Despeect: interfaccia grafica per Speect

Giulio Paci

26 Ottobre 2017

http://www.mivoq.it/
info@mivoq.it MIVOQ S.R.L. +39 049 0998335

Indice

1 Obiettivo del progetto

L'obiettivo di questo progetto è realizzare un'interfaccia grafica per Speect [Meraka Institute(2008-2013)], una libreria per la creazione di sistemi di sintesi vocale, che agevoli l'ispezione del suo stato interno durante il funzionamento e la scrittura di test per le sue funzionalità.

1.1 Sintesi vocale

La sintesi vocale da testo scritto (TTS) è quella tecnologia che permette la conversione di un qualsiasi testo in voce. Negli ultimi anni si è assistito ad un rapido diffondersi di questo tipo di tecnologia in numerosi ambiti (per es.: voci guida dei navigatori satellitari, annunci dei mezzi di trasporto pubblico, centralini telefonici, lettori di messaggi) e al suo affermarsi come una delle interfacce di fruizione abituale per tutte quelle applicazioni in cui è impedito, o comunque limitato, l'utilizzo della vista (per es.: durante la guida).

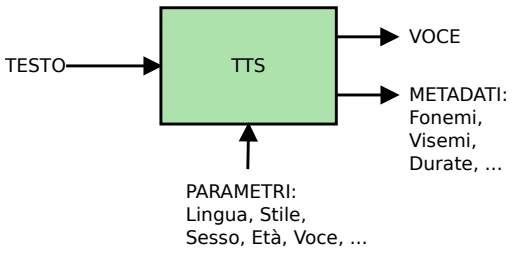

Figura 1: Schema semplificato di un sistema di sintesi vocale

Da un punto dello sviluppatore che utilizza questa tecnologia per realizzare applicazioni come quelle appena citate, un sistema di sintesi vocale appare come un sistema in grado di ricevere un testo in ingresso e di restituire in uscita un file vocale, come illustrato in figura 1. Lo sviluppatore è di solito in grado di pilotare la sintesi specificando alcuni parametri (per esempio la voce da utilizzare per la generazione dell'audio) e può avere accesso a vari metadati (per esempio la sequenza di fonemi/visemi pronunciate) che possono essere utili per la realizzazione di applicazioni specifiche (per esempio l'animazione automatica del labiale di un personaggio virtuale).

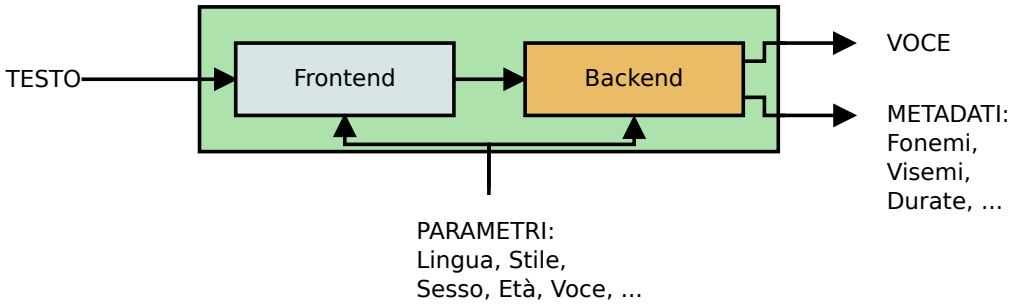

Figura 2: Schema semplificato di un sistema di sintesi vocale

Nella maggioranza dei casi questi sistemi sono progettati in due blocchi principali, come illustrato in figura 2:

- il frontend, che si occupa di effettuare l'analisi linguistica del testo in ingresso e di estrapolare da questa le informazioni necessarie al backend per generare il file vocale. Queste informazioni consistono generalmente in una descrizione dettagliata di ogni fonema da pronunciare, rappresentano cioè una sequenza fonetica;
- il backend, che converte le informazioni fornite dal frontend (una sequenza fonetica) in una forma d'onda.

1.1.1 Frontend di un sistema TTS

Il frontend di un sistema di sintesi vocale si occupa di effettuare l'analisi linguistica del testo in ingresso, con lo scopo di ricavare la sequenza fonetica da pronunciare. Un frontend è solitamente composto da diverse componenti di analisi, eseguite in sequenza, ciascuna delle quali si occupa di un compito ben preciso ed ha lo scopo di interpretare opportunamente alcuni elemento del testo in ingresso. Ciascuna componente può fare affidamento sul testo d'ingresso e sui risultati delle componenti già eseguite per produrre i propri risultati.

Come illustrato in figura 3, la maggior parte dei frontend dei sistemi TTS attuali contiene almeno le seguenti componenti:

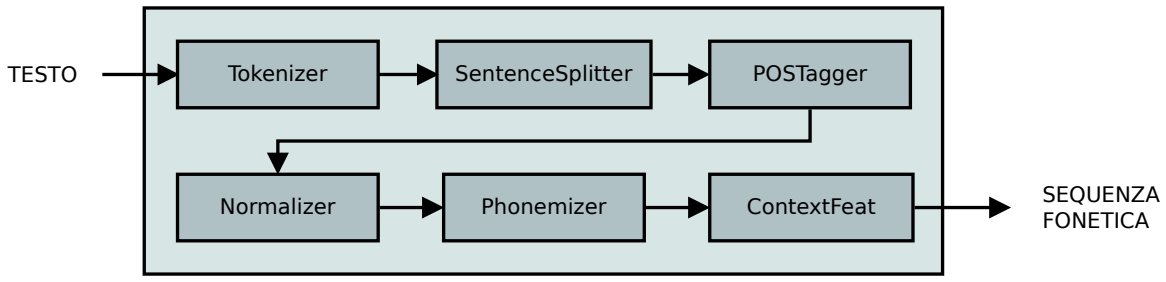

Figura 3: Schema di un ipotetico frontend per un sistema di sintesi vocale

- tokenizer: dato un testo, produce una sequenza di token, generalmente corrispondenti a singole parole o a singoli elementi di testo. I token dovrebbero agevolare l'analisi successiva, così solitamente un token corrisponde ad una singola parola, ad un singolo carattere di punteggiatura, ma può corrispondere anche ad un'abbreviazione, ad un url o ad una data. Il compito, in apparenza semplice, è in realtà complicato dalla presenza di numerosi elementi ambigui (per esempio il carattere ".", che può corrispondere ad un carattere di punteggiatura, ma anche essere parte di un'abbreviazione, di un indirizzo email o di un url);
- sentence splitter: data una sequenza di token, produce una sequenza di frasi. Ogni frase corrisponde ad una sequenza di token. Anche in questo caso sono presenti numerose ambiguità che complicano il lavoro.
- pos tagger: data una sequenza di token appartenenti ad una singola frase, etichetta ogni frase con una categoria grammaticale, cioè effettua l'analisi grammaticale di una frase. Questo permette, fra le altre cose, di disambiguare la pronuncia di alcune parole sulla base del contesto (per esempio quella di "àncora", nome, invece di "ancóra", avverbio);
- normalizer: data una sequenza di token, produce una sequenza di parole "pronunciabili", espandendo numeri, date, url, email... in parole;
- phonetizer: data una sequenza di parole "pronunciabili", produce una sequenza di fonemi, cioè simboli che rappresentano l'intenzione di pronunciare un determinato suono di una determinata lingua.
- context-feat: data una sequenza di fonemi, la arricchisce aggiungendo informazioni di contesto, cioè riportando nella sequenza le caratteristiche del fonema ricavate dai risultati prodotti dalle precedenti componenti di analisi e che consentono di caratterizzare meglio i suoni da riprodurre. Verranno cioè riportate informazioni come la posizione nella frase della parola a cui appartiene il fonema, e/o la lunghezza della frase, e/o la distanza del fonema da un token di punteggiatura e/o se la parola di cui il fonema fa parte è una parola funzione, come gli articoli, o una parola di contenuto, come i nomi, ecc.; la sequenza di fonemi così arricchita è la sequenza fonetica che fornirà l'ingresso per il backend.

1.1.2 Backend di un sistema TTS

Il backend di un sistema di sintesi vocale si occupa di generare una forma d'onda data una sequenza fonetica che rappresenta l'intenzione di pronunciare determinati suoni. La sequenza fonetica è una rappresentazione astratta, che porta informazioni di tipo linguistico e di tipo qualitativo, ma non porta informazioni di tipo quantitativo relative all'intonazione, al ritmo o alla durata di ciascun suono. Queste informazioni quantitative devono essere ricavate dal backend ed utilizzate per generare la forma d'onda.

Esistono numerose tecniche per realizzare un backend di un sistema TTS e le più diffuse a livello commerciale sono la sintesi per selezione d'unità (US) e la sintesi statistica parametrica (SPSS).

La US prevede di disporre di un grande database di frasi pronunciate da uno speaker ed annotate con informazioni di tipo linguistico analoghe a quelle prodotte dal frontend. Al momento della generazione della forma d'onda le informazioni della sequenza fonetica in ingresso vengono utilizzate per cercare all'interno del database porzioni di audio le cui annotazioni corrispondano il più possibile alla sequenza di ingresso. La generazione consiste quindi nella concatenazione di segmenti di audio estratti dal database.

La SPSS prevede di disporre di un database di modelli statistici associati a ciascun suono realizzabile. Le informazioni della sequenza fonetica in ingresso vengono utilizzate per selezionare i modelli statistici più vicini a quelli dei suoni da pronunciare. I modelli statistici sono quindi utilizzati per predire la durata di ciascun suono, l'intonazione, l'inviluppo spettrale e, in generale, una sequenza di parametri che possa essere utilizzata per pilotare un vocoder. Il vocoder trasforma infine i parametri in ingresso in una forma d'onda.

1.2 Speect

Speect [Meraka Institute(2008-2013)] è una libreria opensource per lo sviluppo di frontend e backend di un sistema di sintesi vocale.

La libreria è stata sviluppata presso il Meraka Institute in Sud Africa per coniugare due esigenze principali:

- poter sperimentare velocemente nuove tecniche di sintesi vocale;
- poter trasferire velocemente queste tecniche in produzione.

La libreria è scritta in linguaggio C, con particolare riguardo alla portabilità. Per la maggior parte delle funzionalità espone delle API orientate agli oggetti, accessibili sia in linguaggio C che in Python.

L'implementazione di Speect si divide in due parti, l'engine e i plugin.

L'engine implementa tutte le funzionalità di base che non siano dipendenti da specifici moduli TTS o da specifiche componenti di analisi, mentre ogni funzionalità specifica è implementata attraverso plugin.

Fanno parte dell'engine:

- il sistema di gestione di plugin, attraverso cui è possibile implementare facilmente funzionalità aggiuntive (componenti di analisi, vocoder, scrittura e lettura di file secondo formati specifici, ...);
- il sistema ad oggetti, e la definizione di alcune classi fondamentali;
- la struttura dati principale di Speect, chiamata Heterogeneous Relation Graph (HRG) [Taylor et al.(2001)] e descritta in sezione 1.2.1;
- in generale tutto quanto sia generico e non dipendente da moduli TTS specifici.

Fanno parte dei plugin tutti gli elementi fondamentali di un frontend e di un backend:

- le componenti di analisi linguistica (come quelle mostrate in figura 3 e descritte in sezione 1.1.1);
- le componenti necessarie ad implementare un backend come quelli descritti in sezione 1.1.2

1.2.1 Heterogeneous Relation Graph

Uno degli elementi fondamentali dell'architettura di Speect è il concetto di utterance, che rappresenta sia l'unità di input da analizzare che i risultati delle analisi compiute su di essa. Normalmente si considera come unità di input una porzione di testo corrispondente ad una o più frasi, sulla quale vengono compiute una sequenza di analisi analoghe a quelle di figura 3 e i cui risultati sono considerati parte integrante della utterance.

Per memorizzare le informazioni relative ad una utterance Speect implementa una struttura dati chiamata Heterogeneous Relation Graph (HRG) [Taylor et al.(2001)], di cui viene mostrato un esempio in figura 4.

Un HRG è composto da nodi (i punti neri in figura), archi (le frecce) e relazioni (i riquadri etichettati). A ciascun nodo del grafo sono associate delle proprietà, memorizzate in una mappa chiave-valore.

Un nodo può essere presente in una o più relazioni; in figura 4 i cinque nodi contenuti nella relazione "Tokens" sono anche presenti nelle altre due relazioni, mentre gli altri nodi appartengono ad una sola relazione. Anche se un nodo può appartenere a più relazioni, le proprietà sono comunque del nodo e non dipendono dall'appartenenza o meno ad una determinata relazione (in altre parole, con riferimento alla figura, per i cinque nodi appartenenti a più di una relazione sono presenti solo cinque mappe chiave-valore, una per ogni nodo e non quindici).

In un HRG in ogni relazione, da ogni nodo, possono uscire fino ad un massimo quattro archi: next, prev, daughter e parent. Gli archi hanno una direzione e in figura sono rappresentati con delle frecce. Gli archi next e prev collegano i nodi in figura in orizzontale. Nella relazione "Tokens", per esempio, sono impiegati solo archi di tipo next (da sinistra verso destra) e di tipo prev (da destra verso sinistra), mentre non sono impiegati archi di tipo daughter o di tipo parent. Nelle relazioni "Words" e "Sentences" sono impiegati archi di entrambi i tipi ed in particolare sono archi di tipo daughter (dall'alto verso il basso) gli archi che collegano i nodi della prima riga con quelli della seconda. Sono nodi di tipo parent (dal basso verso l'alto), quelli che collegano i nodi della seconda riga con quelli della prima.

Grazie ai quattro tipi di arco è possibile realizzare, in ogni relazione, altre strutture dati come le double-linked list (se ne vede un esempio nella relazione dei "Tokens", in cui ogni nodo è collegato al successivo e al precedente medianti archi), alberi (se ne vede un esempio nella relazione "Sentences" in cui gli archi di tipo daughter sono utilizzati per indicare il primo figlio di un nodo, mentre gli altri figli sono collegati fra loro con archi di tipo prev e next e al padre con archi di tipo parent), o altre strutture ibride come quella nella relazione delle "Words".

Poiché in un HRG è possibile definire un numero arbitrario di relazioni, è facile intuire come questo sia di norme aiuto nel garantire flessibilità e ordine. In Speect quando una componente di analisi opera su di una utterance, questa può prendere le informazioni di cui necessita sia dall'input originale che dal grafo HRG associato. Ogni componente di analisi assume di poter trovare i dati di cui necessita in specifiche relazioni e, almeno in linea teorica, salva i risultati delle proprie analisi in una relazione diversa, o fra le proprietà dei nodi. In questo modo

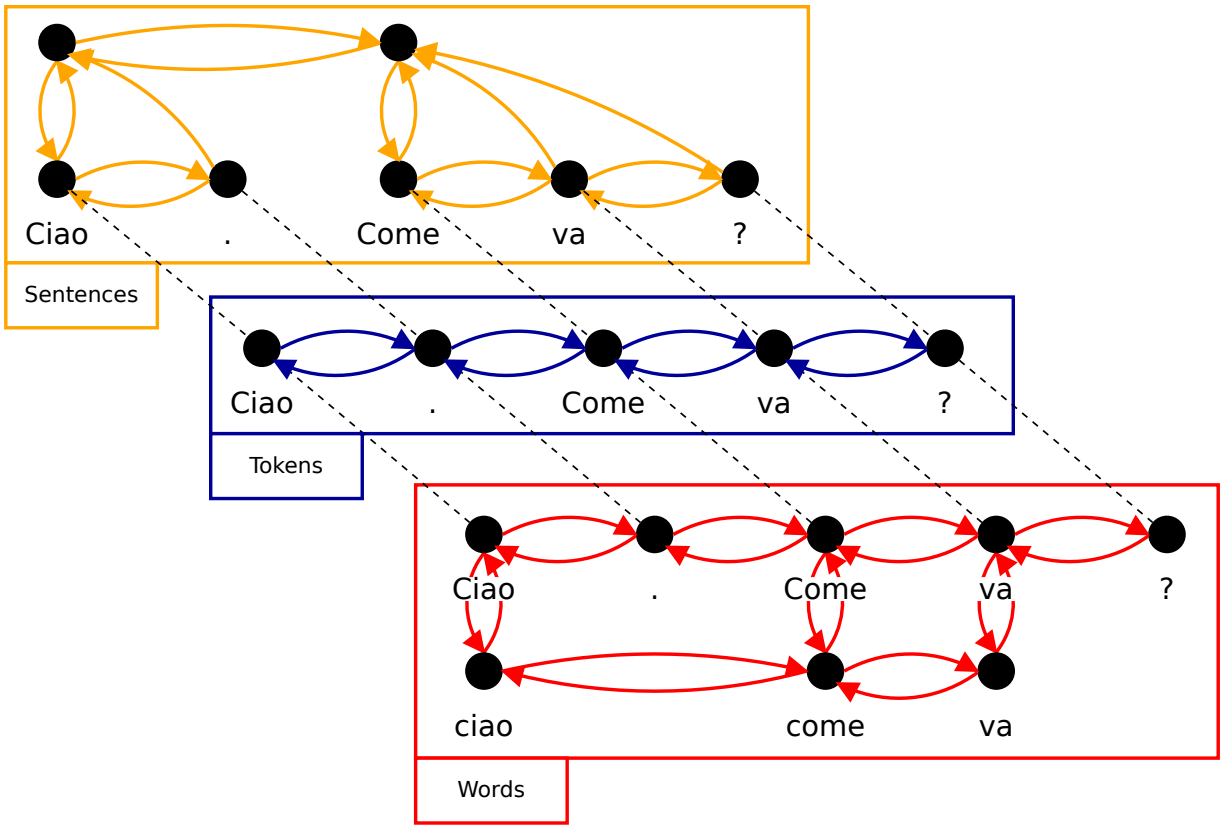

Figura 4: Esempio di un Heterogeneous Relation Graph

ogni componente non distrugge il proprio input, ma lo arricchisce e rendendo disponibile per le altre componenti i propri risultati.

Con riferimento alle figure 3 e 4, dato il testo in ingresso "Ciao. Come va?", per analizzarlo con Speect, il testo verrebbe dapprima aggiunto ad una utterance, verrebbe creato un HRG vuoto e poi la utterance verrebbe analizzata dalle varie componenti in sequenza. Un ipotetico Tokenizer aggiungerebbe al grafo la relazione Tokens così come illustrato in figura 4. Sulla base di questa il SentenceSplitter potrebbe dividere la sequenza di token in due frasi, aggiungendo al grafo la relazione Sentences che rappresenta questa suddivisione utilizzando una struttura ad albero. Un POSTagger aggiungerebbe la proprietà "POS" a ciascun nodo foglia della relazione Sentences, senza aggiungere altre relazioni, perché questa componente aggiunge generalmente solo etichette. Un Normalizer aggiungerebbe invece la relazione Words, partendo dalle informazioni ricavate dal Tokenizer e dal POSTagger. Per ogni token creerebbe zero o più nodi, formando una sequenza di parole pronunciabili. Un ipotetico Phonemizer aggiungerebbe quindi una quarta relazione (non mostrata in figura) contenente una struttura ad albero in cui ad ogni parola pronunciabile prodotta dal Normalizer è associata una lista di fonemi. Infine la componente ContextFeat sfrutterebbe l'intero HRG per arricchire le proprietà di ciascun fonema. Una sequenza di fonemi così prodotta può quindi essere utilizzata come input per il backend.

1.2.2 Configurazione di Speect

Elemento centrale della configurazione di Speect è il concetto di voce, che definisce il tipo di elaborazioni che è possibile effettuare.

Ogni voce è descritta da un file json, contenente i seguenti campi:

- voice-definition: meta-informazioni relative alla (nome, versione, descrizione, ...);
- feature-processors: configurazioni relative a componenti di analisi che agiscono su un singolo elemento. Ogni configurazione è caratterizzata da un identificativo, dal riferimento ad una classe che implementa la componente di analisi e dai parametri di configurazione per la classe;
- utterance-processors: configurazioni relative a componenti di analisi che agiscono globalmente sui dati da analizzare. Ogni configurazione è caratterizzata da un identificativo, dal riferimento ad una classe che implementa la componente di analisi e dai parametri di configurazione per la classe;

- utterance-types: sequenze di componenti di analisi. Per ogni sequenza è specificato un nome ed una lista di identificativi, ciascuno corrispondente ad un elemento descritto in utterance-processors;
- data e features: parametri e dati utilizzabili da diverse componenti di analisi;
- plug-ins: plugin da caricare preventivamente (per esempio per supportare uno specifico formato per il caricamento di dati).

Una volta caricata una voce è possibile applicare una qualunque delle sequenze descritte in utterance-types su di un input. Poiché le sequenze sono del tutto generiche è possibile configurare speect per svolgere molti compiti di analisi del linguaggio naturale (NLP) e non solo per realizzare un sistema di sintesi vocale.

2 Interfaccia grafica per Speect

Il progetto prevede la realizzazione di un'interfaccia grafica per Speect, che agevoli lo sviluppo e l'ispezione dello stato interno della libreria.

Durante lo sviluppo di Speect, ed in particolare dei suoi plugin, sorge spesso l'esigenza di ispezionare lo stato del grafo HRG (vedi sezione 1.2.1), al fine di capire come ogni plugin si sta comportando. In questo caso l'interfaccia si dovrebbe comportare come una specie di debugger specializzato.

Poiché i plugin agiscono su un particolare grafo HRG, i loro programmi di test dovrebbero essere in grado di caricare un particolare stato interno e verificare che gli effetti dell'esecuzione del plugin siano quelli attesi. Purtroppo oggi non è agevole ricreare uno specifico stato. In questo caso l'interfaccia dovrebbe consentire di editare lo stato interno manualmente, al fine di agevolare lo sviluppo di test.

Nella realizzazione delle varie componenti è spesso necessario percorrere il grafo. Per la navigazione Speect offre uno strumento con cui è possibile specificare un percorso mediante stringa. Dato un nodo non è sempre semplice comprendere qual è il nodo corrispondente a tale percorso. Uno strumento di visualizzazione dei percorsi semplificherebbe lo sviluppo.

La configurazione di Speect (vedi sezione 1.2.2) avviene sulla base di file json. Modificare questi file non è sempre intuitivo ed è facile commettere errori. Un'interfaccia grafica opportuna potrebbe semplificare la configurazione di Speect.

3 Specifiche di progetto e riferimenti

Scopo del progetto è quello di realizzare un'applicazione grafica, simile come concetto a quella illustrata in figura 5.

L'applicazione illustrata in figura 5 consente la visualizzazione di un grafo HRG (vedi sezione 1.2.1). Nel corpo centrale è visualizzato il grafo: ogni nodo è rappresentato da un punto nero ed è etichettato utilizzando il valore della sua proprietà "name". In figura 5 il grafo mostra simultaneamente tre relazioni. Ed ogni arco assume un colore diverso a seconda della relazione in cui tale arco è presente. Utilizzando la lista "Relations" a destra del grafo è possibile decidere quali relazioni devono essere mostrate. Selezionando un nodo con un click, è possibile visualizzarne le proprietà dalla tabella "Properties" e modificarle. In figura è selezionato il nodo "come" in basso al centro. Selezionando un arco è possibile rimuoverlo, come è possibile aggiungere un arco collegando due nodi.

È possibile caricare/salvare un grafo da/su file attraverso i bottoni "Load" e "Save" a destra del grafo.

Nella casella di testo "Path" è possibile specificare un percorso relativo. Selezionando un nodo è possibile richiedere l'evidenziazione del nodo che si raggiunge con quel percorso. In figura è stato inserito il percorso "R:Words.parent.n", utilizzando la sintassi dei percorsi di Speect (si veda la documentazione presente sul sito [Meraka Institute(2008-2013)]) e partendo dal nodo selezionato "come", tale percorso equivale a spostarsi sulla relazione "Words", spostarsi al nodo padre (parent) e infine spostarsi al nodo successivo (n), raggiungendo così il nodo evidenziato in giallo "va".

Oltre al caricamento di un grafo da file, l'applicazione consente anche di generare automaticamente un grafo HRG, caricando una configurazione da un file voice.json (vedi sezione 1.2.2). Al caricamento del file viene popolata la lista dei "Processors" in alto a sinistra. Questa lista corrisponde sostanzialmente al campo "utterance-processors" della configurazione (vedi sezione 1.2.2) e riporta tutte le componenti di analisi che sono state configurate e sono quindi utilizzabili. Nella casella di testo "Input Text" è possibile specificare un testo da analizzare. L'analisi si può effettuare premendo la freccia a destra della casella, per effettuare un'intera sequenza di analisi, come quella illustrata in figura 3, oppure premendo le freccie nella lista dei "Processors", per eseguire la singola componente di analisi.

Le ultime due liste mostrate in figura, "Utterance Plugins" e "Features Plugins" riportano la lista dei plugin disponibili e possono essere utilizzate per aggiungere manualmente delle componenti di analisi alla lista dei "Processors".

L'esempio appena descritto vuole servire da linea guida per la progettazione e chiarire almeno in parte gli obiettivi dell'applicazione, ma non è obbligatorio realizzare ogni singola funzionalità descritta (vedi sezione 4.1), né è obbligatorio rispettarne completamente il flusso.

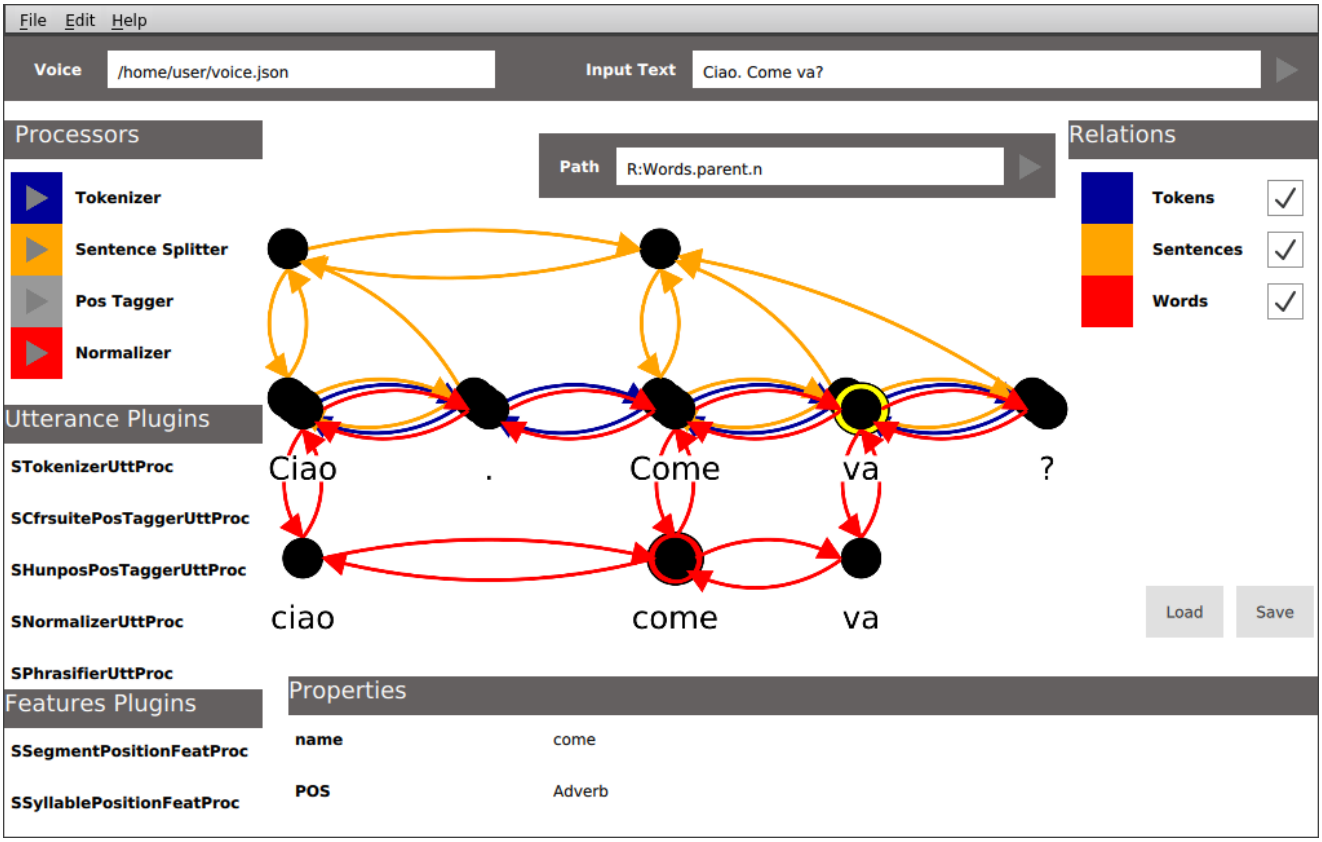

Figura 5: Esempio di interfaccia grafica da realizzare

La progettazione e l'implementazione dovranno tener conto della manutenibilità e dell'estensibilità dell'applicazione, cercando di conciliare i vari casi d'uso (sviluppatore che vuole ispezionare lo stato interno della libreria, sviluppatore che vuole creare dei test, sviluppatore che vuole creare configurazioni particolari). Nelle scelte progettuali si raccomanda di mantenere, per quanto possibile, una forte separazione fra viste e dati da visualizzare.

Potrebbe capitare che per l'implementazione di alcune funzionalità possa rendersi necessaria la modifica di Speect. In tal caso si raccomanda di contattare la proponente e di discutere eventuali strategie e possibili soluzioni di comune accordo.

I commenti al codice, i nomi delle funzioni e delle variabili dovrebbero essere esplicativi ed in lingua inglese. I commenti dovrebbero essere in misura tale da permettere la comprensione del codice: un buon commento non dovrebbe descrivere la sintassi del linguaggio di programmazione in uso (quindi evitare commenti come `i = a + b; /* assign to i the sum of a and b */`), ma piuttosto descrivere lo scopo di tutti quei blocchi di codice la cui comprensione non sia già resa immediata da una corretta scelta della nomenclatura (per es.: `digit_value = digit_character - '0'; /* this line assumes that digits' characters maps to consecutive integers, ordered by value, like in ASCII character set */`).

4 Scelte tecnologiche

L'applicazione dovrà essere compatibile con Linux, ma è incoraggiato lo sviluppo multipiattaforma. Viene lasciato un ampio margine in merito alle tecnologie impiegate, purché compatibili con Linux e, preferibilmente, rilasciate con licenza opensource. L'unico requisito fondamentale in questo senso è l'utilizzo di Speect ed in particolare della versione modificata dalla proponente [Mivoq(2014-2017)]. Qualora fosse necessario è consentito ampliare e modificare questa libreria per venire incontro alle esigenze del capitolato.

Per quanto riguarda l'interfaccia utente si suggerisce l'utilizzo di librerie portabili, come Gtk+ [The GTK+ Team(1998-2017 o Qt [The Qt Company(1995-2017a)] ed in particolare di programmi come Glade [The GNOME Project(1998-2017)] o QtCreator [The Qt Company(1995-2017b)] per lo sviluppo rapido dell'interfaccia.

Per l'automazione della compilazione si suggerisce l'utilizzo di CMake [Kitware Inc.(2000-2017)], in quanto già usato da Speect e alcune sue dipendenze.

4.1 Obiettivi obbligatori

Gli obiettivi obbligatori del progetto sono:

- realizzazione di un'interfaccia grafica per Speect
  - visualizzazione dei risultati delle componenti di analisi linguistica;
- documentazione dell'applicazione
  - analisi dei requisiti;
  - descrizione tecnica.

4.2 Obiettivi desiderabili

- possibilità di manipolare la struttura dati interna di Speect
  - caricamento;
  - salvataggio;
  - modifica manuale;
- possibilità di caricare ed eseguire manualmente singole componenti di analisi;

4.3 Obiettivi opzionali

- possibilità di visualizzare percorsi su un grafo;
- possibilità di confrontare visivamente due stati della struttura interna di Speect;
- possibilità di confrontare automaticamente due stati della struttura interna di Speect;
- possibilità di manipolare la configurazione di Speect
  - caricamento;
  - modifica;
  - salvataggio.

A Note sul capitolato d'appalto

Il presente capitolato d'appalto rappresenta una prima bozza di lavoro per aiutare il fornitore a capire le tematiche del progetto e del tipo di sviluppo che sarà necessario. La proponente si rende disponibile per ogni ulteriore chiarimento.

A.1 Aspettative della proponente

La proponente si aspetta di ottenere un'applicativo che possa essere immediatamente utilizzato per agevolare lo sviluppo di sistemi TTS, ed in particolare semplificando la creazione di test, il debug e la configurazione di Speect.

L'applicazione realizzata, salvo ulteriori accordi, resterà di proprietà esclusiva del fornitore. La proponente incoraggia, ove possibile, il rilascio con licenze opensource e preferibilmente con una licenza in stile BSD/MIT, analoga a quella con cui è rilasciato Speect.

A.2 Contatti

La proponente potrà essere contattata in qualsiasi momento attraverso l'indirizzo email `tech@mivoq.it`, al quale risponde il reparto tecnologico dell'azienda. Le email dovranno contenere in oggetto la sigla "[UNIPD-TTS]" e dovranno essere indirizzate all'attenzione di Giulio Paci, che sarà il referente principale per il progetto.

In alternativa è possibile telefonare al numero 0490998335, al quale risponde il reparto tecnologico dell'azienda. In questo caso sarà sufficiente specificare che si chiama a proposito del progetto di "Ingegneria del software".

B Note sulla proponente

Mivoq è una startup, nata nel 2013, che si occupa di sintesi vocale, con lo scopo di permettere ad ogni singolo utente di avere la propria voce digitale che possa rappresentarlo anche dove normalmente non è presente, attraverso applicazioni innovative per la lettura di SMS con la voce del mittente, la lettura di post su Facebook con la voce dell'autore, la personalizzazione di assistenti virtuali con la voce dell'utente o la produzione della propria voce quando questa è venuta meno a causa di un'operazione o di una malattia (si pensi per esempio al fisico Stephen Hawking).

L'obiettivo di Mivoq non è di realizzare queste applicazioni, ma di fornire la tecnologia necessaria a fare in modo che vengano realizzate. Con questo progetto intendiamo migliorare il processo di sviluppo di questa tecnologia introducendo un'importante strumento di debug.

Riferimenti bibliografici

[Meraka Institute(2008-2013)] Meraka Institute. (2008-2013) Speect. [Online]. Available: http://speect.sourceforge.net/

[Taylor et al.(2001)] P. Taylor, A. W. Black, and R. Caley, "Heterogeneous relation graphs as a mechanism for representing linguistic information," *Speech Communication*, vol. 33, pp. 153–174, 2001.

[Mivoq(2014-2017)] Mivoq. (2014-2017) Speect. [Online]. Available: https://github.com/mivoq/speect

[The GTK+ Team(1998-2017)] The GTK+ Team. (1998-2017) The gtk+ project. [Online]. Available: https://www.gtk.org/

[The Qt Company(1995-2017a)] The Qt Company. (1995-2017) Qt - cross-platform software development for embedded and desktop. [Online]. Available: https://www.qt.io/

[The GNOME Project(1998-2017)] The GNOME Project. (1998-2017) Glade - a user interface designer. [Online]. Available: https://glade.gnome.org/

[The Qt Company(1995-2017b)] The Qt Company. (1995-2017) Qt - libraries and apis, tools and ide. [Online]. Available: https://www.qt.io/qt-features-libraries-apis-tools-and-ide/

[Kitware Inc.(2000-2017)] Kitware Inc. (2000-2017) Cmake - cross platform makefile generator. [Online]. Available: https://cmake.org/