Alma Mater Studiorum
Università di Bologna
Campus di Cesena

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

Implementazione di un sistema di traduzione automatica: un caso di studio per la traduzione italiano-cinese

Relatore:
Prof. Silvia Mirri

Correlatore:
Giovanni Delnevo

Presentata da:
Alessandro Pioggia

Sessione II
Anno Accademico 2021-2022

Desidero dedicare la tesi alla mia famiglia ed ai miei amici.

Introduzione

Il lavoro di tesi presentato è nato da una collaborazione con il Politecnico di Macao, i referenti sono: Prof. Rita Tse, Prof. Marcus Im e Prof. Su-Kit Tang. L'obiettivo consiste nella creazione di un modello di traduzione automatica italiano-cinese e nell'osservarne il comportamento, al fine di determinare se sia o meno possibile l'impresa. Il trattato approfondisce l'argomento noto come Neural Language Processing (NLP), rientrando dunque nell'ambito delle traduzioni automatiche. Sono servizi che, attraverso l'ausilio dell'intelligenza artificiale sono in grado di elaborare il linguaggio naturale, per poi interpretarlo e tradurlo. NLP è una branca dell'informatica che unisce: computer science, intelligenza artificiale e studio di lingue. Dal punto di vista della ricerca, le più grandi sfide in questo ambito coinvolgono: il riconoscimento vocale (speech-recognition), comprensione del testo (natural-language understanding) e infine la generazione automatica di testo (natural-language generation). Lo stato dell'arte attuale è stato definito dall'articolo "Attention is all you need" [1], presentato nel 2017 a partire da una collaborazione di ricercatori della Cornell University.
I modelli di traduzione automatica più noti ed utilizzati al momento sono i Neural Machine Translators (NMT), ovvero modelli che attraverso le reti neurali artificiali profonde, sono in grado effettuare traduzioni o predizioni. La qualità delle traduzioni è particolarmente buona, tanto da arrivare quasi a raggiungere la qualità di una traduzione umana. Il lavoro infatti si concentrerà largamente sullo studio e utilizzo di NMT, allo scopo di proporre un modello funzionale e che sia in grado di performare al meglio nelle traduzioni

da italiano a cinese e viceversa. La tesi è organizzata secondo la seguente struttura:

- Il primo capitolo introduce il contesto della tesi, presentando gli argomenti principali e descrivendo il dominio applicativo. Nella sezione introduttiva vengono introdotti e spiegati gli Statistical Machine Translators (SMT), ovvero modelli di traduzione utilizzati prima dell'avvento del machine learning, i quali fanno affidamento alla teoria della probabilità e alla statistica [2]. Nella seconda parte invece, vengono descritti e spiegati nel dettaglio i Neural Machine Translators (NMT), in seguito ad una veloce introduzione. Nell'introduzione vengono presentate le Recurrent Neural Networks (RNN) e il Transformer, dimostrando perché il secondo è il più indicato in ambito traduzioni [3].

- Nel secondo capitolo vengono presentate le tecnologie utilizzate per la creazione del modello e di conseguenza per la sperimentazione. La parte introduttiva si contraddistingue in quanto vengono enunciati i più noti ed efficaci sistemi di traduzione in cloud esistenti. Essi sono serviti per avere a disposizione un punto di riferimento, in particolare osservando il funzionamento ed i risultati ottenuti a partire dalle traduzioni effettuate. Una volta analizzate le tecnologie di cloud più note, nella seconda sezione del capitolo è stato spiegata la tecnologia Open-Nmt [4], basata su tensorflow [5], utilizzata per la realizzazione del modello. Infine è stato necessario sfruttare anche docker, un noto sistema di virtualizzazione che è risultato utile per rendere il codice del progetto portabile, al fine di semplificarne la fase di addestramento [6].

- Infine nel terzo ed ultimo capitolo vengono mostrati e discussi i risultati degli esperimenti effettuati. Inizialmente, viene spiegato nello specifico il criterio che permette di stimare il punteggio BLEU [7]. BLEU è una metrica che, a partire da due file di traduzione, source e target, è in grado di stimarne il livello qualitativo. Nella sezione successiva, vengono poi spiegati nel dettaglio i passaggi utili per la creazione del modello

di traduzione, con una overview di tutti i file di configurazione utilizzati, oltre al dataset selezionato. Infine nell'ultima parte, è possibile osservare attraverso delle tabelle, i risultati, che sono stati poi discussi e contestualizzati.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Contesto

Il seguente capitolo introduce il contesto della tesi, proponendo una trattazione teorica degli argomenti principali. La prima sezione riguarda la presentazione dei modelli di traduzione utilizzati in passato, l'ultima invece introduce i modelli che sfruttano il machine learning e rappresentano lo stato dell'arte.

1.1 Statistical machine translation

Gli statistical machine translators (SMT) [2] sono modelli di traduzione introdotti nel 1947, che con la successiva introduzione dei personal computers, gli scienziati sono stati in grado di applicare. Si tratta di modelli che, attraverso la teoria della probabilità, unita a concetti statistici, sono in grado di predirre un risultato (o target), a partire da una larga quantità di dati in ingresso (source). Il meccanismo introdotto è utilizzabile per:

– tradurre del testo (non parole singole), la predittività è in grado di legare correttamente le parole fra loro, dandogli un senso;

– implementare delle funzioni di autocompletamento, ad esempio, a partire da una domanda, tenta di indovinare una possibile risposta.

1.1.1 Dataset per traduzioni automatiche

Dal momento che i modelli citati si basano principalmente sulla statistica, è necessario avere a disposizione una grande quantità di dati, catalogati ed organizzati in un data set. In particolare il data set, è ottenuto ed estratto a partire da traduzioni effettuate in precedenza (effettuate ed approvate da esseri umani, non AI) e prende il nome di corpus. Le organizzazioni mondiali più grandi mettono a disposizione le loro banche dati, alle quali si può accedere rispettando la privacy degli individui. Una banca dati, non è un semplice archivio di dati sparsi, dal momento che, per essere considerato tale, deve rispettare determinati requisiti:

– i dati raccolti devono essere organizzati e catalogati, al fine di facilitare l'estrazione;

– deve essere possibile estrarre sia il singolo dato che il tutto.

Una banca dati, a differenza da un archivio, può essere tutelata da diritto d'autore o sui generis. Il primo tutela l'inventore del dataset, ovvero colui che ha avuto l'idea, attraverso la scelta dei dati da inserire e la disposizione del materiale. Il secondo invece sussiste nel caso in cui siano state investite delle risorse, ad esempio tempo o denaro e conferisce il diritto di porre il veto sull'estrazione di tutti i dati o parte sostanziale di essi, non è invece possibile vietare l'estrazione del singolo dato.

1.1.2 Teoria della probabilità

Il corretto funzionamento degli SMT verte sulla creazione di modelli, in grado di predirre con una buona precisione gli eventi futuri. A questo proposito, è di vitale importanza la comprensione e l'utilizzo delle nozioni introdotte nella teoria delle probabilità, che descrive e studia i fenomeni aleatori. Un fenomeno è un qualunque accadimento che porta ad un risultato, o esito, si dice deterministico se il risultato può essere calcolato o previsto con esattezza prima dell'effettuazione del fenomeno. Si dice invece aleatorio se non è possibile prevedere il risultato. L'insieme di tutti i possibili risultati di un fenomeno si indica con Ω. L'evento è un sottoinsieme di Ω, ovvero è l'insieme dei possibili risultati di un fenomeno aleatorio. Una valutazione di probabilità è invece, una funzione che associa ad ogni evento un numero tanto più grande quanto più riteniamo che tale evento possa accedere. Viene denotato da $P(E)$, per descrivere la valutazione dell'evento E.
Esempio dimostrativo:

- $\Omega = \{$ "okay, see you later", "oh really, I'm sorry", "don't bother me please!"$\}$
- $E =$ "la risposta estratta è "okay, see you later"";
- $P(E) = \frac{1}{3}$

A partire da un insieme di possibili eventi Ω , la probabilità che la frase selezionata venga estratta è 1 su 3.

Il fulcro della predittività del modello ruota intorno alla probabilità condizionale, di conseguenza viene sfruttato il teorema di Bayes.

Probabilità condizionale

Considerando due eventi distinti A e B, il calcolo della probabilità condizionale consta nel definire la probabilità che accada l'evento B

sapendo che è giù avvenuto A. Questa probabilità la definiamo come $P(A|B)$ e si calcola nel seguente modo: $P(B|A) = \frac{A \cap B}{A}$, questo perché non è altro che la proporzione di $A \cap B$ in A.

Formula di Bayes

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(A)}$$

La formula di Bayes è relativamente semplice, si tratta di una rimodulazione delle formula di probabilità condizionale, concretizzata sfruttando il metodo della sostituzione.

1.1.3 Variabili aleatorie continue

Intuitivamente, le variabili casuali continue sono quelle che possono assumere un insieme continuo di valori, al contrario delle distribuzioni discrete, per le quali l'insieme dei possibili valori ha cardinalità al più numerabile. Inoltre, mentre per una distribuzione discreta un evento con probabilità zero è irrealizzabile (come, ad esempio, ottenere $3\frac{1}{2}$ da un lancio di un dado tradizionale), questo non è vero nel caso di una variabile casuale continua. Ad esempio, misurando la lunghezza di una foglia di quercia, è possibile ottenere il risultato $3\frac{1}{2}$ cm, ma questo ha probabilità zero poiché vi sono infiniti possibili valori tra 3 cm e 4 cm. Ognuno di questi ha probabilità zero, ma la probabilità che la lunghezza della foglia sia nell'intervallo (3 cm, 4 cm) è non nulla. Questo apparente paradosso è causato dal fatto che la probabilità che una variabile casuale X assuma valori in un insieme infinito, come un intervallo, non può essere calcolata semplicemente sommando la probabilità dei singoli valori.

1.1.4 Phrase table

Nei modelli statistici, l'ingegnere coinvolto nella progettazione costruisce una sorta di Lookup Table (LUT). La tabella generalmente contiene l'elenco di tutte le frasi più frequenti ed è creata a partire dal corpus. La regola generale è: "maggiore è la frequenza con cui una frase è ripetuta all'interno del corpus, più alta è la probabilità che la traduzione sia corretta". Una frase rientra in un range, generalmente è composta da almeno una parola, fino ad un massimo di cinque. La tabella appena descritta prende il nome di Phrase table ed è il nostro modello di traduzione.
L'ingegnere MT costruisce un modello di traduzione utilizzando la frequenza delle frasi che appaiono nel corpus di formazione in una tabella. Questa tabella memorizza la frase e il numero di volte che si ripete nell'intero corpus di formazione. Più una frase viene ripetuta in un corpus di addestramento, più è probabile che la traduzione di destinazione sia corretta. Ogni frase (memorizzata nella Phrase Table) può avere una lunghezza compresa tra una e cinque parole. Questa tabella delle frasi è denominata modello di traduzione.

In seguito alla creazione della table, è richiesta l'implementazione di un modello di probabilità, che si adatti alle richieste. In particolare, il modello cerca di eseguire un adattamento attraverso le densità probabilistiche parametrizzate, tra le più utilizzate troviamo la variabile normale o multi-normale. La densità di Gauss è molto efficace nell'identificare la frequenza di utilizzo di una parola o frase, anche per via dell'espressività grafica, la quale permette anche ad occhio di individuare il punto medio. Dal momento che spesso in modelli complessi si lavora in più dimensioni, è possibile che venga utilizzata la densità multi-normale, la quale ha tanti layer quante le dimensioni della variabile continua.

Il modello probabilistico è in continua evoluzione, in quanto il corpus

viene perfezionato ed adattato dopo ogni esecuzione di traduzione per eliminare/aggiustare eventuali anomalie. Di conseguenza, il corpus con l'aumentare dell'utilizzo, incrementa l'efficienza, si tratta dunque di un processo evolutivo.

Infine, ultimato il modello probabilistico, ne verrà creato un altro, sulla base dei dati di traduzione, quindi il suo input è in parte composto dall'output generato precedentemente. Questo modello gestisce il corretto assemblamento delle frasi, per ottimizzare la fluidità della traduzione, rendendolo più vicino al flusso di linguaggio naturale. In particolare, si vuole differenziare la traduzione di una singola parola isolata, dalla traduzione di un testo, le quali parole possono cambiare in funzione del contesto.

1.2 Neural machine translation - NMT

La neural machine translation (in italiano traduzione automatica neurale (NMT)) è un approccio alla traduzione automatica che utilizza una rete neurale per prevedere la probabilità di una sequenza di parole, modellando in genere intere frasi in un unico modello integrato. La creazione del modello sfrutta interamente il machine learning, dunque è in grado di apprendere autonomamente sulla base dei training alla quale prende parte.

Differenze dai modelli tradizionali

- Rispetto ai modelli di traduzione tradizionali, chiamati SMT(statical machine translation), sfruttano una quantità infinitesimale di memoria;
- grazie all'utilizzo del deep learning nella creazione della rete neurale, è possibile optare per un addestramento del tipo end-to-end,

questo permette di semplificare la pipeline di sviluppo, con conseguente aumento di prestazioni non indifferente. Per andare più nello specifico:

* una pipeline tradizionale ha questo aspetto:
  voce(input) → riduzione del rumore (noise) → estrazione dei fonemi → composizione delle parole → trascrizione
* una pipeline che sfrutta il deep learning si presenta invece così:
  voce(input) → Deep neural network (DNN) → trascrizione

Osservando le pipeline proposte, è possibile osservare che l'approccio tradizionale porta con sè una complessità non indifferente, ogni singolo layer citato deve essere ottimizzato separatamente, attraverso un preciso criterio. La pipeline creata a partire da una rete neurale invece, semplifica la comunicazione fra i due agenti(input → output desiderato).

1.3 Recurrent neural networks

Prima di descrivere in maniera analitica e precisa la struttura che permette la costruzione di un modello di encoding-decoding è necessario definire le RNN, o Recurrent Neural Networks. Le RNN sono delle reti neurali molto interessanti, perché a differenza dalle tradizionali straight forward neural networks, in cui l'informazione può andare solo in un verso (in avanti), in questo tipo di reti i nodi possono essere interconnessi ai livelli precedenti, ammettendo dei loop. Il fatto che i neuroni (o nodi, intesi come elementi che compongono la rete neurale) possano propagare il segnale in tutte le direzioni, permette di introdurre intrinsecamente il concetto di memoria. Questo perché, l'output di un neurone può ipoteticamente influenzare sè stesso in un diverso istante temporale (rispetto al presente). In particolare, una RNN richiede, ol-

tre all'input tradizionale, lo hidden state, che non è altro che l'output prodotto dallo stesso neurone al passo precedente. La tecnologia è stata presentata nell'anno 2014, in un articolo della Cornell University, il quale ha rivoluzionato lo stato dell'arte in ambito neural networks [3].

1.3.1 Perché sfruttare le RNN?

Le RNN, grazie alla memoria, possono operare su dati dinamici mentre le SNN (straight forward neural networks) operano unicamente su dati statici. Grazie a questa proprietà, le reti neurali ricorrenti, sono in grado di trattare delle sequenze temporali, variabili nel tempo. Un esempio di applicazione può convenire nella interpretazione del linguaggio dei segni. In questo ambito applicativo non è richiesto solo di riconoscere la mano, bensì di comprendere la sequenza di movimenti per poter dedurre ed interpretare il significato dei gesti. Quest'ultima constatazione ci rende in grado di percepire l'importanza di questo tipo di rete neurale nell'ambito delle traduzioni automatiche, in quanto si ha a che fare con sequenze di parole, non necessariamente definite in maniera statica che, per dare un senso logico alla frase, necessitano un meccanismo che sfrutti la memoria (e non solo). Per concludere l'approfondimento, considero fondamentale precisare che, by default, la memoria di un neurone è piuttosto breve, dunque gli output generati dai primi layer, difficilmente condizioneranno quelli presenti in livelli più avanzati. A questo proposito sono state studiate delle reti neurali a lunga memoria, quali le LSTM (Long Short Term Memory) e le GRU (Gated Recurrent Units).

Long short term memory

L'acronimo LSTM fa analogia al fatto che questo tipo di rete neurale permette di implementare una memoria a breve termine estesa. Significa che, a differenza di una RNN tradizionale, è in grado di effettuare

le proprie scelte future, ricordando iterazioni effettuate anche migliaia di step precedenti rispetto all'attuale. Una unità LSTM è composta da:

- una cella;
- un input gate, che ricevendo in ingresso oltre all'input tradizionale, il valore di uscita ottenuto alla iterazione precedente, modifica ed elabora la cella;
- un output gate, il quale attraverso una predizione, è in grado di generare in output il nuovo hidden state, avendo a disposizione gli input delle iterazioni precedenti;
- un forget gate, che elabora un semplice meccanismo di attention, classificando le informazioni parametrate in base alla loro utilità. Le informazioni non utili le ignorerà, l'implementazione del meccanismo avviene attraverso l'assegnazione di un peso per ogni elemento, all'aumentare del punteggio maggiore è l'attenzione che bisogna porre.

Gather reccurent units

La rete neurale GRU (Gather recurrent units) sfrutta un approccio molto simile ad LSMT per quanto riguarda l'implementazione di un meccanismo in grado di estendere la capacità di memorizzazione. Rispetto ad LSMT richiede una quantità minore di parametri, dal momento che non ha a disposizione un output gate. Il suo utilizzo è consigliato su dataset di piccoli dataset e reti neurali poco profonde.

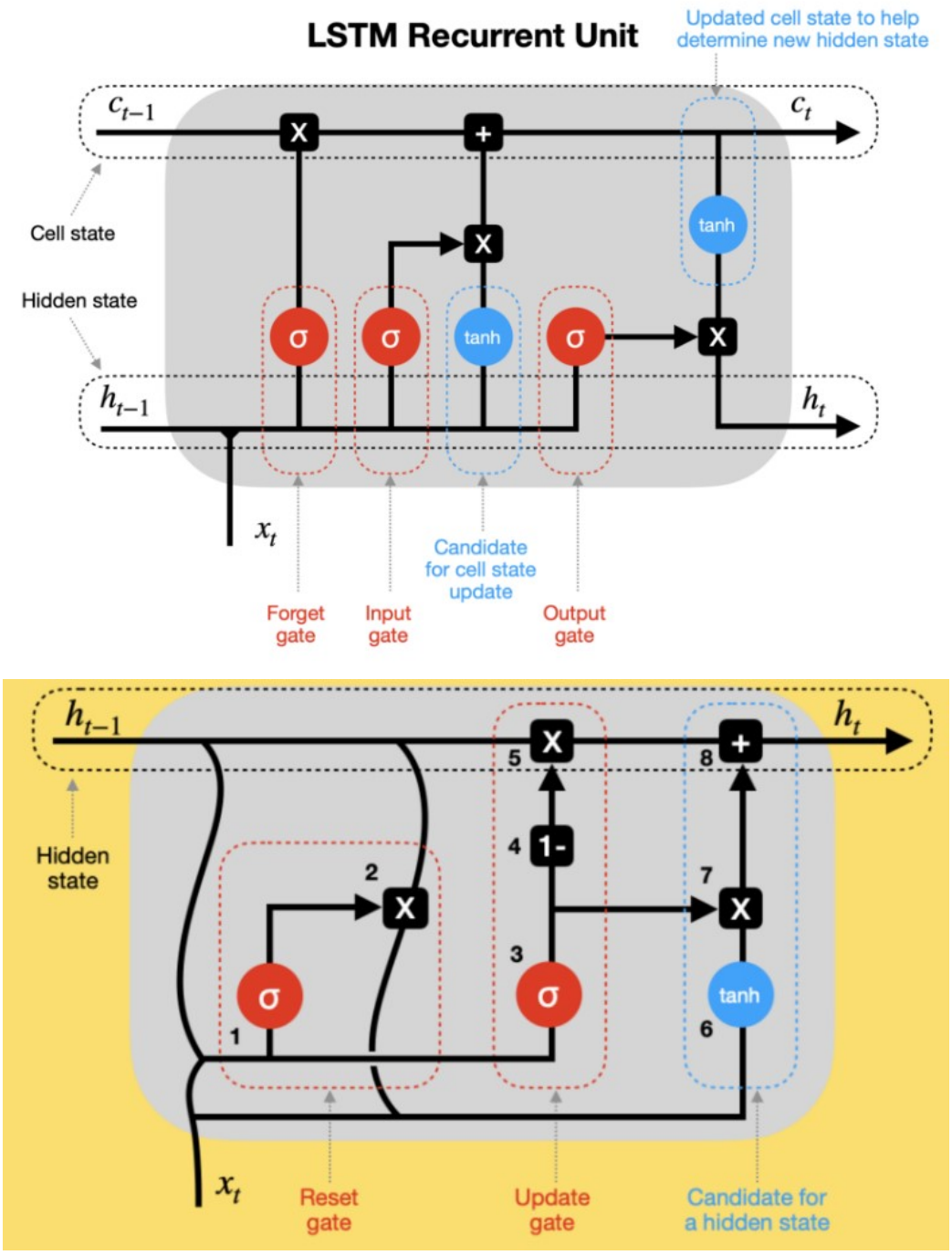

Figura 1.1: Rispettivamente LSMT (in bianco) e gru (in giallo), schema esemplificativo [8]

1.4 Transformers

Nell'ambito del NMT, il modello maggiormente utilizzato è il transformer, che negli ultimi anni ha preso il posto delle reti neurali RNN, descritte nella sezione precedente.

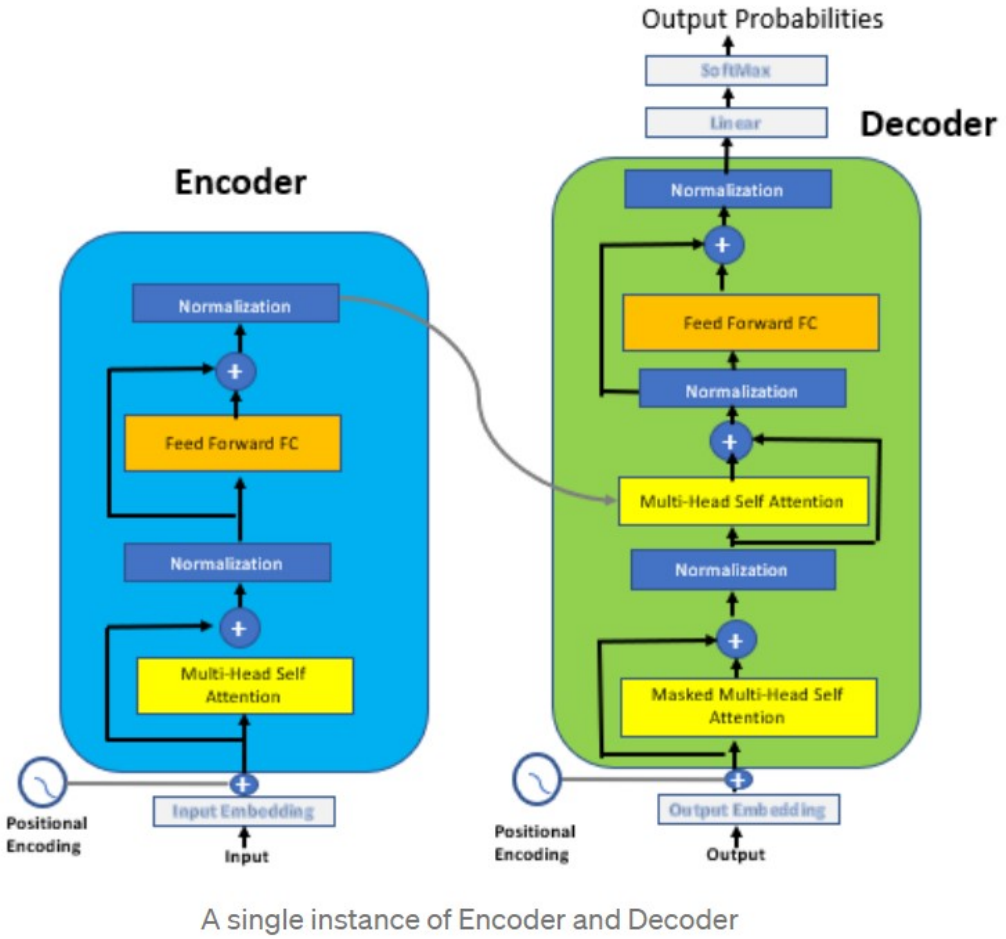

Figura 1.2: Struttura esemplificativa di una singola istanza di un transformer, formato da encoder e decoder [8]

Caratteristiche e peculiarità

La tipologia di modelli citati in questa sezione sono nati con l'obiettivo di evitare la ricorsione, facendo spazio a delle tecnologie che garantissero un tipo di computazione che ammettesse il parallelismo. Le caratteristiche principali sono le seguenti:

- data una sequenza di dati in input, a differenza delle reti RNN, che operano word-by-word (ossia hanno la necessità di analizzare ogni singolo elemento della struttura), il transformers sono in grado di processare l'input **"as a whole"**, ossia come un unico grande dato. Questa caratteristica è molto importante nell'ambito delle traduzioni automatiche, in quanto rende molto più intuitivo individuare il contesto della frase;
- l'introduzione della **self-attention**, che consente un notevole miglioramento nella predittività, ha dato il nome all'articolo che ha presentato questa tipologia di modelli, ovvero "Attention is all you need" [1]. L'attention, volendo semplificarne il funzionamento, concettualmente consta nell'utilizzo di unità logiche che, attribuiscono un punteggio ad ogni parola, il punteggio è un valore che indica la probabilità che la parola sia la successiva all'interno di una frase;
- il **positional embedding**: si tratta di una tecnica che, insieme all'attention, ha come obiettivo quello di eliminare i processi che sfruttano la ricorsione. In particolare, l'idea è di assegnare dei pesi legati alla posizione di una parola all'interno della frase, in modo da verificare quali siano le tuple di parole che hanno più probabilità di essere accostate.

Il primo punto è fondamentale perché consente al modello di non dipendere da neuroni di layer lontani, processando l'intero input in una sola volta, viene eliminato il concetto di perdita di memoria. Il tutto

può essere chiarito attraverso un semplice esempio:
Consideriamo un ipotetico articolo, supponendo che i puntini rappresentino il testo di intermezzo:

La nazionale italiana di basket ..... Pozzecco ha deciso di intervenire in sala stampa, ringraziando lo staff.

Dato questo input, il transformer, analizza l'articolo nel suo intero, mentre un modello tradizionale considera parola per parola, ovvero: {0:La, 1:nazionale, 2: italiana, 3: di, 4: Basket, ..., 1200:Pozzecco }. A giudicare da questo esempio si può osservare che è impossibile mantenere il contesto analizzando una parola per iterazione, specialmente se si ha necessità trovare un nesso fra due parole con indici molto distanti.

Descrizione tecnica dettagliata del modello

Si tratta di un sistema di encoding-decoding. In linea generale, l'encoder mappa l'input ricevuto in un vettore continuo, che contiene tutte le informazioni apprese dai dati ricevuti in ingresso. Il decoder a partire dal vettore continuo, passo per passo, genera un unico output. Il singolo output generato è al contempo condizionato dagli ulteriori vettori continui dati in pasto al decoder in precedenza. Questo passaggio avviene ricorsivamente, fino a quando il decoder non incontra il token <end>, rappresentante la end of sentence (in italiano, fine della frase).

Input embedding

Il primo passo è passare il nostro input al word embedded layer, il quale può essere considerato come una lookup table che contiene dei fattori di rappresentazione per ogni parola. In particolare, ogni parola viene mappata in un vettore avente valori continui, che la rappresentano, dal momento che il sistema apprende attraverso pattern numerici.

Positional encoding

In seguito alla definizione dell'embedded layer è necessario passargli delle informazioni circa la collocazione di ogni vettore all'interno della rete neurale, dal momento che non è più possibile sfruttare la ricorsione a questo scopo. Per fare ciò è stata studiata una brillante soluzione, che verte sull'utilizzo di queste due formule:

- $P(pos, 2i+1) = \cos(\frac{pos}{10.000^{2i/dmodel}})$
- $P(pos, 2i) = \sin(\frac{pos}{10.000^{2i/dmodel}})$

Per ogni step di esecuzione di indice pari, genera un vettore di valori continui attraverso la funzione coseno. Per quelli di indice dispari invece, il vettore viene generato sfruttando il seno. Ad ogni vettore risultante viene sommato l'embedding vector corrispondente (ottenuto al primo step). La scelta è ricaduta sulle funzioni sin e cos per via delle loro proprietà lineari, che l'intelligenza artificiale è in grado di apprendere con maggiore facilità.

1.4.1 Encoding layer

Come introdotto inizialmente, l'encoder mappa l'input ricevuto in un vettore numerico contenente valori continui, rappresentanti le informazioni apprese dai dati ricevuti in ingresso. Contiene due sottomoduli:

- il primo è una unità dedicata alla self- attention chiamata **multi head attention layer**. Andando nello specifico, ;
- l'altro è una vera e propria **rete neurale feed forward.**

I due layer sono collegati fra loro non direttamente, fra loro si interpone un layer intermedio che si occupa della normalizzazione dell'output.

Multi-head attention layer

L'unità citata nel titolo, implementa uno specifico meccanismo di attention chiamato self-attention. Questa tecnologia consente ad un modello, attraverso principi combinatori accurati, di associare le parole presenti nell'input. Ad esempio, supponendo di avere la frase {"ciao", "come", "stai"}, l'unità potrebbe ipoteticamente correlare : {"ciao", "come"} oppure {"stai", "ciao"}. Attraverso questo tipo di associazioni, il modello può ad esempio interpretare la tipologia di frase, rendendosi conto in questo caso, di avere a che fare con una domanda. Queste intuizioni, permettono alla rete neurale di strutturare un possibile output. Perché il meccanismo funzioni l'input deve essere passato come parametro a tre distinti layer, fra loro connessi, i quali genereranno i seguenti vettori continui:

- query vector;
- key vector;
- value vector.

In particolare il query vector contiene la richiesta, generalmente in formato json, che verrà etichettata dal key vector. Questa operazione è molto simile a quella che viene effettuata durante una richiesta ad un motore di ricerca, come ad esempio quello di youtube. Quando viene effettuata la ricerca, google etichetta il testo inserito dall'utente, in particolare associa la ricerca a descrizione, autore o titolo del video. Finita l'etichettatura, interroga il database, cercando i migliori match, ovvero i video strettamente compatibili con la richiesta.
Il vettore delle query dunque, viene moltiplicato per il vettore delle chiavi (o key vector), ottenendo come risultato la matrice degli scores, o punteggi. Le celle della matrice risultante contengono un valore numerico, il quale indica quanto forte sia il collegamento fra una parola e l'altra, ovvero l'attenzione che occorre porre. Quindi, semplificando

il concetto, all'interno di un arco temporale ogni parola avrà a disposizione un punteggio in relazione ad ogni altra parola presente. Il focus aumenta con l'aumentare del valore contenuto nella cella.

81	18	91
509	1000	230
300	200	3300

Tabella 1.1: Ipotetica matrice degli scores, per semplicità utilizzerò la sua nomenclatura originale, ovvero score matrix.

Osservando la matrice in formato tabellare sovrastante, si nota subito che l'ordine di grandezza del contenuto delle celle necessita una sorta di normalizzazione, per evitare che, la moltiplicazione con altre matrici, generi valori troppo grandi. A questo proposito si effettuano le seguenti operazioni:

1.

$$\frac{scorematrix}{\sqrt{d_k}}$$

Divido la matrice per la dimensione delle query e delle keys;

2.

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

In questo passaggio, viene presa come input la matrice ottenuta al passo precedente, (x_i è un elemento arbitrario). La matrice risultante chiamata sealed matrix, conterrà per ogni cella un valore probabilistico, il quale range andrà da 0 ad 1. Alle probabilità più alte verrà attribuita una maggiore attenzione (da qui, il termine attention), mentre quelle inferiori alla soglia verranno ignorate, perché non utili. Questo permette al modello di aumentare la confidenza circa la scelta delle parole sulla quale fare affidamento.

3.

$$sealedmatrix \cdot valuematrix = output$$

Moltiplico la matrice ottenuta al passaggio precedente, che può essere considerata una matrice di attention, per il vettore dei valori definito inizialmente. L'output, come già accennato, oscurerà le parole con un punteggio softmax più basso, dando la priorità alle altre.

4.

Infine nell'ultimo passaggio, l'*output* ottenuto al passaggio precedente viene passato in input ad un layer lineare, che sarà in grado di processarlo.

Perché si parli di multi-headed attention, è necessario che, i vettori presi originariamente in input (query, key e value vectors), vengano scissi n volte e ogni tripletta di vettori ottenuta, attraversi lo stesso processo di self-attention individualmente. La scissione dei vari array, si basa su criteri prevalentemente dimensionali, in quanto, a partire dal vettore originale, lungo *length*, si ricavano n array, che rappresentano una divisione dello stesso in n sottoarray. Ogni singolo processo di self-attention prodotto prenderà il nome di head, da questo deriva il nome di multi-headed attention layer.

Il seguente esempio può chiarire il concetto riguardante la scissione:

- array originali: $q = \{"hi", "how", "are", "you", "my", "friend"\}$, $k = \{1, 2, 3, 4, 5, 6\}$, $v = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ con singolo layer di self attention h, potrebbe diventare:
  - $\{q_1 = \{"hi", how"\}, k_1 = \{1, 2\}, v_1 = \{0.1, 0.2\}\} \rightarrow head_1$
  - $\{q_2 = \{"are", "you"\}, k_2 = \{3, 4\}, v_2 = \{0.3, 0.4, \}\} \rightarrow head_2$
  - $\{q_3 = \{"my", "friend"\}, k_3 = \{5, 6\}, v_3 = \{0.5, 0.6\}\} \rightarrow head_3$

  Abbiamo dunque ottenuto tre tuple di array, di cardinalità 3 e ad ogni tupla è associata una unità individuale di self-attention.

Gli output prodotti da ogni singolo processo derivante dalla scissione, al termine della operazione verranno concatenati in un unico oggetto, che sarà a tutti gli effetti l'input del layer lineare, introdotto nell'ultimo punto della sottosezione precedente. Il fatto di avere a disposizione un input generato da n diversi processi di self-attention, fa in modo che, ogni head unit impari attraverso processi logici diversi e concatenandoli, si è in grado di offrire al modello un potere di rappresentazione più spiccato.

Per riassumere Il multi-headed-attention è un modulo, compreso in una rete di trasformatori, che riceve in input i pesi relativi all'attention (o attention weights) e produce un output, che contiene informazioni codificate. Le informazioni dicono al modello con quale frequenza e modalità le parole di una sequenza, devono essere associate.

Residual connections

Le residual connections (o connessioni residue), sono appartenenti ad un tipo di rete neurale chiamata **residual neural networks**. Si tratta

di una rete la quale, attraverso le proprie connessioni, è in grado di operare non considerando nel processo i layer intermedi.Saltare skip di layer nell'esecuzione potrebbe risultare controproducente, allo stesso tempo però si può osservare che lo skip di layer:

- risolve il noto ed importante problema dei **vanishing gradients**, ottimizzando la rete neurale. Nell'apprendimento automatico, il problema del Vanishing gradient si incontra quando si addestrano reti neurali con metodi di apprendimento basati sul gradiente e sulla back-propagation. In tali metodi, durante ogni iterazione dell'addestramento, ciascuno dei pesi della rete neurale riceve un aggiornamento proporzionale alla derivata parziale della funzione di errore rispetto al peso corrente. Il problema è che in alcuni casi il gradiente sarà di dimensioni infinitesimali, impedendo al peso di cambiare valore in maniera ottimale;
- **mitiga il problema di degradazione, meglio noto come accuracy problem (problema di precisione)**. La degradazione si verifica quando si ha a che fare con una rete con un grande numero di layer, perché è formalmente dimostrato che, aumentare il numero di strati, comporta un margine di errore più ampio.

Nel trasformer, l'importanza delle connessioni residue si può percepire dopo aver studiato ed analizzato con precisione tutti i singoli passaggi, indicati nel prossimo paragrafo.

Si parte unendo l'output generato dal **multi-level attention layer** all'input originale. Il risultato ottenuto sarà poi normalizzato, passando attraverso il normalization layer. Supponendo che $multiLevelHeadOutput$ sia l'output ottenuto dal multi-headed layer e che $input$ sia l'input originale,dato in pasto allo stesso, il tutto può essere formalizzato attraverso la formula:

- $LayerNorm(ResidualConnection = multiLevelHeadOutput, input)$

Ciò che ottengo dalla normalizzazione, passa attraverso una rete neurale feed-forward, che dunque non ammette backtracking e non ha memoria, nemmeno a breve termine. La rete neurale artificiale citata, è composta da 2 layer lineari, come quelli visti in precedenza, con la differenza che fra essi, viene collocato un layer ReLu (Rectified Linear Unit). ReLu è una funzione di attivazione non lineare, molto nota in ambito deep learning. Si tratta di una sigla che sta per Rectified Linear Unit, semplificabile in "rettificatore". La particolarità della funzione è che attiva **solo un** neurone alla volta e non gli è concesso attivarli tutti contemporaneamente.

Il valore di uscita dalla rete feed forward, verrà a sua volta legato all'input originale, formando un'ulteriore connessione residua, che verrà poi normalizzata da un layer di normalizzazione. Incorporare al modello una rete feed-forward serve per elaborare ulteriormente l'output della attention, dandogli potenzialmente una rappresentazione più ricca.

Questo ultimo tassello conclude la logica procedurale del modello encoder, il quale scopo, come già spiegato nell'introduzione del capitolo, è di offrire al decoder delle indicazioni da seguire, circa le parole alla quale deve essere dedicata una maggiore attenzione. Come con l'attention è possibile creare uno stack di differenti encoder, dove ogni singolo elemento della pila ha possibilità di imparare diverse rappresentazioni dell'attention. Quest'ultimo aspetto è molto interessante, perché aumenta il potere predittivo del rete del modello.

1.4.2 Decoder

Il ruolo del decoder consiste nel generare sequenze di testo, la sua struttura è molto simile a quella dell'encoder. In particolare, ha a

sua disposizione:

- 2 multi-headed attention layers;
- un feed fordward layer, ovvero una rete neurale senza backtracking;
- un sub-layer di normalizzazione per ogni layer padre presente, oltre alla presenza di residual connections derivanti dall'utilizzo di una residual neural network (rnn);

Si tratta a tutti gli effetti di un modello autoregressivo (in inglese autoregressive model). Il decoder infatti, prende in ingresso gli output generati dall'encoder, oltre alle informazioni sull'attention degli stessi. Il layer elabora richieste fino a quando non viene generato lo <end> token, il quale indica la fine della frase.

Modello autoregressivo

Un modello autoregressivo si fonda sulla misurazione della correlazione fra osservazioni, effettuate ad un istante temporale antecedente rispetto al tempo di misurazione, allo scopo di predirre il valore che la variabile considerata assumerà al prossimo step (il nostro output). Nel caso in cui la variabile di input e output varino nella stessa direzione, ovvero incrementando o diminuendo insieme, si otterrà una correlazione positiva. Al contrario, se la variazione segue il senso opposto, la correlazione sarà negativa. All'aumentare del valore di correlazione, che sia negativo o positivo, aumenta la probabilità che attraverso l'input la macchina sia in grado di prevedere il futuro. Dal momento che la correlazione è fra la variabile al presente e sè stessa al passo precedente, si parla di autocorrelazione.

1.4.3 Passaggi effettuati nella fase di decoding

Embedding dell'output e positional encoding

L'input passa attraverso due layer, si tratta dell'embedding layer e il positional encoding layer. Il procedimento denota la posizione delle parole analizzate rispetto alla frase presa come riferimento.

Decoder multi headed attention 1

L'output ottenuto dal passaggio precedente viene dato in pasto al primo dei due multi headed attention layer. In questa fase, il componente ha lo scopo di estrarre i punteggi relativi all'attention dell'input. Non bisogna confonderlo con il multi-headed attention layer presente nella struttura dell'encoder, questo perché, l'autoregressione richiede soluzioni leggermente diverse. In particolare, dal momento che processa l'input parola per parola è ritenuto necessario evitare che token futuri vengano condizionati. Ad esempio, supponendo che il modello stia analizzando la parola "tutto", della frase { "Io", "tutto", "bene"; "tu"?}, esso non deve avere accesso alle parole successive, perché generate in un istante futuro. Detto ciò, quindi, è importante fare in modo che la computazione di una parola, possa essere condizionata solo dalle precedenti (generate anteriormente), non dalle successive (generate successivamente). Il metodo che rende possibile tutto ciò prende il nome di masking, e segue il seguente funzionamento:

- per evitare che il decoder prenda in considerazione token futuri, viene applicata una "look ahead mask" prima di calcolare il softmax della matrice dei valori e dopo la normalizzazione degli score (osservare la sezione della attention per chiarire). La maschera è una matrice delle stesse dimensioni della sealed matrix, ovvero la matrice degli score in seguito alla normalizzazione. Le celle sopra

la diagonale assumono valore - ∞, mentre le altre valore 0. Il risultato è la matrice chiamata **masked scores**.

– il motivo per il quale si applica questo tipo di maschera, è legato al fatto che, applicando il softmax (visto nei capitoli precedenti), i valori con -∞ verranno sostituiti da zeri. La matrice risultante valorizza il punteggio di attention che verrà attribuito ad ogni token, a questo proposito, abbiamo che azzerando la parte sopra alla diagonale della matrice, stiamo implicitamente escludendo i token futuri dal calcolo.

0.7	*	*		0	$-\infty$	$-\infty$		0.7	$-\infty$	$-\infty$
0.5	0.1	*	+	0	0	$-\infty$	=	0.5	0.1	$-\infty$
0.3	0.2	0.2		0	0	0		0.3	0.2	0.2

Tabella 1.2: L'esempio sovrastante ci mostra l'applicazione della maschera alla matrice degli score (post normalizzazione). In particolare, alla matrice degli score viene sommata la maschera (matrice centrale), che va ad incidere solo sui valori situati sopra la diagonale. Gli asterischi presenti nella matrice di sinistra, vanno ad indicare valori randomici, non è importante conoscerli, dal momento che diventeranno pari a - ∞

.

$$softmax\left(\begin{array}{|c|c|c|}\hline 0.7 & * & * \\ \hline 0.5 & 0.1 & * \\ \hline 0.3 & 0.2 & 0.2 \\ \hline\end{array}\right) = \begin{array}{|c|c|c|}\hline 0.7 & 0 & 0 \\ \hline 0.5 & 0.1 & 0 \\ \hline 0.3 & 0.2 & 0.2 \\ \hline\end{array}$$

Tabella 1.3: Applicazione della funzione softmax alla matrice ottenuta in seguito al masking

.

Multi headed attention layer 2

Riceve in input il vettori delle query e values derivanti dall'encoder, mentre il vettore degli scores è l'output ottenuto dal primo layer di multi head attention.

Feed forward linear layer (Classifier)

L'output ottenuto in seguito all'elaborazione delle informazioni a carico del secondo layer di attention, passa attraverso una rete feed forward, la quale accede ad un classifier. A classifier in machine learning is an algorithm that automatically orders or categorizes data into one or more of a set of "classes." One of the most common examples is an email classifier that scans emails to filter them by class label: Spam or Not Spam. Il classifier (o classificatore, in italiano), in questo caso, crea una classe per ogni singola parola data in input, quindi, per chiarire il concetto, data in input una frase composta da 500 parole, il classificatore la mapperà in 500 diverse classi, indicizzate e con identificatore univoco. L'output del classificatore viene poi immesso nel softmax layer, il quale ha prodotto i punteggi di probabilità compresi tra zero e uno per ciascuna classe. Infine il modello prenderà la classe con il punteggio di predittività più alto, restituendolo in output, si tratta del risultato della predizione del modello. La parola risultante verrà poi aggiunta all'elenco degli input del decodificatore, il quale continuerà di nuovo la decodifica fino a quando non incontrerà il token di fine, la quale classe ha la previsione di probabilità più alta. Il decoder, similmente all'encoder, può essere impilato e sovrapposto, ogni strato riceve l'input dall'encoder e dai livelli precedenti. Impilando i livelli, il modello può imparare a estrarre e concentrarsi su diverse combinazioni attraverso l'attention, aumentando potenzialmente il suo potere predittivo.

Capitolo 2

Tecnologie

In questo capitolo verranno discusse le tecnologie sfruttate, al fine di costruire e testare un modello di traduzione automatico, che possa tradurre del testo da italiano a cinese, o viceversa. Nella prima parte del capitolo verranno discusse i framework utilizzati per eseguire un confronto con il modello da me creato. Nella seconda parte invece, verranno discussi e analizzati gli strumenti utilizzati per la realizzazione del modello.

2.1 Servizi cloud per traduzione automatica

Per verificare la correttezza del modello implementato, è stato necessario studiare ed analizzare i più famosi framework di traduzione automatica, i quali lavorano sfruttando il cloud. In particolare ci si riferisce a:

- DeepL [12];
- ModernMt [13];
- Google translator [14];

– Azure cognitive service (Microsoft) [15].

Le tecnologie citate, funzionano similmente, ovvero attraverso delle richieste https alla loro restful api. RedHat [16] precisa che, perché una api possa essere considerata restful, è necessario che rispetti un insieme di vincoli architetturali. Dunque è richiesto che:

– venga implementata un'architettura di tipo client-server, con approccio stateless. Stateless significa che, ogni richiesta HTTP, avviene in maniera completamente isolata. Quando il client effettua una richiesta, essa contiene tutte le informazioni necessarie al server per rispondere correttamente. Il server non si affida mai ad informazioni derivanti da richieste effettuate in precedenza dal client;
– sia presente una cache;
– a livello architetturale, il progetto sia gestito in layer, organizzati gerarchicamente.

2.1.1 ModernMT

ModernMT è un progetto open source che integra una tipologia di traduzione adattiva, real-time e basata sul machine learning, in un unico prodotto. La peculiarità del framework sta nel fatto che, la qualità delle traduzioni è direttamente proporzionale al numero di interazioni con l'utente. Questo significa che, il modello di MT è in grado di interpretare e contestualizzare le richieste effettuate dall'utente, in maniera sempre più accurata. Il tutto è stato reso possibile integrando ad una rete neurale generica, una memoria dinamica, la quale ha lo scopo di memorizzare lo storico delle traduzioni richieste da un determinato utente. Quando ModernMT riceve una request, ne analizza il contesto, recupera dalla memoria dinamica la traduzione più simile e adatta la sua rete neurale alla query.

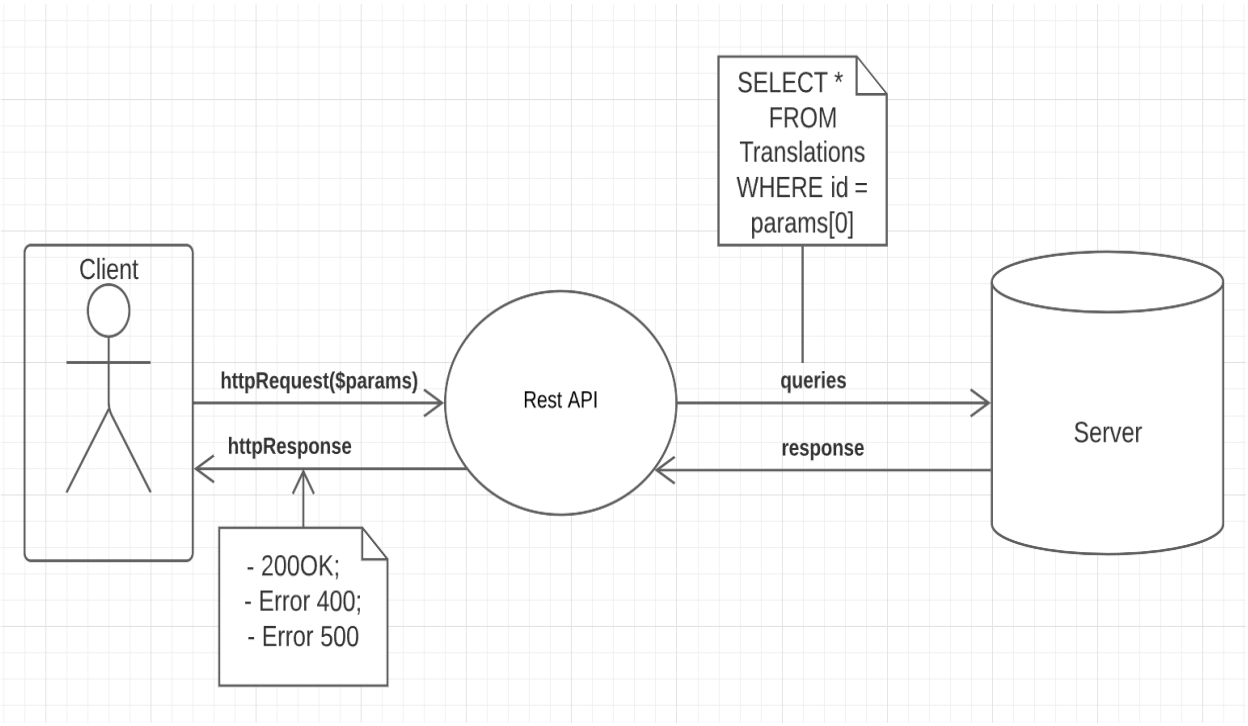

Figura 2.1: Rappresentazione grafica del funzionamento di un sistema che interroga una REST API.

Requests & responses

Come già anticipato nell'introduzione, ModernMT mette a disposizione dell'utente, una REST API, protetta da una chiave, ottenibile solo in seguito all'iscrizione al servizio. In caso di successo della request effettuata dall'utente, viene ritornato un oggetto di tipo json all'interno della response, il quale contiene la traduzione effettuata. Una response http ritorna inoltre un codice status, che assume il valore:

- 200, se la comunicazione è avvenuta correttamente;
- 400, se il server non riesce ad elaborare la richiesta effettuata lato client, che è dunque formulata in maniera errata;
- 500, se il server è in stato di errore e non è in grado di elaborare risposte.

Memorie

Le memorie sono delle entità che memorizzano delle frasi con la corrispettiva traduzione. Vengono sfruttate attraverso i context vector, per effettuare un adattamento real-time. Al fine di ottenere i migliori risultati è opportuno classificare le memorie, in funzione del dominio d'interesse.

Traduzione

Per richiedere una traduzione dunque, è necessario comporre una request, in cui oltre al testo da tradurre, è possibile strutturare dei parametri. In particolare, ci si riferisce ai seguenti:

- source language:
  - ovvero la lingua del testo fornito in input. Se omessa, l'intelligenza artificiale la individua in autonomia, con una buona accuratezza;
- target language:
  - si tratta della lingua in cui ci si aspetta la traduzione;
- context vector:
  - è una lista di memorie, ciascuna delle quali caratterizzate da un peso. Il peso determina l'influenza che i dati presenti nella memoria hanno nei confronti del risultato della traduzione. Il vettore è un'unica grande stringa, in cui sono presenti coppie chiave-valore, separate fra loro attraverso la virgola. La chiave è la collocazione logica della cella di memoria, mentre il valore è l'entità del peso, che gravita fra 0 ed 1. Esempio: vector = "2389:0.2, 2489:0.8", il primo elemento ha peso 0.2 e si trova nella cella 2389, mentre il secondo elemento ha peso 0.8 ed è collocato nella cella 2489.

- hints:
  - nel caso in cui venga omesso il context vector, è possibile indicare gli hints, sotto-forma di stringa. Il contenuto della stringa sarà caratterizzato dalla presenza di identificativi di celle di memoria, dunque a differenza del vettore dei contesti, non è presente il peso relativo alla cella. Questo perché a partire dagli hints, ModernMT genera automaticamente i pesi da attribuire ad ogni porzione di memoria.
- multi-line:
  - il seguente parametro, se attivo, dato un testo di gradi dimensioni, lo suddivide in frasi.
- timeout:
  - è possibile indicare in millisecondi, dopo quanto tempo di attesa si consideri invalida una request. Il valore di default in genere si aggira intorno ai 3 minuti complessivi.

Risultato traduzione

Il risultato della traduzione, è un oggetto json che contiene, oltre al testo tradotto, altri parametri, tra i quali: il context vector utilizzato (per garantire trasparenza), il numero di caratteri del testo di input e del testo di output.

2.1.2 Azure translation service

Il servizio di traduzione messo a disposizione da Microsoft, attraverso una REST API, fa parte della famiglia degli azure cognitive services, che presenta soluzioni pratiche, le quali coinvolgono l'uso di intelligenza artificiale a problemi comuni, come ad esempio la traduzione di testo. Il progetto, nonostante sia di stampo Microsoft, offre un ampio livello di

portabilità, essendo multi-piattaforma. Tendenzialmente è sufficiente operare una richiesta http per usufruire del servizio, però dal momento che è possibile indicare tanti e articolati parametri, è più indicato utilizzare una client library che automatizzi questo tipo di operazione.

Requests

Le requests sono delle chiamate https, in particolare delle GET. Per accedere alla richiesta azure mette a disposizione una chiave privata, la quale viene inserita nel corpo della richiesta, insieme all'area geografica, in questo modo Microsoft è in grado effettuare il reindirizzamento sul data center più vicino.

Traduzione

Il processo di traduzione, richiede una serie di parametri, che differiscono dai sistemi di traduzione convenzionali. Una peculiarità del servizio è la possibilità di accedere al punteggio BLEU di una traduzione, contenuto nel corpo del json ricevuto dalla GET request. Il punteggio BLEU è uno standard internazionale per la traduzione, ne determina l'accuratezza e la precisione.

2.1.3 DeepL

DeepL è un progetto closed-source, nato nella primavera del 2017, dunque, a differenza di ModernMT non è possibile accedere o contribuire allo sviluppo del software.

Controversie

Ha ottenuto sin da subito molta attenzione mediatica perché a detta dell'azienda, le prestazioni del loro traduttore sono addirittura superiori

rispetto alla controparte di Google e Facebook. Un articolo elaborato dall'Università di Bologna [17], a cura di Christine Heiss e Marcello Soffritti ha però smentito il tutto, dimostrando come i risultati presentati sul sito di DeepL fossero in realtà basati su campioni selezionati accuratamente dall'azienda, che oltre ad essere più adatti alle loro reti neurali erano di dimensione molto ridotta (si parla di testi con circa 1000 caratteri) e di conseguenza non dimostra la superiorità nei confronti dei traduttori più noti. Come risaputo, la validità e il potenziale predittivo di un modello è più accurato se testato su un training set più grande, che nel caso della traduzione automatica, introduce inoltre il problema dell'interpretazione del contesto, che in frasi isolate non si presenta. L'articolo dimostra inoltre infatti che, su un testo molto articolato e di grandi dimensioni, il punteggio BLEU risulta più basso rispetto alla concorrenza. Si nota anche che, la capacità interpretativa è limitata, in quanto il sistema non è in grado di intuire il fatto che il testo da tradurre provenga da una forma dialettale, come ad esempio il tedesco parlato dagli abitanti del Trentino Alto Adige.

Peculiarità e potenzialità della tecnologia

DeepL è un progetto molto ambizioso e con grandissimo potenziale, nonostante attualmente, non sia migliore rispetto alla concorrenza. Degno di nota è il fatto che, le dimensioni dell'azienda che ha presentato le soluzioni tecnologiche, è di dimensioni infinitesimali rispetto alle concorrenti. A mio avviso è dunque molto interessante conoscere la struttura tecnologica del progetto, che con infrastrutture relativamente piccole ha ottenuto risultati incredibili. Il servizio offre un corollario di lingue disponibili molto limitato, dovuto al fatto che c'è una ricerca dell'eccellenza della traduzione, piuttosto che una quantità spropositata di combinazioni linguistiche. A questo proposito, uno dei modelli di traduzione più efficaci presentati nel 2020, è quello inglese-cinese, noto per essere uno dei più complessi, in quanto risulta molto difficile

raggiungere risultati accettabili. Dal momento che il trattato analizza un caso di studio circa la traduzione italiano-cinese, DeepL è a tutti gli effetti un ottimo metro di paragone.

Web Api

La REST API messa a disposizione è gratuita, la chiave per accedere al servizio si ottiene in seguito all'iscrizione. Le librerie lato client che facilitano la creazione di richieste http sono disponibili in:

- Java;
- .NET;
- Python;
- Node.js;
- PHP.

Se il testo da tradurre è molto voluminoso viene messa a disposizione la possibilità di sfruttare la parallelizzazione, ovvero invece di una singola request http, vengono elaborate n requests, contemporaneamente, le quali porteranno ad una singola response.

Requests

I parametri delle requests http generate dalle librerie client del servizio, seguono in parte una struttura simile ad altre tecnologie (es: ModernMT). I parametri principali richiesti sono infatti:

- il testo da tradurre;
- la lingua del testo passato in input;
- l'eventuale split del testo in frasi.

Di conseguenza è probabilmente più interessante e utile soffermarsi sulle peculiarità, dunque i parametri caratteristici della tecnologia, i quali non sono presenti in altri framework. Sono i seguenti:

- preserve formatting:
  - si tratta di un flag che, se attivato, concede al modello la libertà di inserire la punteggiatura e la formattazione del testo che viene introdotto in input. Il modello decide inoltre quando è opportuno o meno utilizzare caratteri in maiuscolo o in minuscolo. Di default è disattivato;
- formality:
  - è un parametro che modella una delle feature più interessanti ed innovative della tecnologia. In particolare, è possibile determinare, attraverso una stringa, il livello di formalità della traduzione. Un documento universitario richiede l'utilizzo di formalismi a differenza di una lettera mandata ad un amico, che è generalmente del tutto informale. Sarebbe interessante scoprire quanto incida questa scelta sulle prestazioni del modello, purtroppo essendo codice proprietario si può operare solo empiricamente, osservando i risultati proposti.
- outline detection:
  - è un parametro che se attivato, interpreta il formato del testo da tradurre, cercandolo di riprodurre in output.

Glossari

DeepL permette di creare dei glossari personalizzati, seguendo delle linee guida. Si possono incorporare al modello di traduzione, riportando l'id del glossario all'interno dei parametri della request inviata alla api.

2.1.4 Google translator

Si tratta di uno dei traduttori più utilizzati al mondo [18], è in continua evoluzione, a maggior ragione in seguito della ideazione della neural machine translation. Google mette a disposizione il prodotto non solo attraverso una REST API, bensì offre la possibilità di accedere al servizio con le chiamate di procedura remota (RPC).

gRPC

Il modello gRPC, è una implementazione del paradigma di chiamata a procedura remota, curata da Google. Attraverso chiamate, generalmente http, vengono attivate subrutine su computer remoti, dunque non più in locale, sulla propria macchina.Il paradigma è pensato principalmente per un tipo di comunicazione client-server, la chiamata di subrutine corrisponde alla request inviata dal client, il valore di ritorno invece è la response ricevuta dal server. Viene dunque eseguito un server gRPC per gestire le chiamate dei client, il quale ha uno stub che fornisce gli stessi metodi del server.

gRPC ha numerosi punti di forza, i principali sono:

- La portabilità, dato che le procedure vengono gestite in computer remoti, non si è legati ad un unico linguaggio di programmazione o stack tecnologico. Attraverso delle client library implementate correttamente, è inoltre possibile accedere allo stesso servizio, con gli stessi risultati, potenzialmente con qualsiasi linguaggio esistente (che permetta chiamate http).
- La scalabilità, a questo proposito sono consigliati nella progettazione di sistemi distribuiti;
- La semantica, l'obiettivo è infatti quello di emulare le chiamate a routine locali, nonostante ci si stia interfacciando con altre macchine attraverso la comunicazione di rete. Dal momento che

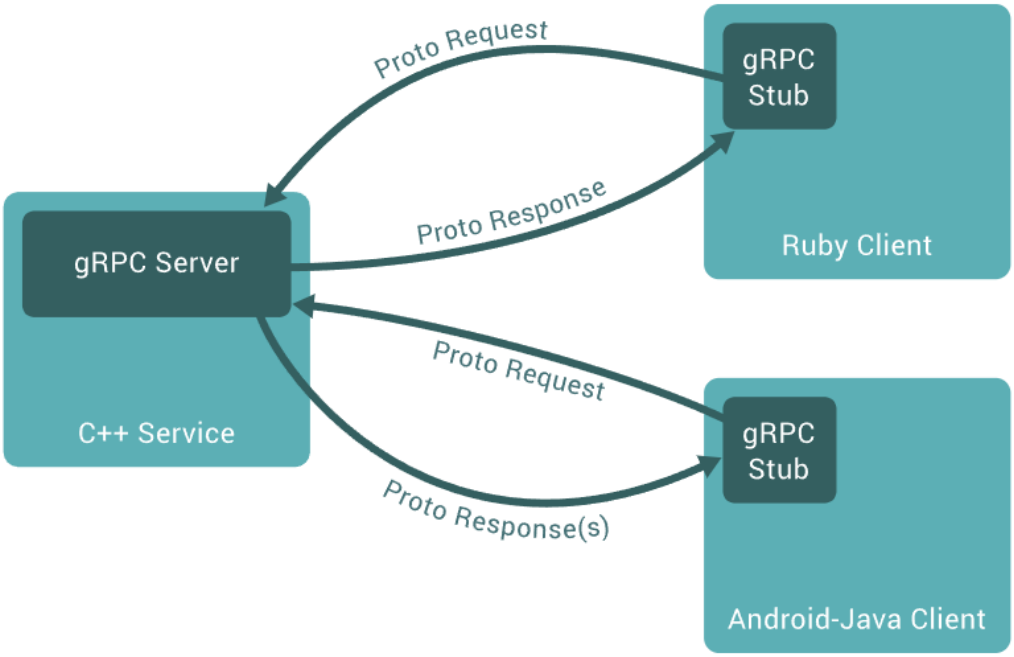

Figura 2.2: Rappresentazione grafica di una iterazione effettuata attraverso il servizio [9].

la trasmissione di pacchetti può portare a complicazioni, è stato necessario ideare delle strategie che le limitassero. La semantica:

* at most once, in questo caso una chiamata può fallire, però è garantito che non venga ripetuta più volte;
* at least once, è l'approccio opposto del precedente, garantisce che la procedura venga chiamata almeno una volta;
* exactly once, coincide con la semantica tradizionale, ovvero una procedura viene chiamata esattamente una volta.

Analisi tecnica

Indipendentemente dall'approccio scelto (rest o rpc), le funzionalità messe a disposizione da google sono le stesse. L'elenco sottostante

mostra i parametri più unici ed interessanti, validi per l'esecuzione della request:

- model
  - in questo caso, google dà la possibilità di scegliere quale modello utilizzare per la traduzione. Nonostante sia molto interessante in termini di personalizzazione e testing, è generalmente più indicato utilizzare i modelli proposti dal servizio, in quanto già ottimizzati per la tipologia di task;
- labels
  - L'API Cloud Translation supporta l'aggiunta di etichette definite dall'utente (coppie chiave-valore) alle richieste TranslateText, BatchTranslateText e DetectLanguage. Le informazioni sull'utilizzo di una richiesta vengono inoltrate al sistema di fatturazione, dove puoi suddividere gli addebiti mediante l'applicazione di un'etichetta.

2.2 OpenMT

OpenNMT è un framework open source per la neural machine translation, è molto flessibile, permette infatti di adoperare tante diverse tecniche per la traduzione, mettendo a disposizioni librerie client e funzionalità sfruttabili da linea di comando. Il progetto, per la costruzione di reti neurali profonde (deep learning), si appoggia a tensorflow [5] e pytorch, offrendo due differenti versioni. Il lato sperimentale di questa tesi è basato principalmente su open-nmt (versione tensorflow), il quale garantisce un alto grado di personalizzazione, oltre ad essere particolarmente efficiente.

2.2.1 Tensorflow

Tensorflow [5] è una piattaforma open source che si occupa di facilitare l'addestramento di reti neurali profonde. L'ecosistema comprende strumenti, librerie e altre risorse, le quali permettono ai ricercatori di creare e pubblicare applicativi basati sul machine learning. Questo permette di aggiornare costantemente lo stato dell'arte della branca di computer science legata all'intelligenza artificiale.

2.2.2 Modello

Il servizio mette a disposizione dei modelli predefiniti, i quali necessitano solo di addestramento e appositi parametri perché siano operativi. Essi vengono inoltre classificati in base alla dimensionalità, ad esempio, nel caso in cui si abbia a disposizione una bassa capacità computazionale per un esperimento, si utilizza il Tiny Trasformer. Allo stesso modo, se si ha a disposizione un sistema distribuito con una grande capacità, è possibile osare ed utilizzare la versione Huge. La scelta è inoltre veicolata dalla dimensione del dataset, il quale deve essere compatibile con la dimensione del modello. Open-nmt è inoltre caratterizzata dal fatto che, è possibile costruire modelli personalizzati. I custom models, si possono implementare attraverso la client library messa a disposizione dal servizio, che verte su python. In particolare, si parte da una sorta di classe astratta, che definisce il modello base, della quale verrà fatto l'override definendo i valori dei parametri prestabiliti. Si ha la possibilità dunque di modellare le seguenti caratteristiche:

- embedding size:
    - rappresenta la dimensionalità del vettore multidimensionale correlato ad una parola, non esiste una regola precisa per determinarlo, si definisce empiricamente, le dimensioni più comuni sono: 100, 300, 512, 618, 712. Attenzione, una classi-

ficazione (piatta) delle parole, richiede un embedding size più piccolo, mentre la predittività richiede una dimensione più grande (come nel nostro caso);

- num layers:
  - esprime il numero di layer intermedi di una rete neurale, più layer ho, maggiore sarà l'espressività. Un numero elevato di layer però, comporterà un largo uso di memoria, di conseguenza si perde in termini di prestazioni. E' dunque necessario trovare un equilibrio.
- ffn dropout
  - parametro di una variabile di bernoulli che determina la probabilità che un elemento venga posto a zero o droppato. E' fondamentale per combattere l'overfitting;
- num of heads:
  - indica il numero di unità di attention messe a disposizione al modello, come già spiegato nel primo capitolo del trattato, più alto è il numero di unità, più sarà precisa la predittività e corretto il contesto della frase;
- ffn inner dim:
  - Se per assurdo avessi una rete neurale infinitamente ampia, avrei a disposizione una distribuzione gaussiana attraverso la quale sarei in grado di mappare ogni funzione desiderata. Di conseguenza, con l'aumentare dell'ampiezza della rete neurale, maggiore è il potenziale approssimativo. Il parametro in questione permette dunque di modulare questo aspetto, ovvero l'ampiezza della rete neurale. La scelta di questo valore, unita al dropout, rappresentano un tentativo di risolvere in maniera ottimale i problemi di identificazione e classificazione.

2.2.3 Parametri all'esecuzione

Meglio noti come run parameters, contenuti in un file YAML (diviso in sezioni), definiscono le specifiche e le informazioni necessarie al modello durante la fase di training. Le specifiche trattate sono le seguenti:

Data

Contiene le informazioni relative all'utilizzo e alla gestione dei dati. Richiede il path relativo dei dataset source e target, oltre al vocabolario, che è generabile da linea di comando. Opzionalmente viene data la possibilità di assegnare un peso per ogni file di training, solo però nel caso in cui ne vengano definiti molteplici. Infine è presente una sezione in cui indicare, attraverso un file, l'applicazione dei token ai dataset (in inglese tokenization). L'etichettatura (in inglese tokenization) è un semplice processo che prende in ingresso dati grezzi e li converte in archivi elaborati. Nonostante questa tecnica si estenda in tantissime diramazioni dell'informatica, è considerata di vitale importanza nell'ambito del neural language processing. Viene utilizzata appunto per eseguire uno split di paragrafi in sotto-paragrafi o frasi, alle quali verrà poi assegnato un contesto.

Params

La seguente label riguarda il fine-tuning, che consiste nella scelta di parametri in grado di ottimizzare il modello selezionato. E' buona prassi utilizzare gli algoritmi di ottimizzazione predefiniti, disponibili all'interno del package Optimizer. Sempre allo stesso fine, viene data la possibilità di selezionare un algoritmo, anch'esso predefinito, il quale è in grado di gestire il weight decay (o decadimento del peso). Si tratta di tecniche che, regolarizzando il peso di determinati parametri, massimizzando la predittività del modello, diminuendone di fatto l'errore.

Infine, per combattere efficacemente l'overfitting viene indicato il dropout, al quale può essere assegnato un punteggio con range da 0 ad 1.

Train

La label citata nel titolo, modella le informazioni riguardanti la fase di training, successiva dunque al setup del modello. Si tratta di un parametro molto delicato, in quanto un training gestito erroneamente può portare a conseguenze disastrose, anche in termini di discriminazione. Dal punto di vista tecnico, è possibile effettuare il training in batch, dunque piuttosto che elaborare un esempio (o token) alla volta, è possibile computarli in una sola soluzione. In tal senso, nel file di configurazione del modello, sotto la label "batch size" si può specificare quanti esempi elaborare in una sola iterazione. Se posto zero, l'IA calcola in automatico il valore ottimale, in funzione dell'hardaware della macchina sulla quale viene effettuato il training.

Eval

Eval, abbreviazione di evaluation, è la sezione del file di configurazione che permette di attivare e definire la valutazione della traduzione appena effettuata. Le metriche valutative sono le più note in ambito traduzioni, è possibile sceglierne una fra:

- bleu;
- rouge;
- wer;
- ter;
- prf.

Questo passaggio è tranquillamente evitabile, una volta che il modello è completo, è possibile utilizzare i tool ufficiali forniti dai servizi di valutazione elencati.

2.2.4 Etichettatura e generazione di vocabolari

Open-nmt si aspetta e genera del testo etichettato, spetta all'utente applicare i token al testo di input, attraverso il tool che preferisce. In caso non fosse noto, l'etichettatura dei dati è l'atto di etichettare o annotare diversi set di dati per insegnare ad una macchina ad identificare differenze e pattern ricorrenti tra loro. Nonostante sia consigliato avere dataset già strutturati ed annotati, attraverso il file di configurazione dei parametri del modello, consente di specificare un tokenizer per effettuare l'etichettatura on the fly (al momento dell'esecuzione).

SentencePiece

Fra i vari algoritmi di applicazione di token, emerge SentencePiece, un progetto di stampo tensorflow. Esso implementa le subword units, sfruttando l'algoritmo BPE e modelli i quali apprendono a partire da un dataset di proposizioni sparse e non strutturate. Si parla dunque di un sistema end-to-end che non dipende da un specifiche di linguaggio, sia nella fase di pre-processing che in quella di post-processing.

2.3 Virtualizzazione - Docker

Il framework visto nella sezione precedente, Open-Nmt ha due caratteristiche alla quale è necessario porre attenzione:

- non è multi-piattaforma, è eseguibile solo su sistema unix, inoltre gli script di esecuzione sono scritti in bash;

– richiede l'utilizzo di GPU con conseguente configurazione dell'ambiente.

Per i motivi elencati, per garantire portabilità e efficienza è indispensabile affidarsi alla virtualizzazione, a questo proposito nella quasi totalità dei casi ci si affida a Docker, il quale obiettivo è di risolvere questo genere di problematiche attraverso una soluzione software.

2.3.1 Docker Overview

Docker [6] è un progetto Open Source pubblicato nel 2014 dall'ideatore Solomon Hykes, progettato per eseguire processi informatici in ambienti isolabili, minimali e facilmente distribuibili chiamati container, con l'obiettivo di semplificare i processi di deployment di applicazioni software. La peculiarità del progetto, motivo del suo successo, consiste nel gestire la OS-level virtualization di linux, abbandonando l'approccio tradizionale. Si tratta di un metodo, per il deployment di una applicazione, che isola le risorse hardware e software in uso, mantenendo però il tutto all'interno di un solo sistema operativo, di kernel linux. L'istanza isolata prende il nome di container e rispetto alla virtualizzazione tradizionale, non sono presenti le macchine virtuali.

Per garantire la separazione delle risorse (memoria, cpu, ecc..), docker implementa della API di alto livello che gestiscono container che eseguono processi in ambienti isolati. Nonostante condividano lo stesso kernel, i container possono essere programmati per utilizzare solo una determinata quantità di risorse, è possibile dunque eseguire un partizionamento. Ogni singola istanza è quindi indipendente rispetto alle altre, dunque è possibile interrompere o creare nuovi task, senza condizionare gli altri container in esecuzione. Per convenzione, è solitamente consigliato assegnare un singolo task per ogni container. Sempre in tema di gestione risorse, docker è estremamente più efficace rispetto alle tradizionali macchine virtuali, in quanto, nonostante i container siano

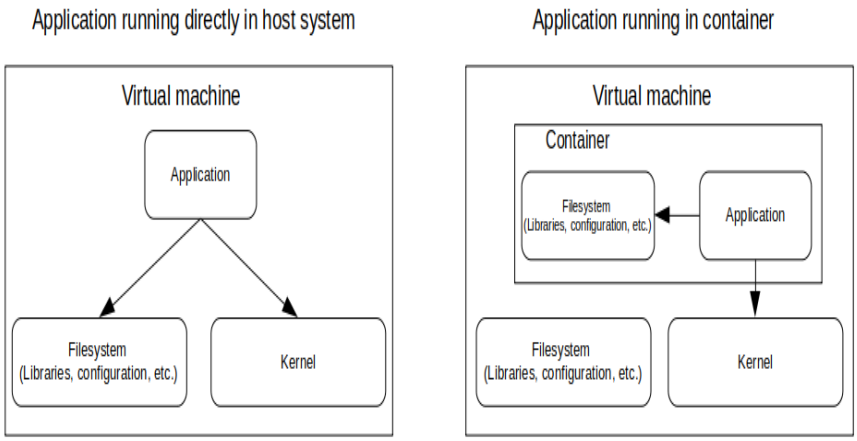

Figura 2.3: Differenza fra virtualizzazione tradizionale e Os-Level-virtualization [10]

separati e abbiano una loro memoria dedicata, non sarà mai presente un file duplicato. Questo perché, una volta aperto il container in lettura, il file visualizzato, sarà sempre lo stesso, mantenuto in memoria condivisa.

2.3.2 Come utilizzare docker

La creazione di una immagine docker, richiede una serie di passaggi, effettuati a partire dalla linea di comando.

1. Dockerfile

Il Dockerfile è un semplice file di testo, contenente le istruzioni utili per la creazione della immagine docker. All'interno del file viene indicato l'id di una immagine software preesistente, la quale verrà poi estesa. In seguito è possibile, attraverso una sintassi prestabilita, in-

cludere istruzioni che permettano di scaricare, installare ed eseguire applicazioni.

2. Docker image

A partire dal docker file, viene generata l'immagine, con il comando:

- docker build -t <myDockerImage> <pathToDockerFile>
  - <myDockerImage> è il nome che si vuole assegnare all'immagine;
  - <pathToDockerFile> è invece il percorso assoluto che porta alla directory in cui è contenuto il file di configurazione (dockerfile).

Un enorme vantaggio da docker è la gestione delle dipendenze, che rendono il container operating's system agnostic. Una volta installati tutti i pacchetti all'interno del container, è possibile pubblicare ed aggiornare l'immagine sul portale dockerhub o in alternativa eseguirla richiamando il comando:

- docker run <imageId>
  - <imageId> è l'id associato alla immagine docker, è possibile ottenerla richiamando il comando "docker images", il quale elenca tutte le immagini create sulla macchina.

Capitolo 3

Risultati

Nel seguente capitolo saranno presentati e contestualizzati i risultati ottenuti dalle sperimentazioni. Nella parte introduttiva verranno invece descritti i criteri di valutazione.

3.1 Punteggio Bleu

BLEU (Bilingual Evaluation Understudy) è una metrica che misura la differenza tra una traduzione automatica ed una effettuata dall'uomo. Il meccanismo funziona perché nella correlazione fra linguaggio macchina e umano, BLEU piuttosto che tentare di imitare il giudizio umano per ogni frase, media gli errori di valutazione delle singole frasi su un corpus di test [7].

Calcolo del punteggio

Il confronto fra traduzioni, avviene attraverso il confronti fra unigrammi, il primo unigramma è ottenuto a partire dal risultato della traduzione effettuata dall'uomo, il secondo da quella risultante dalla macchina. Un unigramma non è altro che un sottoinsieme di parole rispetto alla

totalità di parole presenti nella frase. Ad esempio, data la frase: "il cielo stamani sembra piuttosto nuvoloso", un possibile unigramma è {"cielo", "sembra"}. L'algoritmo tenta di confrontare fra loro tutti i possibili unigrammi, di conseguenza attraverso la combinatoria è possibile definire con esattezza il numero totale di possibili combinazioni. La formula che sta alla base del ragionamento prende il nome di unigram precision ed è composta nel seguente modo:

– Unigram precision $= \frac{numberOfMatches}{totalGeneratedWords}$

La precisione dunque valuta la proporzione di parole che fanno match fra la machine ed human translation (numberOfMatches), sul totale delle parole generate dalla macchina (TotalGeneratedWords). Il risultato è dunque un valore che oscilla fra 0 ed 1, più alto è il punteggio, maggiore sarà l'accuratezza della traduzione.

La formula appena vista può però non offrire sempre un'informazione del tutto corretta ed utile, in quanto non è in grado di identificare i casi in cui sono presenti parole duplicate, come si può osservare in figura 3.2.

La formula più adatta al contesto è dunque la modified unigram, la quale molto semplicemente elimina i duplicati durante la stima del numero di match corrispondenti fra le traduzioni. Il calcolo si presenta così:

– Modified unigram precision $= \frac{clip(numberOfMatches)}{totalGeneratedWords}$

Il calcolo ora è molto più accurato ed può essere un ottimo punto di partenza per la misurazione. In particolare, il calcolo di unigrammi modificati, non considera l'ordine, in quanto si basa su unico layer. A questo preciso scopo, BLEU applica la stessa formula tante volte quante le parole generate dalla macchina, considerando dunque non più solo gli unigrammi, ma gli n-grammi. Il concetto viene chiarito dalla figura 3.3, in cui viene mostrata una iterazione arbitraria.

Sulla base dell'esempio appena analizzato, BLEU applica la formula per ogni possibile dimensione di chunks, ricoprendo un range che parte da un unigramma fino ad arrivare, al massimo a 4-grams. A partire dalle osservazioni proposte si può formalizzare il concetto, presentando la formula generale che copre tutte le casistiche:

– BLEU = $\min(1, \frac{clip(numberOfMatches)}{totalGeneratedWords}) \cdot (\prod_{n=1}^{4} precision_i)^{\frac{1}{4}}$

Il punteggio BLEU è basato principalmente sul calcolo della media geometrica, considerando le precisioni. A questo proposito viene utilizzata una produttoria, per generalizzare il concetto, ad esempio alla quarta iterazione, il calcolo risulterebbe così: $\sqrt{p_1 \cdot p_2 \cdot p_3 \cdot p_4}$.

3.2 OpenNmt

Come già accennato nel secondo capitolo (Tecnologie), per la sperimentazione è stato fatto affidamento su Open-nmt, in questa sezione verrà presentata la logica e la struttura di esso, prima di presentare effettivamente i risultati. Importante ricordare che, il training del modello predefinito, viene effettuato a partire da un file yaml di configurazione, contenente i parametri. In questa sezione verrà nello specifico analizzata la struttura di questo file.

3.2.1 Modello e parametri

Codice python

```
# Model and optimization parameters.
params:
  optimizer: Adamax
  optimizer_params:
  beta_1: 0.8
  beta_2: 0.998
```

```
learning_rate: 1.0
...
decay_type: NoamDecay
decay_params:
model_dim: 512
warmup_steps: 4000
# (optional) The number of training steps that make 1
                              decay step (default: 1).
decay_step_duration: 1
# (optional) After how many steps to start the decay
                             (default: 0).
start_decay_steps: 50000
# (optional) The learning rate minimum value (default
                             : 0).
minimum_learning_rate: 0.0001
...
contrastive_learning: true
```

Gli esperimenti sono stati effettuati basandosi su modelli predefiniti, in particolare Transformer e TinyTransformer, ai quali sono stati passati parametri personalizzati, che sono definiti nel seguente modo:

- Transformer:
  - 6 livelli di encoding;
  - 6 livelli di decoding.
- Transformer Tiny:
  - 2 livelli di encoding;
  - 2 livelli di decoding.

Le linee di codice sottostanti mostrano le principali scelte riguardanti i parametri di ottimizzazione del modello. Un primo e molto importante parametro di ottimizzazione del modello è l'optimizer, il quale riveste un ruolo fondamentale, in quanto può migliorare o peggiorare notevolmente la qualità del risultato. Gli optimizers sono funzioni

che aggiornano il modello, a partire dall'output della loss function (o più semplicemente, funzione di errore). Dunque, una volta ricevuto in input il valore di ritorno della funzione d'errore, l'optimizer corregge l'errore, rimodulando la linea di funzione. Per questo esperimento è stato deciso di optare per un algoritmo predefinito di ottimizzazione, chiamato Adamax, versione aggiornata di Adam presentato nel (inserire autore e data). Successivamente viene indicato l'algoritmo che gestirà il weight decay (spiegato nel secondo capitolo) e la scelta è ricaduta su NoamDecay, definito nell'articolo Attention is all you need. Il minimum learning rate invece serve per definire la quantità minima di informazioni che il modello sarà in grado di apprendere per step. In questo caso si è deciso di optare per il minimo valore consentito, in quanto l'obiettivo è di massimizzare le risorse a disposizione. Infine il contrastive learning consente di apprendere informazioni da un dataset indipendentemente dalla etichettatura, è stato deciso di attivarlo, in quanto può risultare utile nei casi in cui ci siano imperfezioni nella banca dati.

3.2.2 Training

In questa sotto-sezione, verranno illustrate le scelte effettuate circa le modalità di training del modello.

Codice python

```
# Training options.
train:
# (optional) Training batch size. If set to 0, the
                                training will search the
                                largest
# possible batch size.
batch_size: 0
```

```
# (optional) Tune gradient accumulation to train with
                              at least this effective
                              batch size
# (default: null).
effective_batch_size: 25000
# (optional) Save a checkpoint every this many steps (
                              default: 5000).
save_checkpoints_steps: 2000
# (optional) How many checkpoints to keep on disk.
keep_checkpoint_max: 10
# (optional) Dump summaries and logs every this many
                              steps (default: 100).
save_summary_steps: 200
# (optional) Maximum training step. If not set, train
                              forever.
max_step: 1000000
# (optional) The maximum length of feature sequences
                              during training (default:
                              null).
maximum_features_length: 90
# (optional) The maximum length of label sequences
                              during training (default:
                              null).
maximum_labels_length: 90

# (optional)
moving_average_decay: 0.9999
# (optional)
average_last_checkpoints: 8
```

I modelli NMT, sono molto più efficienti rispetto ai predecessori durante il training, in quanto le loro reti neurali ammettono l'addestramento in batch (o lotti). In particolare, rispetto ad un testo da tradurre, piuttosto che elaborare word by word per step, viene elaborata l'intera frase. Oltre ad ottimizzare le prestazioni, facilita la comprensione ed estrazione del contesto. Per quanto riguarda i singoli parametri, è inte-

ressante osservare che per la batch size (ovvero il numero di esempi per lotto), è stato inserito un valore pari a 0. Questo significa che, l'intelligenza artificiale determina autonomamente il batch size, ritornando il massimo valore possibile.
Successivamente viene data la possibilità di scegliere come gestire i checkpoint, ovvero ogni quanti step effettuare un backup. I dati del backup vengono inseriti in una directory creata in automatico durante il training, riutilizzabili per riprendere l'addestramento a partire dall'ultimo checkpoint. Infine, è stata indicato il moving average decay, parametro che definisce con quale frequenza effettuare la media dei valori ritornati dai parametri addestrati. Le medie vengono utilizzate negli step successivi di addestramento, come punto di riferimento, per l'ottimizzazione dei calcoli. La sua omissione può compromettere significativamente i risultati, a questo proposito è buona pratica inserire un valore che si avvicini all'1.

3.2.3 Inference e Score

L'ultima porzione del file è dedicata all'estrapolazione dei punteggi e risultati dell'esperimento, vengono differenziati:

- Inference, consiste nel calcolo di un singolo output numerico, da parte di un modello di machine learning. Il meccanismo viene generalmente utilizzato per fare label prediction, oppure, una implementazione più sofisticata può essere il fulcro di un algoritmo di reccomendation. Si tratta dunque di una stima.
- Score, a differenza della inference, consiste nel calcolo di un valore esatto, determinato da una formula, che varia in funzione del tipo di criterio di valutazione selezionato. Serve per valutare la qualità di traduzione del modello creato.

Codice python

```
infer:
  # (optional)
  batch_size: 10
  # (optional)
  batch_type: examples
  # (optional)
  n_best: 1
  # (optional) For compatible models, also output the
                              score (default: false).
  with_scores: true
  # (optional) For compatible models, also output the
                              alignments
  # (can be: null, hard, soft, default: null).
  with_alignments: soft
  # (optional)
  length_bucket_width: 5

# (optional) Scoring options.
score:
  # (optional)
  batch_size: 64
  # (optional)
  batch_type: examples
  # (optional)
  length_bucket_width: 0
  # (optional)
  with_token_level: false
  # (optional)
  with_alignments: null
```

Come si può osservare dalla porzione di codice presentata, è possibile eseguire il setup batch sia per lo score che per l'inference. Il calcolo dell'output desiderato richiede, per ogni step, un insieme di esempi, indicato dal batch size (come succede nel training).

Il parametro length bucket width indica l'ampiezza dei bucket dalla quale estrarre gli esempi. Nel caso in cui venga selezionato, i dati di test verranno ordinati in funzione della loro lunghezza, allo scopo di migliorare l'efficienza. I parametri non citati permettono di selezionare quali informazioni mostrare nello standard output.

3.3 Dataset

La scelta del dataset, utilizzato per la fase di apprendimento del modello, è ricaduta su un estratto di TED-CDB. Si tratta di un dataset creato sulla base dei ted talks, ovvero conferenze pubbliche gestite da un'organizzazione privata non-profit statunitense nominata Sapling Foundation. I realizzatori del progetto sono dei ricercatori dell'università di Edinburgo, i quali lo hanno presentato nell'articolo "TED-CDB: A Large-Scale Chinese Discourse Relation Dataset on TED Talks". TED-CDB comporta una rivoluzione nel mondo delle traduzioni, dal momento che in precedenza, le trascrizioni in cinese venivano recuperate unicamente a partire dalle news [19]. La trascrizione è avvenuta secondo lo standard definito dalla Chinese Discourse Treebank (CDT). CDT è stato sviluppato come parte integrante del progetto Treebank e consiste in circa 70.000 parole di cinese annotate a partire da TED talks. Segue l'approccio del Penn Discourse Treebank (PDTB), con un adattamento basato sulle caratteristiche del cinese scritto. I ricercatori hanno poi effettuato delle sperimentazioni, addestrando dei modelli di NMT, sia con il dataset di TED-CDB, che con le trascrizioni prese a partire dalle news. Dai risultati ottenuti, si è potuto evincere che, il modello apprende molto più velocemente e correttamente a partire dal dataset TED-CDB, ciò ha permesso di definire un nuovo stato dell'arte in questo ambito.

3.3.1 L'annotazione a 4 vie

Nell'articolo citato in questa sezione, è stato utilizzato un sistema di annotazione, nominato "4 way annotation". Si tratta dunque di un confronto a quattro vie e viene regolarmente utilizzato dal Pennybank treebank (PDTB). Il concetto si basa sul fondamento che, uno speech, viene inizialmente suddiviso attraverso un'astrazione in singole argomentazioni. Lo scopo dell'annotatore è di collegare argomenti affini tra loro, in modo da comporre un unico discorso coeso.

I 4 parametri di valutazione sono i seguenti e seguono una struttura gerarchica, considerando come riferimento due argomenti, arg1 ed arg2:

- Temporal:
  - Nel PDTB originariamente, potevano essere collegate argomentazioni, solo se sequenziali a livello temporale. In questo caso invece, è stata data la possibilità di collegare logicamente anche frasi o argomentazioni enunciate in diversi istanti temporali. A questo proposito, la congiunzione può essere:
    - synchronous;
    - asynchronous.
- Contingency:
  - classifica la causalità del discorso, mettendo su due diversi piani le argomentazioni. In particolare, è possibile osservare un rapporto causa-effetto, in cui ad esempio nella prima parte del discorso (arg1) viene enunciata la causa, mentre nella seconda (arg2) l'effetto. Tutte le possibili combinazioni sono presenti nella figura 3.4.
- Comparison:
  - confronta le due argomentazioni, verificando se possono essere considerate simili o contrastanti.

- Expansion:
  - mostra nel dettaglio tutte le possibili caratteristiche del discorso, introducendo, come si può evincere dall'immagine a pagina successiva, numerosi parametri al secondo e terzo livello di gerarchia.

3.4 Presentazione risultati e discussione

In questa ultima sezione, verranno mostrati, attraverso delle tabelle, i punteggi ottenuti a partire dalla sperimentazione, parametrati a BLEU. La tabella 3.1 descrive le tecnologie hostate su github, mostrando il loro punteggio in termini di github stars. Questa valutazione può tornare utile nella scelta del servizio, perché generalmente, maggiore è la quantità di start, più affidabile è il servizio, in quanto va a dimostrazione del fatto che, è stato testato da un grande numero di utenti. Nella tabella non è presente google translate, dal momento che la repository del progetto non è disponibile pubblicamente su github. Nonostante ciò, si può osservare che l'utenza di Google raggiunge cifre superiori al miliardo, ponendolo tra i traduttori più utilizzati di sempre[14].Infine la tabella 3.3 mostra, a partire da range realizzati sulla base del punteggio bleu, una interpretazione circa la qualità della traduzione.

Nome	**Linguaggio**	**Technology**	**Github stars**
Open-Nmt	python	tensorflow	1.3k
ModernNmt	python/java	pytorch	0.3k
DeepL	python	tensorflow	0.5k
Azure	python, .NET	tensorflow	3.2k

Tabella 3.1: Elenco delle tecnologie open source utilizzate durante la realizzazione della tesi. Se si tratta di un progetto open source, il valore indicato in "Valutazione" è definito in funzione del numero di github stars, un criterio di valutazione gestito appunto dalla piattaforma github, indica il numero di utenti che hanno apprezzato il progetto.

Blue score	Interpretazione
< 10	Quasi inutile
10 - 19	Difficile comprendere il contesto della frase
20 - 29	Il contesto è chiaro ma presenta numerosi errori grammaticali
30 - 39	Buona traduzione, comprensibile
40 - 49	Traduzione di qualità
50 - 59	Traduzione di alta qualità e fluente
>= 60	Qualità maggiore rispetto alla traduzione effettuata da esseri umani

Tabella 3.2: Legenda che determina, in funzione del punteggio BLEU ottenuto, il livello qualitativo della traduzione. Da come si può osservare, i risultati accettabili partono da un punteggio minimo di 30, fino ad arrivare all'ottimo che si aggira intorno ai 60.

3.4.1 Procedimento

Gli esperimenti, come già anticipato, sono stati realizzati utilizzando un estratto di TED-CDB. I punteggi sono stati valutati nelle casistiche:

- it → zh (italiano → cinese semplificato);
- zh → it (cinese semplificato → italiano).

Il vocabolario invece, è stato generato, in tutti gli esperimenti, a partire dagli strumenti messi a servizio dalla tecnologia scelta. Open-Nmt ad esempio, permette da terminale di generare un vocabolario, mettendo a disposizione solo il dataset e la dimensione finale del vocabolario desiderata. Questo perché a partire da esso, effettua una tokenization on fly, sulla base della dimensione indicata in input. Si è scelto di creare un vocabolario di dimensioni piuttosto ridotte, ovvero di 5000 parole,

per facilitare l'addestramento e il testing del modello. Per valutare quale potesse essere l'impattto del volume del dataset, è stata presa la decisione di addestrare sia un modello di tipo Transformer che TinyTransformer. Il secondo è meno voluminoso ed elaborato del primo, la semplificazione riduce inevitabilmente i tempi di computazione, rendendolo però meno potente rispetto all'altro. La versione tiny viene integrata spesso nelle sperimentazioni perché se si ottengono gli stessi risultati ottenuti con il Transformer tradizionale, si può osservare che:

- probabilmente le dimensioni del dataset e di conseguenza, del vocabolario, sono piuttosto ridotte;
- per avere un miglioramento si ritiene necessario avere un dataset con un maggior numero di esempi, in determinati casi viene effettuato un merge. Si può ad esempio creare un unico dataset, unendo TED-CDB e TED-new-CDB (banca dati creata sulla base delle news).

3.4.2 Discussione

Gli esperimenti hanno restituito, purtroppo, punteggi bleu molto deludenti, rientrando tutti nella categoria interpretativa "almost useless". Nonostante ciò, può risultare utile analizzare e studiare i motivi che hanno portato al fallimento, in modo da facilitare la stesura di articoli futuri.

Problemi traduzione italiano cinese

La traduzione da una lingua latina al cinese è ostica, l'elenco sottostante, il quale riprende i concetti di un articolo elaborato all'università del Montana, permette di capirne il motivo:

- Non linearità:

    - è noto che, il cinese scritto può seguire diverse direzioni:
        - alto → basso;
        - sinistra → destra;
        - destra → sinistra,

      Questo complica la traduzione, non godendo della linearità tipica delle lingue tradizionali, che vengono scritte ed intepretate da sinistra verso destra.

- Pinyin:
    - si tratta di una tecnica che consente di traslare la scrittura cinese tradizionale, ovvero scrivendo in cinese utilizzando lettere dell'alfabeto arabo. Un sistema di traduzione spesso necessita di questa tecnica, la quale però ha un problema non indifferente. Il metodo non è in grado di trasporre determinate espressioni da cinese ad una lingua latina, per problemi legati alla pronuncia e complessità.
- Grammatica:
    - il cinese non ammette forme in singolare o plurale, oltre all'assenza di coniugazioni che permettono di interpretare la proposizione. Per capirne il senso bisogna intuire il contesto, che è il compito più difficile e costoso in termini computazionali per un NMT.
- Idiomi:
    - l'espressività del cinese differisce dall'italiano, in quanto usa frequentemente forme indirette. Il problema si pone principalmente nelle descrizioni, in cui è molto difficile capirne il significato.
- Caratteri
    - i caratteri cinesi sono più di 50.000, nonostante ciò, i 500 caratteri più usati coprono l'80 per cento delle conversazioni

giornaliere. Per questo motivo spesso si sceglie di prendere come riferimento il cinese semplificato e non quello tradizionale.

Divario tecnico

Dal punto di vista tecnico, è necessario riconoscere che, la dimensione e qualità dei dataset presi in considerazione è di vitale importanza. Le sperimentazioni che hanno luogo in aziende molto importanti, hanno a disposizione molto spesso banche dati con centinaia di milioni di informazioni. Sempre le stesse aziende, per fare fronte ad una grande mole dati, possiedono strumenti in grado di manipolarli, hanno dunque a disposizione una grande capacità computazionale. A fronte di queste considerazioni, si è in grado di intuire uno dei principali motivi per il quale gli esperimenti condotti, attraverso il servizio Open-Nmt durante la stesura della tesi, non siano andati a buon fine. L'estratto del dataset TED-CDB utilizzato, non andava oltre alle 30.000 parole, motivo per il quale il training non ha portato i risultati sperati. Una dimostrazione della insufficiente dimensionalità del dataset è data dal fatto che, come si può osservare nelle tabelle 3.4 e 3.5, il training ha dato gli stessi risultati sia nel caso in cui si sia ricorso ad un Transformer classico, che alla versione Tiny. A fronte della grandezza delle multinazionali coinvolte, le quali conducono anche ricerche private, è difficile competere dal punto di vista quantitativo.

La tabella sottostante (3.6), mostra i punteggi ottenuti a partire da traduzioni effettuate con le più note tecnologie cloud, le quali sono servite per effettuare un confronto diretto con il modello creato con Open-Nmt.

Dataset (training + test)	Direzione	Blue Score
TED-CDB	it → ch	< 1
TED-CDB	ch → it	< 1

Tabella 3.3: Risultati dell'addestramento sulla base del Transformer

Dataset (training + test)	Direzione	Blue Score
TED-CDB	it → ch	< 1
TED-CDB	ch → it	< 1

Tabella 3.4: Risultati dell'addestramento sulla base del Tiny Trasformer

Considerazioni

I risultati non sono di certo quelli sperati, nonostante ciò credo sia possibile trarre vantaggio dalle deduzioni ed esperimenti realizzati. In particolare, dal momento che il problema si ipotizza che sia prevalentemente quantitativo (dimensione dei dataset), può ritenersi ancora potenzialmente valido il setup del modello. In particolare, il file di configurazione, responsabile del fine-tuning del modello, può risultare utile ed è riutilizzabile nel caso in cui si abbiano a disposizione più risorse. Uno dei vantaggi derivanti dagli esperimenti effettuati con Open-ntm, è la quantità di informazioni generate e memorizzate in locale in dei file di log. A tal riguardo, una volta catalogati ed organizzati i dati già memorizzati in locale, inserendoli all'interno di un database, si apre alla possibilità dell'utilizzo delle tecniche di estrazione. In questo modo, avendo catalogati i dati degli esperimenti nel tempo, si può effettuare una analisi completa, graficando poi i risultati e ottenendo poi una curva che mostri l'andamento degli esperimenti, con cadenza mensile o annuale. I risultati dei log, attraverso un sink fornito da .NET (fruibile anche in python), chiamato serilog, possono essere

Piattaforma		**Azure**		**ModernMT**		**DeepL**		**Google**	
Target		**ita**	**chi**	**ita**	**chi**	**ita**	**chi**	**ita**	**chi**
BLEU		11.58	3.77	12.31	2.72	13.58	2.06	14.67	4.22
1-gram	**Individuale**	35.68	20.12	35.84	22.09	35.89	18.14	38.90	22.84
	Cumulativo	35.68	20.12	35.84	22.09	35.89	18.14	38.90	22.84
2-gram	**Individuale**	13.17	3.18	15.15	3.04	16.28	4.33	17.14	4.38
	Cumulativo	22.17	9.22	23.30	8.20	24.17	8.86	25.82	10.00
3-gram	**Individuale**	6.19	3.14	8.62	0.85	9.80	0.41	10.28	2.19
	Cumulativo	12.33	7.24	16.72	3.85	17.80	3.19	18.99	6.03
4-gram	**Individuale**	3.21	1.22	4.91	0.96	5.93	0.56	6.75	1.45
	Cumulativo	10.11	3.78	12.31	2.72	13.58	2.56	14.67	4.22

Tabella 3.5: Risultati di esperimenti ottenuti a partire dalle più famose tecnologie di traduzione cloud.

archiviati e gestiti in cloud, attraverso Log Analytics, un servizio messo in piedi da microsoft. I log vengono memorizzati in un database non relazionale, che è possibile interrogare attraverso KQL, ovvero kusto query language [20].

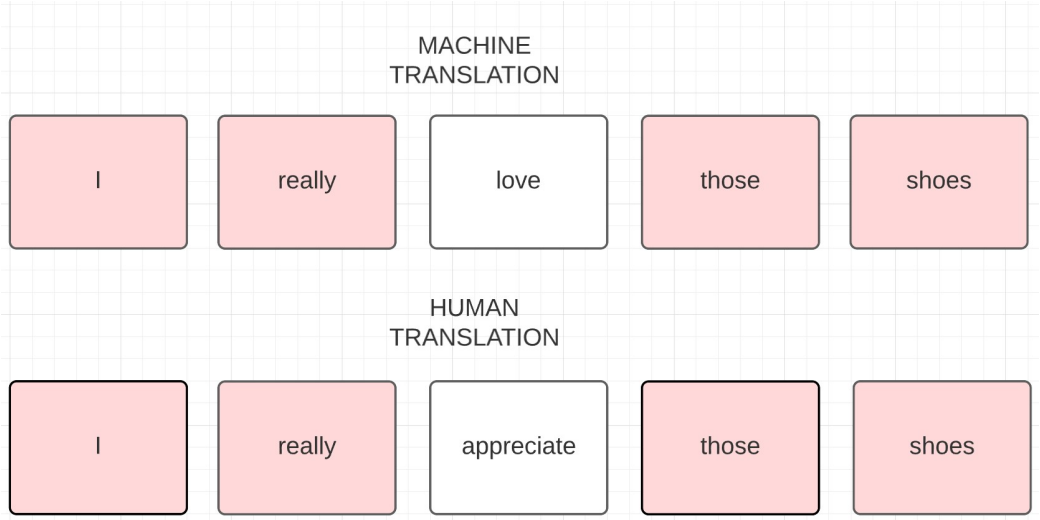

Figura 3.1: Rappresentazione grafica di un possibile esempio di confronto, in questo caso la precisione assume il valore: $\frac{numberOfMatches}{totalGeneratedWords} = \frac{4}{5}$. Il numberOfMatches è 4, perché dal confronto risulta che le parole "I", "Really", "those" e "shoes" corrispondano sia nella traduzione umana che della macchina. Il denominatore totalGeneratedWords è 5, perché l'ia durante la traduzione ha generato una frase composta da 5 parole, ovvero "I", "really", "love", "those", "shoes".

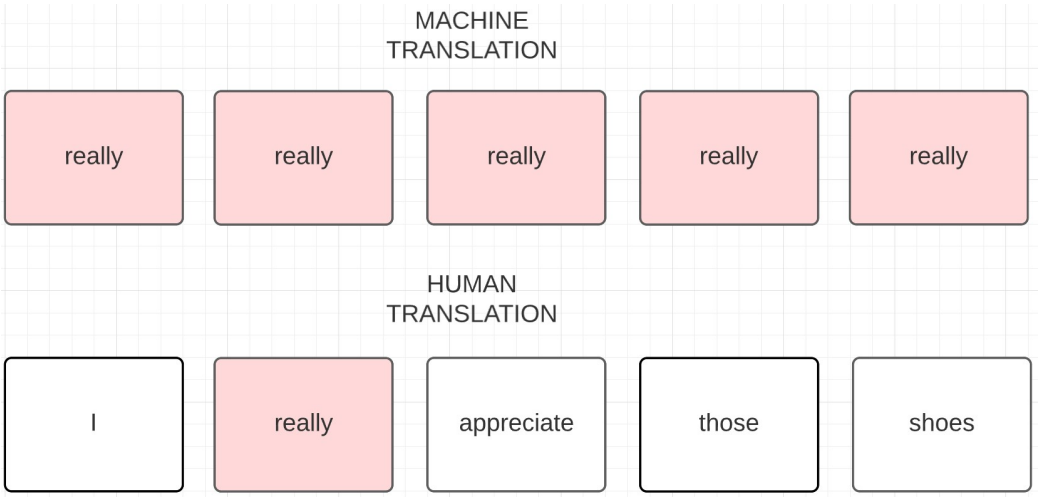

Figura 3.2: In questa circostanza, nonostante sia evidente l'errore di traduzione della macchina, la unigram precision restituisce il seguente risultato: $\frac{numberOfMatches}{totalGeneratedWords} = \frac{5}{5}$. La precisione restituita dalla formula è 1, che indica una traduzione esatta, dunque si tratta di un errore di valutazione, che è risolvibile applicando una modifica.

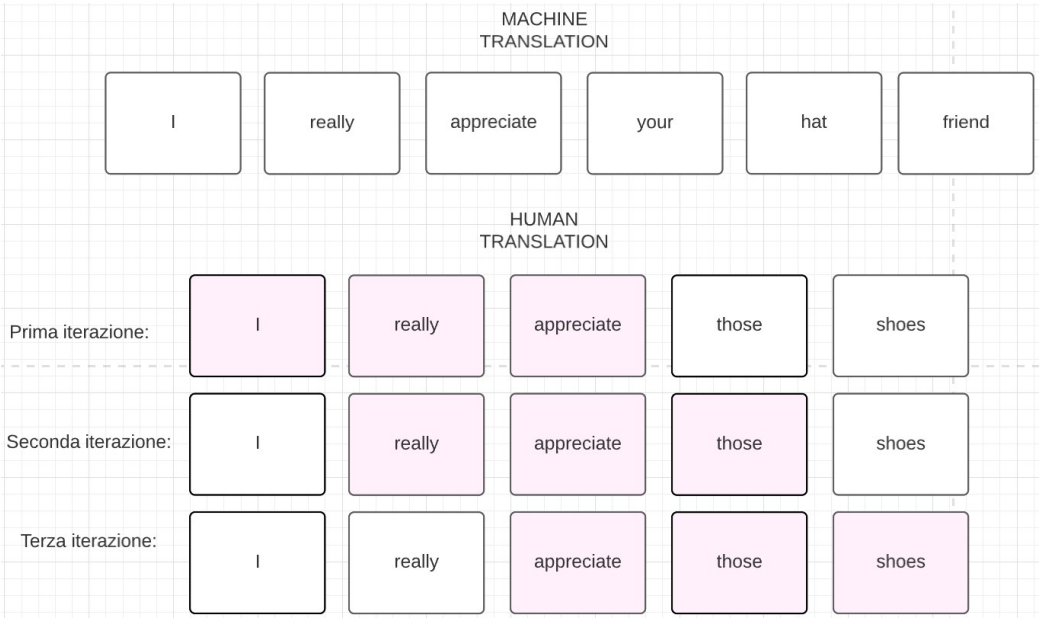

Figura 3.3: In questo esempio viene mostrato il calcolo della precisione dei 3-grams, ovvero il parametro di confronto è composto da chunks (porzioni) di frase, composti da 3 parole. In ogni iterazione viene, in ordine, confrontato un chunk con la traduzione effettuata dalla macchina. In questo caso, il match si concretizza solo durante la prima iterazione, dal momento che "I", "really", "appreciate" compare in ambo le versioni. Il risultato è dunque $\frac{1}{6}$

Temporal	Synchronous	–
	Asynchronous	Precedence
		Succession

Contingency	Cause	Reason
		Result
		Negative-result
	Condition	Arg1-as-cond
		Arg2-as-cond
	Negative condition	Arg1-as-negcond
		Arg2-as-negcond
	Purpose	Arg1-as-goal
		Arg2-as-goal
		Arg2-as-negGoal
Comparison	Contrast	–
	Similarity	–
	Concession	Arg1-as-denier
		Arg2-as-denier

Expansion	Conjunction	–
	Disjunction	–
	Equivalence	–
	Instantiation	Arg1-as-instance
		Arg2-as-instance
	Level-of-detail	Arg1-as-detail
		Arg2-as-detail
	Substitution	Arg1-as-subst
		Arg2-as-subst
	Execption	Arg1-as-excpt
		Arg2-as-excpt
	Manner	Arg1-as-manner
		Arg2-as-manner

Figura 3.4: Rappresentazione dei 4 parametri di valutazione, i quali come si evince dalle tabelle, si espandono fino a 3 livelli di profondità[11]

.

Conclusioni

Il modello di traduzione automatica italiano-cinese, elaborato all'interno del lavoro di tesi, è funzionante, anche se probabilmente non ancora utilizzabili in determinati contesti, in quanto non del tutto accurato. Le ragioni che non hanno permesso di perfezionarlo comprendono una scarsa quantità di dati e capacità computazionale, fondamentali perché si possano ottenere punteggi rispettabili. Tuttavia ci si può ritenere in parte soddisfatti, in quanto il modello ed i parametri utilizzati, possono essere un punto di partenza per approfondimenti futuri. Partire dal presupposto di conoscere quali siano i limiti dimensionali dei modelli e le tecnologie sfruttate, è fondamentale per effettuare miglioramenti ed eventualmente costruire basi solide per future pubblicazioni. Il lavoro può risultare utile in funzione di guida, seguendone i passaggi indicati infatti, è possibile orientarsi con più chiarezza, oltre ad imparare a conoscere con più facilità il dominio di riferimento. Uno spunto interessante presentato nel lavoro, è la possibilità di sfruttare un sistema di gestione di logging, il quale sia in grado di memorizzare e catalogare in un database in cloud, tutti i messaggi di output restituiti durante l'addestramento. A partire da esso infatti, si apre alla possibilità di descrivere l'andamento del fenomeno del tempo, attraverso una distribuzione statistica. Per rimanere sempre in tema di possibili sviluppi futuri, osservando le più recenti pubblicazioni a cura della comunità scientifica si può

approfondire i seguenti argomenti, al fine di raggiungere progressi in materia:

* metodi per la costruzione automatica di un corpus di traduzione italiano-cinese [21];
* utilizzo del metodo divide-et-impera per massimizzare le prestazioni, in particolare nella traduzione di frasi di grandi dimensioni [22];
* introduzione di un sistema di autocalibrazione per gestione del fallimento nella traduzione inglese-cinese basato su big data, il quale può ridurre notevolmente il consumo energetico [23];

Bibliografia

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[2] Lise Volkart, Pierrette Bouillon, and Sabrina Girletti. Statistical vs. neural machine translation: A comparison of mth and deepl at swiss post's language service, 2018. URL `https://archive-ouverte.unige.ch/unige:111777`.

[3] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[4] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation, July 2017. URL `https://www.aclweb.org/anthology/P17-4012`.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sher-

ry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[6] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014 (239):2, 2014.

[7] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://aclanthology.org/P02-1040`.

[8] Saul Dobilas. Lstm recurrent neural networks — how to teach a network to remember the past, 2021. URL `https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-th`

[9] Google. Introduction to grpc, 2015. URL `https://grpc.io/docs/what-is-grpc/introduction/;`.

[10] Differenza fra virtualizzazione e os-level virtualization. URL `Natlibfi-arlehiko`.

[11] Yuping Zhou, Jill Lu, Jennifer Zhang, and Nianwen Xue. Chinese discourse treebank 0.5. *LDC2014T21*, 2014.

[12] DeepL SE. Deepl, 2017. URL `https://www.deepl.com/it/publisher/`.

[13] ModernMT. A more human translation, 2010. URL `https://www.modernmt.com/`.

[14] Google. Google translator.

[15] Microsoft translator.

[16] RedHat. What is a rest api? URL `https://www.redhat.com/en/topics/api/what-is-a-rest-api`.

[17] Christine Heiss and Marcello Soffritti. Deepl traduttore e didattica della traduzione dall'italiano in tedesco, 2018. URL `https://cris.unibo.it/bitstream/11585/659221/1/Heiss%20Soffritti%20DeepL.pdf;`.

[18] Milam Aiken and Shilpa Balan. An analysis of google translate accuracy, 2011.

[19] Wanqiu Long, Bonnie Webber, and Deyi Xiong. TED-CDB: A large-scale Chinese discourse relation dataset on TED talks, November 2020. URL `https://aclanthology.org/2020.emnlp-main.223`.

[20] Marshall Copeland. Kusto query language and threat hunting. In *Cloud Defense Strategies with Azure Sentinel*, pages 185–211. Springer, 2021.

[21] Yuming Zhai, Lufei Liu, Xinyi Zhong, Gbariel Illouz, and Anne Vilnat. Building an english-chinese parallel corpus annotated with sub-sentential translation techniques, 2020.

[22] Yao Huang and Yi Xin. Deep learning-based english-chinese translation research, 2022.

[23] Zonghui He. Self-calibration system for pragmatic failure in english-chinese translation based on big data. *International Journal of Applied Systemic Studies*, 9(2):141–158, 2020.

[24] Google. Bigquery ml model inference overview, 2021. URL https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-inference-overview#:~:text=Machine%20learning%20inference%20is%20the,machine%20learning%20model%20into%20production.%E2%80%9D;.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-libra pdf.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[27] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[28] Shihua Chen Brazill. Chinese to english translation: Identifying problems and providing solutions, 2016.

Ringraziamenti

Mi sento di ringraziare il prof. Giovanni Delnevo per l'attenzione e la cura nell'assistermi durante la stesura del trattato e la prof. ssa Silvia Mirri per avermi proposto questo interessante topic.