velocità e il tipo di voce usata.
Con la seconda fase della sperimentazione si è andato incontro all'esigenza di testare non solo le capacità del sistema di dialogo, ma anche se possa essere un'introduzione vantaggiosa e intelligente. Le modalità con cui è avvenuta questa, infatti, differiscono notevolmente dalla prima, è vertono a testare sia l'esperienza utente che la capacità del sistema di comprendere e di rispondere alle domande degli utenti.

I risultati prodotti in questo caso mostrano da una parte come il sistema di dialogo si sia generalmente comportato bene, dall'altra come possa essere notevolmente migliorato. I feedback, invece, lasciati dagli utenti sono stati molto positivi e segnale di un entusiasmo dettato dalle novità e possibilità che un sistema del genere possa dare loro.

4.0.1 Sviluppi futuri

In quest'ultima sezione della tesi si vogliono racchiudere dei diversi possibili futuri sviluppi per questo lavoro che nascono tra l'altro anche dai feedback e critiche positive degli utenti:

- Rimodellare il sistema di dialogo formalizzando l'iterazione sugli atti linguistici in modo da migliorare l'interazione con l'utente nel rispetto del modulo di Speech Recognition e favorendo la modularità del sistema;
- Migliorare il sistema di dialogo introducendo nuove funzionalità come la capacità di far impostare agli utenti la velocità e la voce dello speaker;
- Ampliare i comandi vocali del sistema di dialogo;
- Estendere l'insieme di operatori gestiti dal sistema linguistico e di generazione di frasi matematiche;
- Ampliare il sistema in modo da tradurre non solo formule ma anche grafici matematici e formule chimiche;
- Integrare il sistema con altri strumenti, come i browser;
- Aggiungere la possibilità di navigare ed esplorare le frasi matematiche anche con la tastiera;

Figure

Tabelle

Bibliografia

[1] "Simple NLG." https://github.com/simplenlg/simplenlg.

[2] G. Grätzer, *More Math into LaTeX 4th edition*. New York: Springer-Verlag, 2007.

[3] W3C, "Mathematical Markup Language (mathml)," vol. Version 3.0, no. 2nd Edition, 2014.

[4] "Treccani - Linguaggio Naturale." https://www.treccani.it/enciclopedia/linguaggio-naturale_%28Enciclopedia-della-Matematica%29/.

[5] D. J. . J. H. Martin, "Speech And Language Processing," vol. 2°, no. figura 12.3, p. 400, 2014.

[6] D. J. . J. H. Martin, "Speech And Language Processing," vol. 2°, no. figura 12.2, p. 400, 2014.

[7] D. J. . J. H. Martin, "Speech And Language Processing," vol. 2°, no. figura 12.4, p. 400, 2014.

[8] Marneffe, *Stanford Parser*. 2006.

[9] A. Gatt and E. Reiter, "SimpleNLG: A realisation engine for practical applications," pp. 1–4, 2016.

[10] L. Speakeasy, "Handbook for Spoken Mathematics," 1983.

[11] "Treccani - lingua parlata." https://www.treccani.it/enciclopedia/lingua-parlata_(Enciclopedia-dell%27Italiano)/.

[12] "Clojure." https://clojure.org/.

[13] "Java." https://www.oracle.com/it/java/.

[14] "Pandolfi." http://www.integr-abile.unito.it/Libri/Analisi1/html/analisi1.html.

[15] P. Anderson, B. Fernando, M. Johnson, and S. Gould, *SPICE: semantic propositional image caption evaluation*. abs/1607.08822: CoRR, 2016.

[16] "IBM Watson." https://www.ibm.com/it-it/cloud/watson-text-to-speech,.

[17] "NVDA." https://www.nvda.it/.

[18] J. Plass, R. Moreno, and R. Brünken, *Cognitive Load Theory*. New York, NY: Cambridge University, 2010.

[19] "Cognitive Load in UX and Voice Design." https://careerfoundry.com/en/blog/ux-design/voice-ui-design-and-cognitive-load/.

[20] P. Labinaz, "PAUL GRICE," pp. 321–325, 2012.

[21] D. J. . J. H. Martin, "Speech And Language Processing," vol. 3°, pp. 594–588, 2019.

[22] "Ito, K. and L. Johnson. 2017. The LJ speech dataset.." http:/keithito.com/LJ-Speech-Dataset/.

[23] "AWS Polly." https://aws.amazon.com/it/polly/.

[24] "eSpeak." http://espeak.sourceforge.net/.

[25] "What is Speech to Text?." https://aws.amazon.com/it/what-is/speech-to-text/.

[26] D. J. . J. H. Martin, "Speech And Language Processing," vol. 3°, pp. 584–588, 2019.

[27] "AWS Transcribe." https://aws.amazon.com/it/transcribe/.

[28] "AWS S3." https://aws.amazon.com/it/s3/features/?nc=sn&loc=2.

[29] Y. Wang, "WAKE WORD DETECTION AND ITS APPLICATIONS," pp. 2–3, 2021.

[30] "Benchmarking a Wake Word Detection Engine." https://picovoice.ai/blog/benchmarking-a-wake-word-detection-engine/.

[31] "Picovoice Console — Rhino Speech-to-Intent Engine." https://picovoice.ai/docs/quick-start/console-rhino/.

[32] "How to benchmark a rhino context." https://picovoice.ai/docs/benchmark/how-to-benchmark-a-rhino-context/.

[33] "Speech-To-Intent - Risultati benchmark." https://github.com/Picovoice/speech-to-intent-benchmark/blob/master/README.md.

[34] "Python." https://www.python.org/.

[35] "Flask." https://flask.palletsprojects.com/en/2.0.x/.

[36] "Rhino console." https://console.picovoice.ai/rhn.

[37] "User Experience Questionnaire." https://www.ueq-online.org/.