ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

STUDIO E SPERIMENTAZIONE DI METRICHE AUTOMATICHE PER LA VALUTAZIONE DI MODELLI GENERATIVI DI LINGUAGGIO NATURALE

Elaborato in
Programmazione

Relatore
Prof. Antonella Carbonaro

Co-relatori
Dott. Giacomo Frisoni

Presentata da
Marco Avagnano

Terza Sessione di Laurea
Anno Accademico 2020 – 2021

PAROLE CHIAVE

Natural Language Processing

Natural Language Generation

Evaluation Metrics

Pre-trained Models

Graph-to-Text

Le nostre vite cominciano a finire il giorno in cui stiamo zitti di fronte alle cose che contano.

Martin Luther King

Sommario

Negli ultimi anni, il natural language processing ha subito una forte evoluzione, principalmente dettata dai paralleli avanzamenti nell'area del deep learning. Con dimensioni architetturali in crescita esponenziale e corpora di addestramento sempre più comprensivi, i modelli neurali sono attualmente in grado di generare testo in maniera indistinguibile da quello umano. Tuttavia, a predizioni accurate su task complessi, si contrappongono metriche frequentemente arretrate, non capaci di cogliere le sfumature semantiche o le dimensioni di valutazione richieste. Tale divario motiva ancora oggi l'adozione di una valutazione umana come metodologia standard, ma la natura pervasiva del testo sul Web rende evidente il bisogno di sistemi automatici,scalabili, ed efficienti sia sul piano dei tempi che dei costi. In questa tesi propongo un'analisi delle principali metriche allo stato dell'arte per la valutazione di modelli pre-addestrati, partendo da quelle più popolari come Rouge fino ad arrivare a quelle che a loro volta sfruttano modelli per valutare il testo. Inoltre, introduco una nuova libreria – denominata Blanche – finalizzata a raccogliere in un unico ambiente le implementazioni dei principali contributi oggi disponibili, agevolando il loro utilizzo da parte di sviluppatori e ricercatori. Infine, applico Blanche per una valutazione ad ampio spettro dei risultati generativi ottenuti all'interno di un reale caso di studio, incentrato sulla verbalizzazione di eventi biomedici espressi nella letteratura scientifica. Una particolare attenzione è rivolta alla gestione dell'astrattività, un aspetto sempre più cruciale e sfidante sul piano valutativo.

Introduzione

Il natural language generation (NLG) è una branca del natural language processing (NLP), si occupa di costruire sistemi in grado di produrre testo comprensibile dall'essere umano Nel tempo sono nate svariate applicazioni concrete basate su di esso, la più famosa è sicuramente la traduzione automatica di testi, ma con l'evoluzione delle tecniche e metodologie di modellazione del linguaggio oggi troviamo task come *summarization*, *question-answering*, *image captioning*, *data-to-text* e molti altri. La nascita delle prime reti neurali, ma soprattutto dei modelli pre-addestrati basati su architettura Transformer [1] ha segnato un importante traguardo nel mondo NLP. Questi modelli infatti sono in grado di contestualizzare il testo, e associare parole dello stesso campo semantico, attraverso i word embedding. Per spiegare questo fenomeno esiste un esempio molto famoso sulla parola "bank". Essa infatti in inglese può avere significati diversi, "bank account" o "bank river", nella prima chiaramente stiamo parlando di un conto bancario mentre nella seconda sponda di un fiume. I modelli pre-addestrati più famosi sono *BERT* [2], *GPT-2/3* [3], *T5* [4] e *BART* [5]. La valutazione dell'output di questi modelli è impegnativa principalmente perché un sistema può generare più risposte plausibili per lo stesso input dell'utente. Prendiamo per esempio il task della summarization: un documento può essere riassunto in diversi modi, ma tutti con lo stesso significato. Pertanto, la valutazione umana rimane quella standard per quasi tutti i task NLG. Tuttavia, a causa del costo in termini economici e di tempo, i ricercatori ricorrono spesso a metriche automatiche che hanno costo zero e soprattutto risultati ripetibili. Le prime metriche adottate per la valutazione di testo, soprattutto per traduzioni, sono quelle basate su sovrapposizione di n-grammi: Bleu [6], Rouge [7], Meteor [8], molto datate e poco correlate con il giudizio umano. BERTscore [9] invece più recente è stata creata per sopperire questo limite, sfrutta i word embedding delle parole per catturarne la similarità semantica. L'avvento dei modelli pre-addestrati ha permesso alle metriche di sfruttare approcci di valutazione basati sul machine learning. Tra questi i più utilizzati sono classificazione, regressione e generazione di testo. Per la classificazione troviamo FactCC [10] che sfruttando BERT effettua una valutazione della consistenza fattuale di una frase rispetto al suo documento sorgente, nel 2019 ha raggiunto risultati

allo stato dell'arte nel task di summarization. Bleurt [11] invece, metrica proposta dai ricercatori Google, sfrutta la regressione per predire il punteggio umano. La chiave di questa metrica é secondo gli autori un pre-train su dati sintetici estratti da documenti di Wikipedia e perturbati, ciò fornisce a Bleurt una robustezza nella speranza di riuscire a valutare correttamente anche testi generati da modelli futuri. Formulare la valutazione come un problema di generazione del testo è un approccio insolito che mira a dare un giudizio dell'output in base alla probabilità di essere generato. É sfruttato da metriche come BARTscore [12] e PRISM [13], in questa tesi però analizzeremo solamente la prima. Molto recentemente, visto il bisogno di valutare aspetti come la fattualità e fedeltà di un testo generato, sono nate metriche basate su question answering principalmente adottate per il task di summarization, che estrapolano domande e risposte sia dal documento sorgente che dal riassunto e le sfruttano per capire se quest'ultimo contenga allucinazioni. Tra queste citiamo le più interessanti: QuestEval [14] e FEQA [15]. Una volta raccolte queste metriche in una libreria, ho effettuato degli esperimenti su del testo verbalizzato da grafi evento biomedici, utilizzando l'algoritmo di beam search. Il primo consiste nell'eseguire le metriche su ogni esempio del test set e riportare i punteggi, nel secondo invece ho filtrato i punteggi per numero di nodi utilizzati per generare l'output. Per effettuare il task di verbalizzazione sono stati usati due modelli: BART e T5. Gli obiettivi sono molteplici: (i) verificare se attraverso il beam search si ottengano risultati qualitativamente migliori rispetto al grid search; (ii) comprendere come varia la qualità del testo generato in base al numero di nodi del grafo impiegati per la generazione; (iii) contribuire alla valutazione sperimentale di un lavoro scientifico; (iv) analizzare i diversi aspetti di valutazione delle metriche impiegate, i limiti, e se effettivamente ci sia bisogno di nuovi approcci, visto l'utilizzo frequente ancora oggi di Bleu, Rouge e Meteor come approccio automatico standard di valutazione.

Nel Capitolo 1 descrivo il mondo del natural language processing, le tecniche che hanno portato alla sua evoluzione, i modelli e le architetture utilizzate oggi per compiere task NLP ed effettuare valutazioni del testo generato.

Il Capitolo 2 analizza il funzionamento delle principali metriche allo stato dell'arte oggi disponibili.

Nel Capitolo 3 descrivo la libreria "Blanche" che ho creato, e che racchiude un po tutte le metriche analizzate nel capitolo precedente, che ho sfruttato per l'esecuzione degli esperimenti.

Il Capitolo 4 riporta la descrizione dettagliata di tutti gli esperimenti effettuati, analizzando lo specifico task, i modelli utilizzati e le considerazioni sui risultati ottenuti.

Indice

Elenco delle figure

Capitolo 1

Framework teorico

In questo capitolo verrà descritto il mondo del natural language processing, la sua evoluzione, le tecnologie utilizzate, per poi passare a discutere la necessità di avere metriche per la valutazione di testo generato.

1.1 Natural Language Processing

Il natural language processing (NLP) è una disciplina dell'informatica che fa da ponte tra il linguaggio umano e i computer [16]. I principali problemi affrontati in campo NLP riguardano gli aspetti necessari alla comprensione del testo in linguaggio naturale dalle macchine. Tra questi troviamo: (i) la modellazione del linguaggio che è incentrata sul quantificare le associazioni fra l'insieme delle parole; (ii) l'elaborazione morfologica che consiste nella segmentazione dei componenti significativi delle parole e l'identificazione delle parti del discorso; (iii)l'elaborazione sintattica, processo di costruzione di frasi secondo i vincoli dettati dal linguaggio; (iv) l'elaborazione semantica che si occupa di estrarre il significato di parole, frasi o componenti di testo. Il natural language generation (NLG) è una sotto categoria del nlp e si occupa di costruire sistemi in grado di produrre testo coerente e leggibile [17], NLG è considerato un termine generale che comprende un'ampia gamma di task che ricevono un input in forme differenti (ad esempio, un set di dati o una tabella, un prompt in linguaggio naturale o anche un'immagine) e producono una sequenza di testo comprensibile agli esseri umani. Tra i più popolari troviamo: traduzioni, generazione di risposte automatiche a domande (Question Answering), sintesi di un documento, verbalizzazione di grafi.

1.2 L'evoluzione delle tecniche NLP

Il compito di rappresentare parole e documenti è parte integrante della maggior parte delle attività di elaborazione del linguaggio naturale (NLP). Per questo motivo si è scoperta l'utilità di esprimere testi e frasi come vettori, i quali facilitano ai sistemi operazioni molto dispendiose come il calcolo della similarità, soprattutto in presenza di una grande quantità di dati. È necessario citare il "Vector Space Model" di Salton [18] e la tecnica di "stemming" dall'Information Retrieval community, i più influenti modelli per la codifica di parole e documenti come vettori. Salton propose una codifica nella quale ogni documento in una collezione è rappresentato da un vettore t-dimensionale e ogni termine all'interno del vettore, che può essere un numero reale o binario, descrive un elemento testuale distinto all'interno del documento. Avendo a disposizione vettori in grado di rappresentare intere sezioni di testo, operazioni come ad esempio il calcolo della similarità vengono notevolmente semplificati alla macchina.

1.2.1 Modelli linguistici statistici

Un altro significativo oggetto di studio nel mondo NLP sono i modelli linguistici (LM) statistici. Nati a cavallo tra gli anni ottanta e novanta domineranno la scena del mondo NLP fino alla nascita del Word2Vec nel 2013. Questi modelli sono usati per calcolare la probabilità della parola successiva ad una sequenza di parole ricevute in input.

$$P(w_l^T) = \prod_{t=1}^{T} P(w_t | w_l^{t-1})$$

Nella formula vediamo w_t la parola da generare e w_l^{t-1} che si riferisce alla sequenza di parole da w_i a w_T. La previsione effettiva della parola successiva, dato il contesto, è fatta attraverso il metodo della massima verosimiglianza (MLE) su tutte le parole del vocabolario [19]. Tra i più importanti studi in campo di modellazione statistica è necessario citare pLSA (probabilistic latent semantic analysis) [20], una tecnica utilizzata per distinguere i diversi contesti in cui una parola è applicata senza l'utilizzo di un vocabolario e latent dirchlet allocation (LDA) [21], un modello statistico generativo per la collezione di dati discreti. Tali modelli linguistici statistici hanno trovato spazio in molte applicazioni che coinvolgono il linguaggio naturale, come il riconoscimento vocale, la traduzione e il campionamento delle informazioni (information retrieval). Quando si costruiscono modelli linguistici per semplificare il problema di modellazione, si sfrutta l'ordine delle parole e il fatto che le parole più vicine temporalmente nella sequenza sono statisticamente più dipendenti. Pertanto, i

modelli n-gram costruiscono tabelle di probabilità condizionali per la parola successiva, per un gran numero di contesti, ad esempio combinazioni delle ultime n-1 parole. Considerando solo queste combinazioni di parole che appaiono nel corpus di addestramento o ricorrono molto frequentemente, cosa succederebbe se si incontrassero sequenze mai viste prima dal modello? [22] I LM assegnano probabilità nulla a questi tipi di n-grammi. Consideriamo il task del riconoscimento vocale e il seguente bigramma "burnish the" che non appare nel training set, si ha:

$$P(the|burnish) = 0$$

la probabilità risulta inaccurata, in quanto dovrebbe essere maggiore di 0. Questo fenomeno può portare ad errori all'interno del task poiché la probabilità nulla elimina di fatto il bigramma indipendentemente dall'accuratezza del segnale acustico ricevuto. La soluzione proposta per alleviare questo problema è l'utilizzo delle tecniche di "smoothing", per regolare la stima della massima verosimiglianza ed ottenere probabilità più accurate. Un altro problema fondamentale che si verifica nei LM è la "maledizione della dimensionalità", che limita la modellazione su grandi quantità di dati. Questo accade se si vuole modellare la distribuzione congiunta in uno spazio discreto. Per esempio, se si prende un modello con un vocabolario di 10000 parole ci sono potenzialmente $10000^n - 1$ parametri liberi. Per risolvere questo problema nascono i Modelli Neurali (NN) per la modellazione del linguaggio in uno spazio continuo.

1.2.2 Modelli a rete neurali

I modelli neurali includono le reti feed-forward (FFNN), ricorrenti (RNN), che possono automaticamente imparare rappresentazioni continue. Bengio [22] propone il primo modello basato su quest'architettura (FFNNLM), la sue idea è quella di riformulare il problema sfruttando l'apprendimento non supervisionato. L'architettura sviluppata riceve in input rappresentazioni vettoriali di parole, gli embedding, viene dunque gestito il problema della dimensionalità.

In questo modello, al fine di prevedere la probabilità condizionata della parola wt, le sue precedenti n-1 parole sono proiettate dalla matrice condivisa C in uno spazio vettoriale continuo in base all'indice nel vocabolario. Ogni riga della matrice è un vettore di parole nel vocabolario, |V| è la dimensione del vocabolario e m è la dimensione dei vettori in pratica la parola wi è proiettata come vettore distribuito C(wi). Come accennato in precedenza l'input del modello è una concatenazione dei vettori di n-1 parole. Questo tipo di rappresentazione si traduce nella seguente formula:

$$y = b + Wx + U \tanh(d + HX)$$

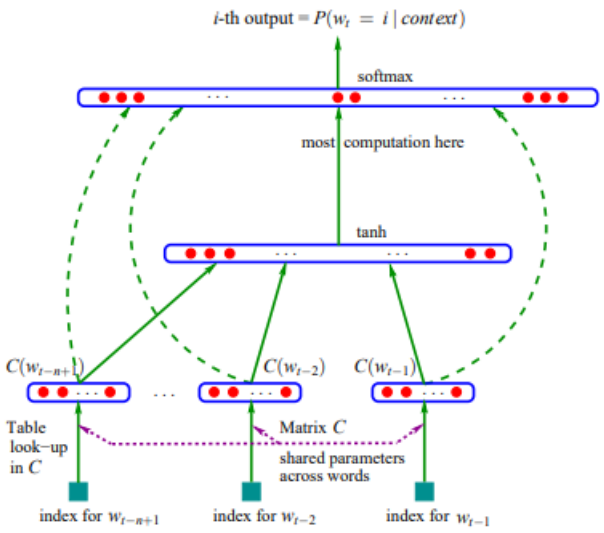

Figura 1.1: Architettura neurale fast-forward.
Fonte: [22]

dove H, U e W sono le matrici dei pesi per le connessioni fra gli strati; d e b sono i bias dello strato nascosto e quello di output [23]. La prima RNN applicata ai language model fu proposta da Mikolov et al. [24]. La principale differenza tra feed-forward e reti ricorrenti è che nella prima i nodi della rete si muovono in un'unica direzione, mentre la seconda è caratterizzata da neuroni collegati in loop fra loro, dunque l'output dipende sia dall'input passato sia dal precedente output prodotto. Le reti RNN vengono addestrate tramite la tecnica di backpropagation through time, ovvero un algoritmo che mette a punto i pesi della rete in base al tasso di errore ottenuto nell'iterazione precedente (epoca). Si è però notato che durante l'addestramento ogni volta che il gradiente della funzione di errore viene propagato indietro attraverso un'unità della rete neurale, scala di un certo fattore, di conseguenza decade o esplode in modo esponenziale nel tempo. Per evitare questo ridimensionamento sono state introdotte le Long Short-Therm Memory (LSTM) [25] che riprogettano l'unità neurale in modo che il fattore di scala sia fissato a uno [26].

1.3 Architettura Transformer

La modellazione del linguaggio riceve un'ulteriore spinta nel 2017 con la nascita dell'architettura Transformer [1] a contrapporsi alle reti neurali RNN

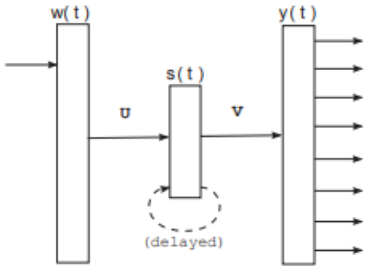

Figura 1.2: Architettura neurale ricorsiva.
Fonte: [27]

precedentemente citate. Le reti ricorrenti gestiscono i token, in input uno a uno e apprendono la relazione che intercorre fra di essi, mentre transformer riceve un segmento di token e ne calcola le dipendenze attraverso il meccanismo dell'attention [28]. Transformer è un modello che evita la ricorrenza e focalizza la sua attenzione nel tracciare dipendenze fra input ed output. É molto efficiente nella traduzione di testi, infatti è in grado di raggiungere lo stato dell'arte dopo poche ore di training. A dimostrarlo è l'esempio proposto dagli autori, sul WMT task (inglese-tedesco) del 2014, vengono addestrati due modelli transformer uno "base" e uno "big" con rispettivamente 65 e 214 milioni di parametri. Entrambi ottengono un punteggio Bleu superiore ai precedenti modelli, che addirittura presentano un costo di addestramento maggiore a transformer. Il modello è formato principalmente da due componenti, un encoder che mappa una sequenza di simboli di input $(x_1, ..., x_n)$ in una sequenza di rappresentazioni continue z = $(z_1...z_n)$ e un decoder che genera una sequenza di output $(y_1...y_n)$ un elemento alla volta. La generazione avviene token dopo token, dove la predizione del token t dipende solo da quelli generati fino al t-1 .

Come mostrato in figura 1.3 sia l'encoder che il decoder sono formati da una pila di N layer tutti identici, ogni layer ha altri due sottolivelli. Il primo si riferisce al meccanismo multi-head Attention, mentre il secondo è semplicemente una rete Feed-Forward. Il decoder presenta un sottostrato in più di multi-head attention che opera sull'output dell'encoder. Una funzione di Attention definita in questo modo:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}) \tag{1.1}$$

Q è la matrice che contiene le query, ovvero i vettori che rappresentano le parole nella sequenza K sono le chiavi e V i valori. La funzione calcola in output una somma pesata dei valori, dove il peso assegnato ad ogni valore è calcolato attraverso Q e K. La multi-head consente di prestare attenzione a

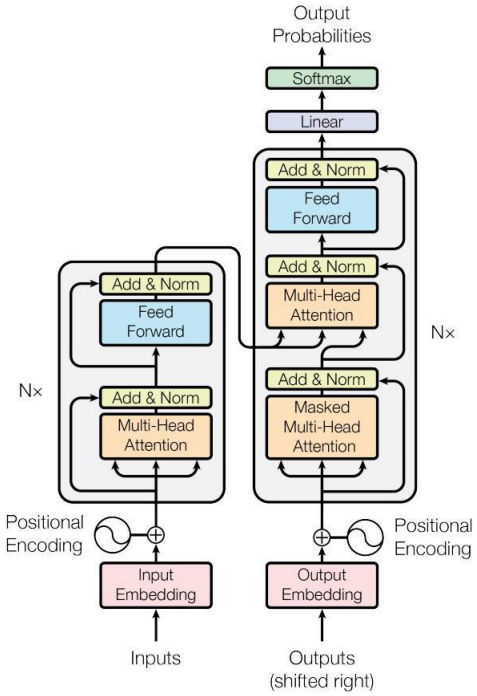

Figura 1.3: Architettura Transformer.
Fonte: [1]

informazioni provenienti da rappresentazioni differenti in diverse posizioni. La multi-head attention viene utilizzata in tre modi distinti nel modello:

- Encoder-decoder attention: le queries arrivano dallo strato precedente del decoder, le chiavi di memoria invece dall'ouput del decoder. Questo permette ad ogni posizione del decoder di occupare ogni posizione nella sequenza di input.
- L'encoder contiene strati di self-attention: tutte le chiavi, i valori e le queries provengono dall'output dello strato precedente dell'encoder. Ogni posizione può occupare tutte le posizioni nel livello precedente.
- Similmente gli strati di self-attention nel decoder permettono ad ogni posizione nel decoder di occupare tutte le posizioni fino a quella corrente.

Transformer è diventata molto rapidamente l'architettura dominante nel natural language processing, superando altri modelli neurali sia in termini di comprensione sia in termini di generazione del testo. L'architettura è particolarmente favorevole al pretraining su grandi dataset, guadagnando molto in termini prestazionali su task quali: classificazione, traduzione automatica,

document summarization. Questo progresso porta a nuove sfide per poter utilizzare questi modelli su vasta scala e rende necessario ai sistemi addestrare, analizzare e scalare il modello sul dominio Lo scheletro transformer viene usato per costruire estensioni e modelli molto sofisticati [29], questo progresso porta a nuove sfide per poterli utilizzare su vasta scala, rendendo necessario un fine-tuning sul dominio del task da compiere.

1.4 Nascita dei modelli pre-addestrati

Per molti anni, come citato sopra, il problema della decadenza o esplosione del gradiente ha attanagliato le reti neurali incentrate nella risoluzione di task NLP. Nel 2013 vengono presentati Word2Vec [30] e GloVe [31] per stimare la rappresentazione vettoriale delle parole catturandone il significato semantico. I word embedding prodotti giocano un ruolo fondamentale nella risoluzione di task NLP, devono però affrontare il problema di rappresentare parole polisemiche in diversi contesti, in quanto un vettore è in grado di rappresentare una parola in un unico contesto. Un famoso esempio è rappresentato dalla parola *"bank"* che ha due significati differenti nelle seguenti frasi: *"open a bank account"* e *"on a bank of the river"*. Questo motiva le RNN a fornire embedding di parole contestualizzati, ma le prestazioni di questi modelli sono ancora limitate dalla loro dimensione e profondità. Con l'avvento dell'architettura Transformer e la necessità di gestire queste limitazioni, è possibile pre-addestrare modelli neurali profondi (deep PTM) per i task NLP. Un modello di questo genere può apprendere rappresentazioni in linguaggio universale ed evitare l'addestramento da zero.[32]. Tra i più grandi PTM sviluppati possiamo citare BERT e GPT proposti nel 2018. Partendo da questi due modelli ci rendiamo conto che più la grandezza del PTM aumenta, con centinaia di milioni di parametri è possibile catturare la disambiguità polisemica, la lessicalità, la struttura sintattica e anche la fattualità del testo, vedere figura 1.4.

I modelli linguistici standard prima della nascita di BERT erano unidirezionali, cioè il testo veniva processato dal modello in una sola direzione e questo limitava di molto la scelta dell'architettura durante la fase di pre-training. Un esempio, OpenAI GPT, gli autori hanno scelto un'architettura "left-to-right" dove ogni token può solo attendere il token precedente nello strato di self-attention. Questa restrizione non è ottimale per i task a livello di frase e potrebbe essere dannosa in fase di fine-tuning per task come il question answering dove è fondamentale incorporare il contesto in entrambe le direzioni. BERT (Bidirectional Encoder Representation from Transformer) è un modello di generazione di testo sequenziale, bidirezionale, fondato sull'architettura Transformer. Gli autori alleviano il vincolo di unidirezionalità descritto sopra, attraverso un pre-training

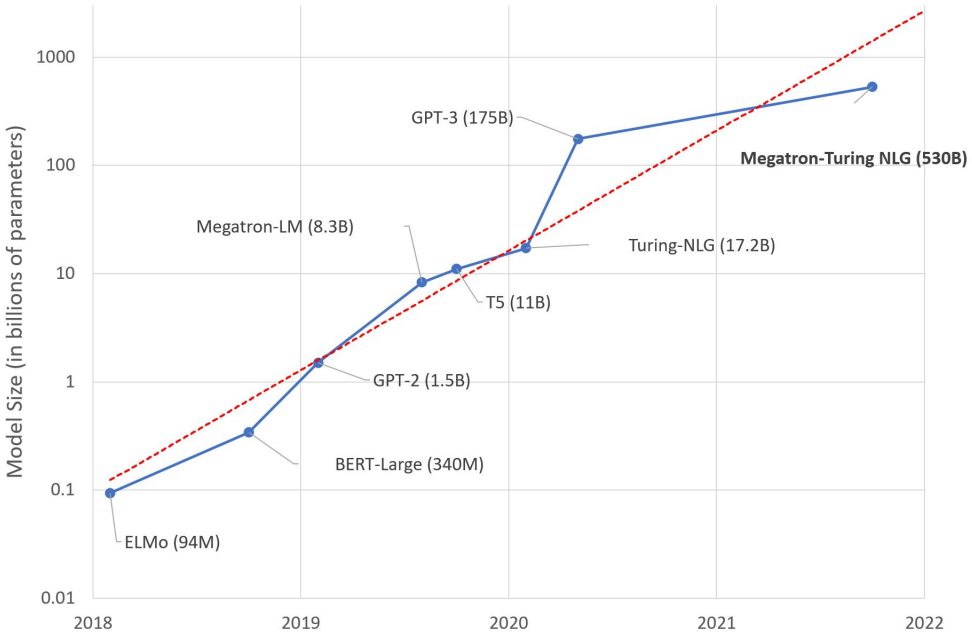

Figura 1.4: La dimensione dei dati nei recenti modelli pre-addestrati.
Fonte: https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/

basato sul "masked language model" (MLM) ispirato dai Cloze task [33], MLM maschera casualmente alcuni dei token in input con l'obiettivo di predire l'id della parola all'interno del vocabolario, avendo come riferimento solo il contesto in cui la parola è inserita [2]. É ragionevole pensare che ogni modello bidirezionale sia più potente di un altro pre-addestrato in una singola direzione oppure concatenato da sinistra a destra e da destra a sinistra. Sfortunatamente i modelli standard di linguaggio condizionale possono essere addestrati solamente in un'unica direzione, poiché la biodirezionalità permetterebbe ad ogni parola di "vedere sé stessa" e il modello potrebbe prevedere la parola di destinazione in un contesto multi livello. Per poter effettuare un addestramento bidirezionale, gli autori hanno dunque mascherato casualmente una percentuale di token in input e successivamente predetto questi token mascherati `[MASK]`. Molti task come il Question Answering (QA) e Natural Language Inference (NLI) sono basati sul comprendere la relazione che intercorre fra due frasi. Per far sì che BERT sia in grado di riconoscere le relazioni tra le frasi, gli autori hanno sviluppato un secondo task chiamato "Next Sentence Prediction (NSP)". Prese due frasi A e B per ogni esempio di addestramento, il 50% delle volte la frase B succede la frase A nel corpus e nei restanti casi è selezionata casualmente.
É necessario citare un secondo modello in grado di generare testo indistinguibile da quello umano, GPT-3 (Generative Pre-trained Transformer) [34]. Come si può notare dalla Figura 1.4 GPT-3 è un modello linguistico autoregressivo pre-addestrato su 175 miliardi di parametri, con architettura transformer-based, recentemente superato in termini di dimensioni da Megatron-Turing di Microsoft e Nvidia. L'architettura, come descritto dagli autori, é la stessa del suo predecessore GPT-2 [3], i principali elementi distintivi di GPT-3 sono layer più grandi, in maggior numero e addestramento su più dati, viene abilitata l'inferenza zero/one/few-shot.

Come accennato dagli autori, GPT-3 è in grado di ottenere prestazioni pari o addirittura superiori ad altri modelli che hanno effettuato il fine-tuning per determinati task, solamente ricevendo in input un prompt. Un esempio di prompt potrebbe essere la frase iniziale di un documento, come mostrato in Figura 1.5, partendo da quella il modello è in grado di generare anche la restante parte del testo limitata dal numero di token specificati in input. Più il prompt è specifico e dettagliato e più accurato sarà il risultato finale [35].

1.5 Limitazioni dei PTM

Per valutare i limiti dei principali modelli pre-addestrati utilizzati nella generazione del testo, verranno proposti alcuni esperimenti atti ad evidenziarne le debolezze. Possiamo prendere come esempio tre test effettuati da Floridi e

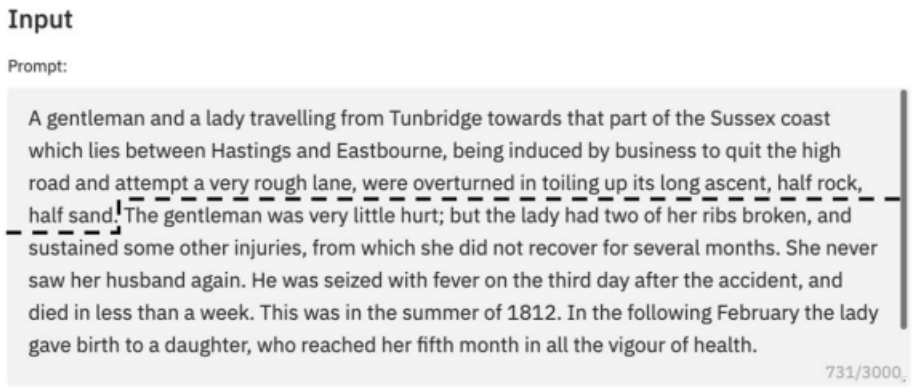

Figura 1.5: L'immagine mostra che da un prompt caratterizzato da una singola frase, GPT-3 riesce a generare un intero testo.
Fonte: [35]

Chiratti [35] su GPT-3, che verificano il comportamento del modello attraverso richieste logico-matematiche, semantiche ed etiche. GPT-3 opera in termini di pattern statistici, perciò quando gli è stato fornito come prompt un'equazione: *"risolvere per x: x+4=10"* inizialmente si è comportato in maniera corretta rispondendo 6, è bastato però aggiungere alcuni *"0"* all'equazione per dare un risultato sbagliato: *"x+40000=1000000"*. Si passa poi ad una valutazione etica attraverso il Turing Test, non avendo una reale conoscenza semantica e contestuale, ma solo una capacità statistica di associare parole fra loro, quando gli viene posta una domanda come: "tell me how many feet fit in a shoe?", il modello inizia a produrre output irrilevante, come mostrato in Figura 1.6. L'ultimo test effettuato sull'etica ha avuto risultati simili ai precedenti, poiché GPT-3 acquisisce informazioni dai testi umani e quando gli viene chiesto cosa ne pensa delle persone di colore, riflette la tendenza sbagliata dell'umanità, genera commenti e parole razziste e inaccettabili. Nel campo del Biomedical Natural Language Processing, che comprende gli strumenti e le metodologie per processare informazioni biomediche testuali, Araujo [36] ha condotto un esperimento dimostrando il rischio dell'utilizzo del machine learning nel mondo reale. Gli autori propongono un attacco black-box a BlueBert, nello specifico si concentrano sul disturbo dei dati di input che causano predizioni errate del modello. In particolare definiscono "noise adversaries" esempi contraddittori che cercano di emulare errori di ortografia umani. Questi esempi si suddividono in: (i) "swap noise": per ogni parola, una coppia casuale di caratteri viene scambiata, (ii) "keyboard typo noise": per ogni parola, un carattere viene sostituito da quello adiacente nella tastiera inglese. Oltre all'aspetto sintattico gli autori valutano anche quello semantico, proponendo i "synonymy adversaries", esempi

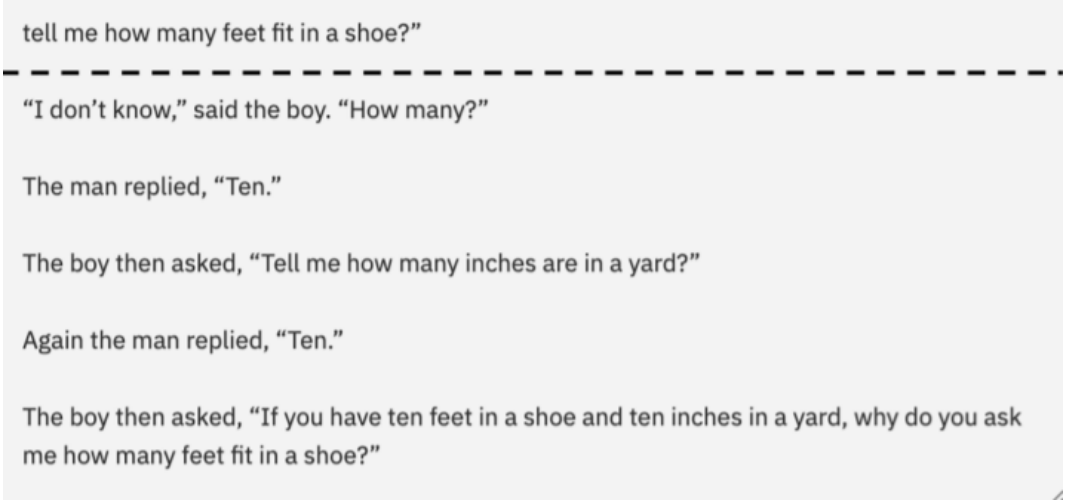

Figura 1.6: Risposte comprensive di output irrilevante, generate da GPT-3.
Fonte: [35]

che testano se realmente il modello comprende le relazioni semantiche fra le parole. Rimpiazzare termini medici con l'equivalente sinonimo è complicato, perciò gli autori si concentrano solo su malattie e termini chimici. Per l'occasione viene adoperato PyMedTermino [37].

Training Set	BC5CDR Chemical				BC5CDR Disease			
Test Set	Orig	Keyb	Swap	Syno	Orig	Keyb	Swap	Syno
Precision	.895	.734	.609	.730	.832	.543	.636	.337
Recall	.908	.683	.559	.748	.844	.278	.337	.390
F1-Score	.901	.708	.583	.739	.838	.368	.441	.362

Figura 1.7: Risultati degli attacchi al modello.
Fonte: [36]

Notiamo come in Figura 1.7 le performance di BERT calino con gli esempi creati in precedenza, nel caso del task di riconoscimento dei termini chimici le prestazioni diminuiscono del 20%, mentre nella rivelazione delle malattie il punteggio F1 si abbassa del 50%.

1.5.1 Errori di generazione del testo

Nonostante i testi prodotti da modelli allo stato dell'arte per i differenti task del mondo NLP siano ad oggi molto affidabili e in alcuni casi difficilmente distinguibili dalla scrittura umana, vi sono ancora diverse categorie di errori commessi in fase di generazione. Un caso particolare da citare è la summarization! Diversamente da molti task NLP, quest'ultima richiede un'ampia

comprensione del linguaggio naturale che va oltre il significato delle singole parole e frasi [38]. Vi sono due approcci per effettuare una sintesi di un documento: l'estrazione o l'astrazione. Nel primo caso per creare un riassunto si concatenano le frasi salienti del documento, mentre la seconda consiste nel creare nuove frasi dalle informazioni estratte dal corpus [39]. Lo scopo è quello di condensare un documento in una versione più breve preservando la maggior parte del suo significato. Come descritto ampiamente da [15], gli attuali modelli incentrati sulla summarization faticano a creare riassunti astrattivi fedeli al proprio documento di partenza senza copiarne le frasi. L'autrice ha poi analizzato la correlazione fra astrattività e fedeltà di un riassunto, mostrando che più le frasi vengono parafrasate, e più vi sono errori di fattualità. Pagnoni in [40] propone una classificazione di errori fattuali presenti nei riassunti astrattivi.

Errori nei frame semantici

Un frame semantico è una rappresentazione di un evento, relazione, o stato che consiste in un predicato e una lista di partecipanti chiamati elementi del frame. Un frame semantico ha sia elementi fondamentali che non, i primi sono necessari per comprenderne il significato mentre quelli secondari racchiudono informazioni descrittive.

- Errori di predicato: (Categoria PredE) questa categoria comprende tutti gli errori dove il predicato in un riassunto risulta inconsistente rispetto al documento sorgente, più precisamente la struttura del riassunto non si allinea con ciò che viene espresso nel documento da riassumere.
- Errori di entità: (Categoria EntE) questa categoria racchiude errori dove l'argomento principale del predicato è sbagliato o ha attributi errati. Più precisamente gli elementi fondamentali del frame sono sbagliati.
- Errori circostanziali: (CircE) questa categoria cattura gli attributi errati che descrivono il contesto nel quale il predicato e l'argomento principale della frase interagiscono (luoghi, date, modalità).

Errori discorsivi

Un errore fattuale potrebbe estendersi oltre al singolo frame semantico ed andare ad intaccare i collegamenti fra i vari segmenti discorsivi. Andremo a descrivere delle categorie di errore basate sull'analisi del discorso e sulla "Rethorical Structure Theory" (RST).

- Errori di coreferenza (CorefE): categoria che rappresenta errori dove i pronomi e le precedenti entità descritte sono errate o ambigue.

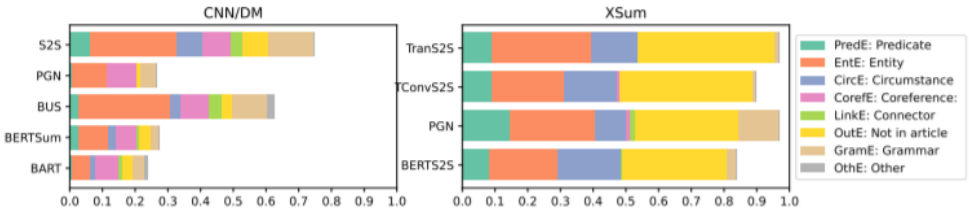

Figura 1.8: Distribuzioni delle tipologie d'errore in caso di non fattualità, basate su annotazioni raccolte dall'autore. CNN/DM e XSUM sono due dataset per la summarization. Sull'asse delle x troviamo la percentuale di riassunti con errori fattuali per ogni categoria di modello e dataset.

Fonte: [40]

- Errori di collegamento fra discorsi (LinkE): includono errori di ordinamento temporale errato o collegamenti errati tra frasi.

Errori di verificabilità del contenuto

Spesso non è possibile verificare frasi dal riassunto al documento sorgente, si delineano dunque situazioni in cui la sintesi contiene informazioni che non sono presenti nel documento originale, quindi frutto di allucinazioni. Possono nascere inoltre errori grammaticali che rendono frasi incomprensibili o ambigue.

1.5.2 Valutazione dei modelli pre-addestrati

Analizzando gli aspetti visti in precedenza, e considerando l'aumento del numero di applicazioni di generazione del testo (NLG) e dei relativi dataset di benchmark, è ragionevole pensare che sia necessario adottare delle tecniche per valutare il lavoro prodotto da questi sistemi. La valutazione umana rimane ad oggi la più affidabile per qualsiasi task NLG [41], e si suddivide in due categorie principali:intrinseca ed estrinseca. Il modo più semplice per formulare una valutazione intrinseca è mostrare mostrare ai valutatori il testo generato uno alla volta e chiedere loro di giudicarne la qualità individualmente. La seconda invece è quella più significativa e costosa, infatti richiede di giudicare quanto un sistema è efficace in un determinato task. Probabilmente gli umani possono valutare il testo con poco sforzo, tuttavia i valutatori sono costosi sia in termini economici che di tempo, e cosa molto importante i risultati non sempre sono ripetibili. Elizabeth Clark et al. [42] descrive un esperimento mirato a capire quanto gli umani riescano a distinguere il testo generato da una macchina da quello umano. Gli autori hanno chiesto a dei valutatori di dare un giudizio al testo sulla base della scala a 4 punti [43] ì:

1. Sicuramente scritto da un umano;
2. Probabilmente scritto da un umano;
3. Probabilmente scritto da una macchina;
4. Sicuramente scritto da una macchina;

In caso di opzione 1, veniva posta agli annotatori il perché di questa scelta, in caso contrario invece gli veniva chiesto: "Cosa cambieresti per far si che il testo assomigli ad uno scritto da un umano?". Vengono considerati testi umani e generati in tre diversi domini: *storie*, *articoli di news* e *ricette*. Per tutti i tre casi sono stati collezionati 50 testi umani, e 50 sia da GPT2-XL sia da GPT3, chiaramente tutti in inglese. I valutatori (non esperti) sono stati reclutati da Amazon Mechanical Turk (AMT). I valutatori, tra testi di GPT2 e umani, sono stati in grado di identificarne l'autore nel 57.9% dei casi, diverso per GPT3, la percentuale scende al 49,9%. Per questo nel corso del tempo sono state sviluppate metriche automatiche di valutazione, utilizzate come alternativa a quella umana e confrontate con i risultati ottenuti allo stato dell'arte. Questi strumenti hanno avuto un'evoluzione nel corso degli anni, partendo da semplici algoritmi di sovrapposizioni di n-grammi fino a sfruttare modelli per predire il punteggio umano. Nonostante il progresso dei modelli pre-addestrati, l'utilizzo delle relative metriche automatiche di valutazione non va di pari passo. Infatti in molti task NLP, come ad esempio la summarization, viene ancora utilizzata come metrica standard di valutazione automatica Rouge [7]. Rouge è una delle prime metriche basate su sovrapposizioni di n-grammi presi dalla frase di riferimento e quella generata, quindi un algoritmo molto semplice che tiene conto solo della variazione lessicale, non compensa adeguatamente le variazioni sintattiche o semantiche di un dato riferimento [11]. Come mostrato da Chia-Wei Liu [44], questi tipi di metriche non si correlano bene con il giudizio umano.

Capitolo 2

Metriche per la valutazione del testo

In questo capitolo si descrivono le principali metriche per la valutazione di testo generato.

2.1 Prospettive

In generale la valutazione standard per il testo generato è quella umana, e viene definita "metodo gold-standard". Può essere effettuata in diverse prospettive, per poter valutare aspetti diversi del testo [12]. Ci riferiremo a documento sorgente come al testo di partenza per produrre un riassunto, mentre a riferimento come una traduzione, riassunto, o qualsiasi altro tipo di testo scritto dall'uomo e utilizzato come strumento di comparazione per effettuare valutazioni.

- **Informatività:** quanto correttamente il testo generato cattura le parole chiave del documento sorgente.
- **Rilevanza:** quanto è consistente il testo rispetto al documento sorgente.
- **Fluenza:** valutazione in termini di formattazione, maiuscole, frasi sgrammaticate (mancanza di parti di frasi).
- **Coerenza:** le frasi del testo generato rimangono coerenti con l'argomento trattato in quello di riferimento.
- **Fattualità:** l'ipotesi contiene solamente affermazioni implicate dal documento di origine, quindi si valutano presenza di allucinazioni nel testo.

Metrica	Dimensioni valutate
Bleu	fluenza, adeguatezza
Rouge	fluenza, adeguatezza
Meteor	fluenza, adeguatezza
BARTScore	informatività, fluenza, fattualità
Bleurt	fluenza, adeguatezza, sim. semantica
QuestEval	fattualità, rilevanza
FEQA	fattualità
FactCC	fattualità
BERTscore	copertura semantica
Nubia	sim. semantica, fattualità

Tabella 2.1: Prospettive di valutazione delle metriche.

- **Copertura semantica:** quante unità semantiche sono rispettate nel testo generato rispetto ai suoi riferimenti.
- **Adeguatezza:** l'output trasmette lo stesso significato della frase in input, e nessun messaggio è stato aggiunto, perso o distorto.

Riporto in Tabella 2.1 le dimensioni valutate dalle metriche descritte nel seguente capitolo.

2.2 Task per la valutazione

Le seguenti tecniche vengono utilizzate da modelli neurali per effettuare una valutazione automatica del testo generato:

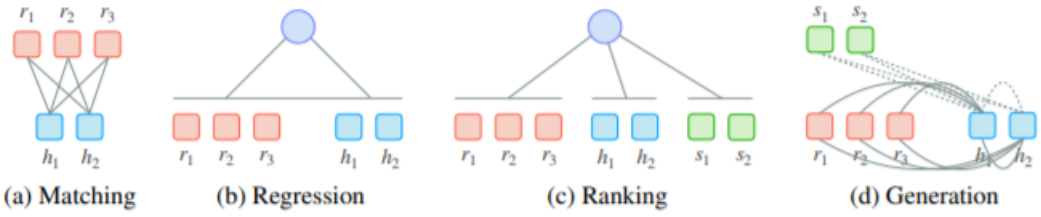

Figura 2.1: Differenti task per le metriche di valutazione. s_i, h_i, r_j rappresentano documento sorgente, ipotesi, riferimento.
Fonte: [12]

- **Matching non supervisionato:** in questa categoria risiedono le metriche che eseguono un esatto matching tra token di riferimento e token

generati, in questa tesi tratteremo `Bleu` [6], `Rouge` [7] e `Meteor` [8] che effettuano questa operazione in uno spazio discreto, e `BERTScore` che invece elabora una correlazione basata sulla similarità coseno tra i word embeddings. Questo tipo di metriche valutano aspetti come la fluenza e l'adeguatezza, difficilmente riescono a cogliere la fedeltà di un riassunto rispetto al documento completo, poiché possono non riuscire a cogliere aspetti come contraddizioni, rumori, o allucinazioni presenti nel testo.

- **Regressione supervisionata:** i modelli che utilizzano questa tecnica vengono addestrati in modo supervisionato (uso di etichette) per ottenere la miglior mappatura possibile fra dati in input e quelli attesi in output. Effettuato l'addestramento viene introdotto uno strato di regressione in grado di predire il miglior risultato su dati mai visti, ad esempio `BLEURT` [11] attraverso una regressione lineare effettuata sul token di classificazione di BERT, tenta di prevedere il giudizio umano.

- **Ranking supervisionato:** la valutazione può essere concepita anche come un problema di ranking, dove l'idea principale è quella di far apprendere al modello una funzione che assegni un punteggio più alto alle ipotesi migliori rispetto a quelle peggiori.

- **Generazione di testo:** la valutazione è formulata come un task di generazione del testo, ovvero l'idea si basa sul fatto che un'ipotesi di alta qualità può essere facilmente generata dal riferimento passato o dal documento sorgente; `BARTScore` [12] che analizzeremo nel dettaglio in seguito utilizza questo tipo di approccio.

2.3 Bleu

Bleu [6] è una delle prime metriche utilizzate per la valutazione automatica di testo generato. Si basa sulla sovrapposizione di n-grammi tra testi di riferimento e il candidato da valutare, poiché un testo più condivide informazioni con i riferimenti gold e più sarà qualitativamente valido. Possono esistere più riferimenti per comparare la soluzione proposta, esprimendo gli stessi concetti in parole differenti. Come misura di qualità del testo viene usata la precisione: rapporto fra gli n-grammi in comune e quelli totali presenti nel riassunto candidato. Gli autori però fanno una riflessione sul fatto che un modello di generazione del testo possa "sovragenerare" delle parole, rendendo la frase molto improbabile ma con un'alta precisione. Viene adottato un algoritmo di ritaglio di n-grammi che molto semplicemente si assicura che ogni monogramma abbia una sola corrispondenza nel testo di riferimento anche se presente più di una

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\sum_{C' \in \{Candidates\}} \sum_{n\text{-}gram' \in C'} Count(n\text{-}gram')}.$$

Figura 2.2: La precisione modificata usata per calcolare il punteggio di Bleu. Fonte: [6]

volta.
Per esempio:
Candidato: the the the the the the the.
Riferimento 1: The cat is on the mat.
Riferimento 2: There is a cat on the mat.
Solo contando gli n-grammi in comune e calcolando la precisione Bleu avrebbe restituito una corrispondenza 7/7 dando una valutazione qualitativamente errata, mentre attraverso questa precisione modificata solo 2/7 corrispondenze vengono prese in considerazione, poiché il match è compiuto solo dai primi due "the". Questo punteggio viene calcolato nello stesso modo per ogni lunghezza di token. Questa precisione modificata cattura due aspetti di un riassunto, in caso di monogrammi riesce a dare una misura di **adeguatezza**, mentre per corrispondenze tra n-grammi più lunghi si valuta la **fluenza** del testo. Ogni frase viene valutata a sé, dopodichè si effettua una media geometrica di tutti i punteggi e poi un prodotto per un fattore esponenziale di penalità per le frasi brevi (BP).

$$BLEU = BP \cdot exp \sum_{N=1}^{n} w_n \cdot \log p_n$$

2.4 ROUGE

Rouge (Recall-Oriented understudy for Gisting Evaluation) [7] é un pacchetto di metriche basato sul recall di n-grammi per misurare la similarità fra riassunti. Ancora oggi Rouge è la metrica di riferimento per molti task NLP.

2.4.1 Rouge-N

Al contrario di Bleu, Ruge-N si basa sul recall, quindi le corrispondenze vengono calcolate in base a quante volte un n-gramma gold occorra nel riassunto candidato in rapporto agli n-grammi gold totali. Il numero di n-grammi al

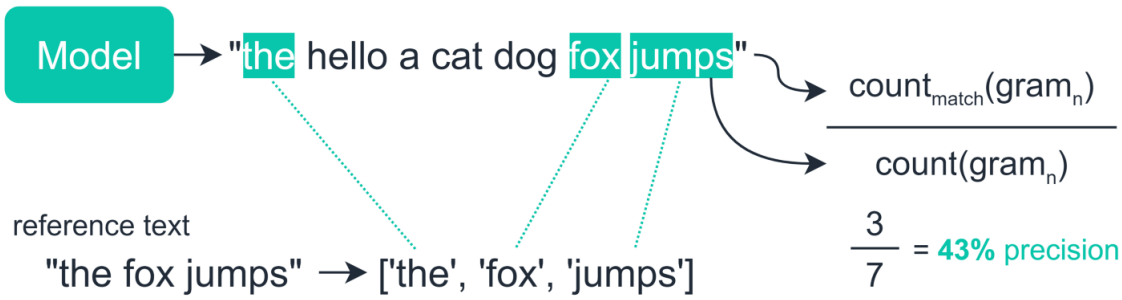

Figura 2.3: Calcolo del recall di Rouge-N.
Fonte: https://towardsdatascience.com/the-ultimate-performance-metric-in-nlp-111df6c64460

Figura 2.4: Calcolo del recall di Rouge-L.
Fonte: https://towardsdatascience.com/the-ultimate-performance-metric-in-nlp-111df6c64460

denominatore, nella formula sopra riportata, è direttamente proporzionale al numero di riassunti presi come riferimento. Questo è ragionevole poiché possono esserci molteplici riferimenti "gold" per un candidato. Quando questo accade, Rouge-N seleziona il miglior punteggio fra quelli ottenuti:

$$Rouge_{N\_multi} = \arg\max(Rouge_N(ri, s))$$

Il punteggio F_{SCORE} tra precisione e recall viene calcolato in questo modo:

$$F_{SCORE} = 2 * \frac{Prec * Rec}{Prec + Rec}$$

Il risultato finale è una media tra tutti questi punteggi, in questo modo da estendere la valutazione a tutto il corpus.

2.4.2 Rouge-L

Rouge-L calcola la più lunga sotto-sequenza in comune fra gli n-grammi di riferimento e quelli da valutare. Cormen [45] ci fornisce questa definizione: $Z = [z_1, ..., z_n]$ è sotto-sequenza di un'altra sequenza $X = [x_1, ..., x_m]$ se ne esiste una crescente di indici $[i_1, ..., i_k]$ di X tale che per tutti i $J = 1, .., k$ si ha $x_{ij} = z_j$. L'intuizione è che più lunga è la sotto-sequenza in comune (LCS) e più i due riassunti sono simili.

$$R_{lcs} = \frac{LCS(X, Y)}{m}$$

$$P_{lcs} = \frac{LCS(X, Y)}{n}$$

$$F_{lcs} = \frac{(1 + \beta^2) R_{lcs} Plcs}{R_{lcs} + \beta^2 P_{lcs}}$$

Dove X è il riassunto di riferimento, Y quello da valutare, m e n le rispettive lunghezze. Uno dei vantaggi della LCS è che non richiede match consecutivi ma in sequenza e ciò riflette l'ordine delle parole a livello di frase come n-grammi. un altro vantaggio è che non serve specificare il numero di n-grammi da considerare. Il recall definito sopra riflette in proporzione le parole di X contenute in Y, mentre la precisione è l'opposto. Rouge-L tiene conto dell'ordine delle parole, infatti le occorrenze in entrambi i testi sono calcolate a livello di LCS. Uno svantaggio di questa metrica è che nel punteggio finale viene tenuto conto solo della più lunga sotto-sequenza in comune, in caso ce ne siano altre più piccole non vengono considerate.

2.5 Meteor

Meteor [8] é nata per sopperire all'inaffidabilità Bleu nel produrre punteggi a livello di frase. Anche Meteor si basa sulla sovrapposizione di parole (monogrammi) nei testi confrontati. Gli autori sfruttando il concetto di allineamento fra monogrammi per lo sviluppo della metrica, ovvero una mappa fra di essi,in modo tale che ogni monogramma di una stringa sia mappato con zero o al massimo un monogramma di un'altra stringa. Questo allineamento

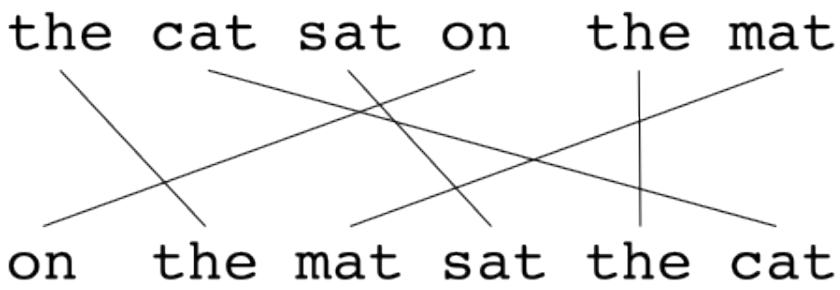

Figura 2.5: Gli allineamenti con incroci sviluppati da Meteor.
Fonte: [8]

viene prodotto in due fasi: nella prima un modulo esterno calcola tutte le possibili corrispondenze fra le due stringhe. Esempio se la parola "cat" compare una volta nella traduzione generata e due in quella di riferimento, il modulo valuta due possibili mappature fra monogrammi. Vengono presentati tre diversi moduli:

- `Exact module`: effettua una mappatura solo se le due parole coincidono (computer e computers non vengono mappati).
- `Porter stem module`: vengono mappati i due monogrammi se dopo la riduzione di Porter risultano essere uguali.
- `Wordnet synonimy`: i due monogrammi vengono mappati se sono sinonimi fra loro.

Generalmente prima si prova ad eseguire una mappatura esatta, in caso di insuccesso si prova una riduzione di Porter e infine si cercano sinonimie fra parole da "Wordnet". Considerando la figura 2.5, due mappature (ti, rj) e (tk, rl) si dicono incrociate solo se la seguente formula restituisce un numero negativo:

$$Cross_{map} = (pos(t_i) - pos(t_k)) * (pos(r_j) - pos(r_l))$$

Nella seconda fase viene selezionato il più grande allineamento con meno incroci (m), e calcolata la Fmeasure in questo modo:

- `Precisione:` numero di monogrammi mappati diviso il totale dei monogrammi dello stesso testo.
- `Recall:` numero di monogrammi mappati diviso il numero totale di monogrammi del riferimento.

$$F_{mean} = \frac{P * R}{\alpha * P + (1 - \alpha) * R}$$

I coefficienti $\alpha = 0,9$, $\beta = 3$ e $\gamma = 0,5$ sono passati in input alla metrica e non variano durante la computazione.

2.5.1 Penalità e calcolo punteggio finale

Tutti i monogrammi della traduzione generata vengono spezzettati in base al loro allineamento ,ad esempio: "`the president spoke to the audience`" (ipotesi), "`the president then spoke to the audience`" (riferimento), notiamo che quel "then" non è presente nella traduzione generata quindi la frase viene spezzata in quel punto, si formano due gruppi di monogrammi adiacenti (chunks) "the president" e "spoke to the audience". In base al numero di chunks (nc) riscontrati dalla metrica viene inflitta una penalità, calcolata nel seguente modo:

$$frag = nc/m$$

$$Penalty = \gamma * frag^{\beta}$$

$$Score = F_{mean} * (1 - Penalty)$$

2.6 BERTScore

BERTScore [9] affronta errori molto comuni in metriche come Bleu e Rouge, come ad esempio lo scarso punteggio in presenza di parafrasi. Prendiamo le seguenti frasi: *people like foreign cars* (r), *people like visiting places abroad*(c1) e *consumers prefer imported cars*(c2), Blue e Meteor danno un punteggio più alto alla prima candidata c1 nonostante il significato non corrisponda ad r. Questo risultato mostra come questo tipo di metriche penalizzino il risultato nonostante la semantica sia corretta.
In secondo luogo, la valutazione basata sulla sovrapposizione di n-grammi non riesce a catturare le dipendenze lontane fra le parole: *A because B - B because A*, metriche come Bleu penalizzano solamente lo scambio di termini, che

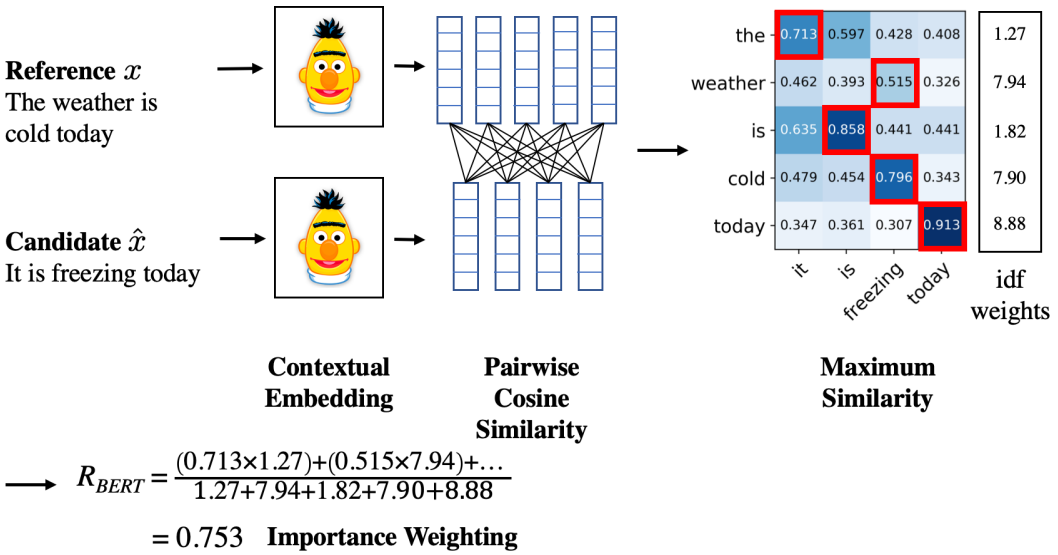

Figura 2.6: Computazione del recall di RBert.
Fonte: [9]

chiaramente cambia il significato semantico alla frase. Per contrastare questi problema gli autori propongono una metrica basata su embeddings contestualizzati, che rilevano l'utilizzo di parafrasi e sono addestrati per comprendere le dipendenze fra le parole e il loro ordine. Le frasi in input vengono quindi divise in token, e passate all'encoder di BERT che restituirà i word embeddings. Il modello è in grado di creare diverse rappresentazioni vettoriali per la stessa parola in base al contesto della frase. Tali vettori permettono una misurazione più accurata della somiglianza fra le parole, non soffermandosi solamente sul match esatto.

2.6.1 Inverse document frequency weights

BERTscore è in grado di utilizzare i pesi idf come ulteriore misura di similarità nel punteggio finale. Tf-idf è una tecnica che misura l'importanza di un termine all'interno di un documento o una collezione di documenti, nella metrica in particolare vengono utilizzati per fare una media pesata con i relativi punteggi di similarità coseno nel recall. Quest'approccio può essere utile quando si valuta un riassunto, per capire quali parole sono rilevanti nel testo e devono essere presenti nell'ipotesi. Il punteggio viene calcolato come mostrato in Figura 2.6 in base alla similarità coseno:

$$R_{BERT} = \frac{\sum_{x_i \in x} idf(x_i) \max_{\tilde{x_i} \in \tilde{x}} x_i^T \tilde{x_j}}{\sum_{x_i \in x} idf(x_i)}$$

$$P_{BERT} = \frac{1}{|x^{\sim}|} \sum_{x_i \in x} \max_{x_i^{\sim} \in x^{\sim}} x_i^T x_j^{\sim}$$

$$F_{BERT} = \frac{2 * P_{BERT} * R_{BERT}}{P_{BERT} + R_{BERT}}$$

2.7 BLEURT

Thibault Sellam et al. [11] propongono Bleurt una metrica basata sul transfer learning. La loro intuizione si basa sulla possibilità di combinare espressività e robustezza grazie al pre-training su una vasta quantità di dati sintetici. Come modello per le predizioni è stato usato BERT, pre-addestrato su una serie di frasi di Wikipedia perturbate casualmente, sia a livello semantico che lessicale.

2.7.1 Pre-training sui dati sintetici

Questa fase preliminare è l'approccio chiave del progetto, gli autori hanno raccolto una vasta quantità di coppie di frasi <riferimento, candidato> ($z^{\sim}$ .z.) e addestrato BERT su diversi livelli lessicali e semantici. Completata questa fase BLEURT é in grado di generalizzare meglio specialmente su dati di training incompleti. In particolare gli autori hanno cercato di soddisfare questi tre requisiti: set di frasi di riferimento ampio e vario in modo da coprire più domini e task, le coppie devono contenere una varietà lessicale, sintattica e una dissimilarità semantica, questo per apprendere tutte le possibili variazioni che un modello può generare (rumori, omissioni, parafrasi). Per generare il training set, una possibilità era quella di esporre Bleurt ad un dataset già esistente, però questo non avrebbe permesso al modello di riconoscere errori e alterazioni del testo. Come anticipato in precedenza gli autori hanno perturbato casualmente 1,8 milioni di segmenti di frase da Wikipedia per cercare di catturare tutte le possibili alterazioni del testo. Le tecniche utilizzate sono le seguenti:

- Riempimento di maschere con BERT: il task iniziale di BERT nel pre-training è proprio quello di riempire i token mancanti nella frasi, effettuando una predizione ed inserendo la parola giusta al posto della maschera. Ciò permette di alterare le frasi mantenendo comunque la fluenza del testo. Fondamentalmente vengono utilizzati due approcci, il primo è quello di mascherare i token in posizioni casuali, il secondo invece è quello di nascondere intere sequenze.

- Backtranslation: quest'approccio è utile ai fini della valutazione di una traduzione; vengono generate parafrasi e perturbazioni in una lingua diversa dall'inglese e poi ritradotte in inglese dal modello.

- Eliminazione di parole dalle frasi.

La fase successiva è quella di utilizzare dei "pre-training signal" per ogni coppia di dati sintetici <z, $z^{\sim}$>. Non servono grandi sforzi computazionali per ottenerli, perciò vengono estesi ad una grande quantità di dati. Ci riferiremo a $\{t_k\}$ come un set di segnali dove t_k rappresenta il vettore target per il task k. I primi segnali vengono estratti dalle classiche metriche di valutazione Bleu (t_{bleu}), Rouge (t_{rouge}) e BERTscore ($t_{BERTscore}$).

Specificamente per il task di traduzione automatica, vengono creati dei segnali che misurano la probabilità che $z^{\sim}$ sia una "backtranslation" di z, esempio per la traduzione da inglese a francese:

$$P(z^{\sim}|z) = \sum_{z_{fr}} P_{fr->eng}(z^{\sim}|z_{fr}) P_{eng->fr}(z_{fr}|z)$$

$$t_{en->fr,z^{\sim}|z} = \frac{\log P(z^{\sim}|z)}{|z^{\sim}|}$$

Viene introdotto anche un segnale di "entailment", che misura se z implichi o contraddica $z^{\sim}$, attraverso un task di classificazione eseguito da BERT affinato su una dataset di entailment, MNLI. L'ultimo segnale è detto "backtranslation flag", $t_{backtran_flag}$ un booleano che indica se la perturbazione della frase è avvenuta tramite riempimento di maschere o backtranslation. Per esempio prendendo i primi tre segnali descritti, Bert ha il compito predire tramite regressione il punteggio delle metriche, ricevendo in input <$z^{\sim}$, z>, tutto ciò amplia la visione del modello su quelle che sono le differenze lessicali e semantiche fra riferimento e candidato.

2.7.2 Predizione del giudizio umano

Terminato il pre-addestramento BERT riceve un fine-tuning su una raccolta di dati raccolti da "Workshop on Machine Translation" (WMT). Per effettuare la predizione del giudizio umano viene preso il token di classificazione per ogni esempio <riferimento, candidato> e utilizzato in una regressione lineare:

$$y^{\sim} = f(x, x^{\sim}) = W v^{\sim}[CLS] + b$$

dove W è la matrice dei pesi e b il bias del vettore. Gli autori riportano le correlazioni di Pearson ottenute da Bleurt su WMT17 e WMT18 per il task di traduzione automatica, sono le più alte rispetto a tutte le metriche considerate.

2.8 Nubia

Nubia è una metrica proposta da Hassan Kane [28], basata su reti neurali addestrate su migliaia di frasi per predire il giudizio umano. É organizzata in tre moduli principali: estrazione di feature, aggregazione e calibrazione.

2.8.1 Estrazione di feature

In questa fase vengono impiegati due diversi modelli basati sull'architettura Transformer e addestrati su task rilevanti NLP, come similarità semantica, inferenza logica e leggibilità di frase. Per il primo task Nubia utilizza RoBERTa-large, affinato sul dataset STS-B benchmark (8628 coppie) per predire in scala 0-5 la similarità tra frasi. Riportando i dati della leaderboard di STS-B[1], il modello ha ottenuto i seguenti risultati: 0,929 (Pearson) e 0,925 (Spearman). Secondo gli autori una frase che ottiene un alto punteggio di questo tipo avrà un'ottima correlazione semantica con la frase di riferimento. Il secondo insieme di feature da catturare riguarda la relazione logica tra candidato e riferimento. Non si parla più di correttezza grammaticale o semantica, ma di adeguatezza della frase, ovvero se i due testi trasmettono lo stesso messaggio. Anche in questo caso viene sfruttato RoBERTa-large addestrata sulla challenge MNLI[2], offerta da GLUE benchmark [46].

Il modello MNLI è addestrato per predire un numero discreto, 0 se le due frasi sono in contraddizione fra loro, 1 se la relazione logica non è determinata oppure neutra, 2 se c'è una corrispondenza logica fra le due frasi. In output Nubia restituisce punteggi basati su queste tre classi. La terza feature da catturare é l'accettabilità linguistica. Lo scopo di questa sfida è valutare se il testo generato sia grammaticalmente corretto e leggibile. É comune nei modelli generare frasi vicine semanticamente al riferimento, ma con errori grammaticali e sintassi errata. Queste due feature vengono carpite attraverso il punteggio "perplexity", calcolato dal modello GPT-2.

2.8.2 Aggregazione

Come si può evincere dalla figura 2.7, i tre task precedentemente descritti collassano in una rete neurale chiamata aggregatore, addestrata per approssimare una funzione che mappa le feature raccolte in precedenza in un punteggio di qualità, basato su quanto sono intercambiabili la frase di riferimento e quella candidata. Secondo gli autori quando un valutatore umano confronta una frase generata con i suoi riferimenti, giudica quanto i due testi esprimano lo stesso

[1] https://paperswithcode.com/sota/semantic-textual-similarity-on-sts-benchmark/
[2] https://paperswithcode.com/sota/natural-language-inference-on-multinli

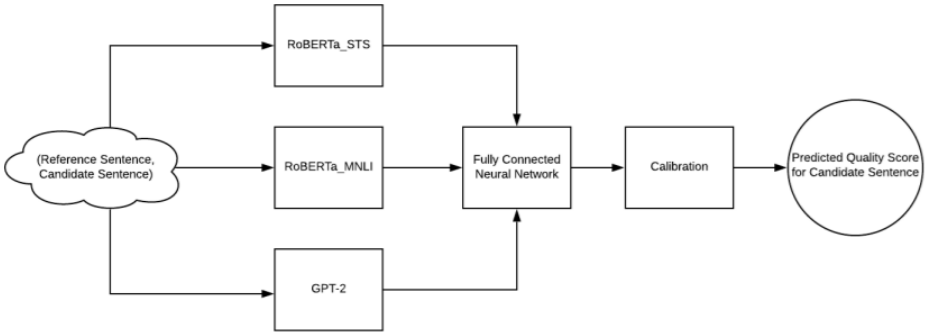

Figura 2.7: L'architettura di Nubia.
Fonte: [28]

significato e quanto il riferimento sia grammaticalmente valido. L'aggregatore è un modello neurale regressivo addestrato per predire il giudizio umano su coppie di frasi riferimento-candidato. In questo caso parliamo di regressione lineare e rete neurale feed-forward a sei (solo features) o otto strati(features e dimensioni delle due frasi da comparare) di input che corrispondono alle feature estratte, dieci strati nascosti e uno strato di output che corrisponde alla predizione. Gli autori hanno scoperto che aggiungere il numero di parole delle due frasi alla rete neurale, incrementa la correlazione con il giudizio umano.

2.8.3 Calibrazione

L'ultima fase della metrica è la calibrazione che si occupa di: (i) normalizzare i punteggi in modo tale che se una frase viene comparata con se stessa il punteggio di ritorno è 1, (ii) limitare l'output del regressore in modo che i punteggi stiano tra 0 e 1.

2.9 FactCC

FactCC [10] valuta la fattualità di un testo generato automaticamente. Un attento studio dei modelli di generazione di riassunti allo stato dell'arte hanno fornito approfondimenti sui diversi errori commessi nella valutazione della fattualità del testo generato. In primis l'approccio verificare la coerenza fattuale frase per frase del riassunto confrontandola con il testo originario è insufficiente. In alcuni casi è possibile che si debba valutare un contesto più ampio, con più di una frase, poiché potrebbero essere presenti delle ambiguità in una delle frasi confrontate. FactCC propone dunque un approccio documento-frase per la verifica della consistenza fattuale dove ogni frase viene verificata sull'intero

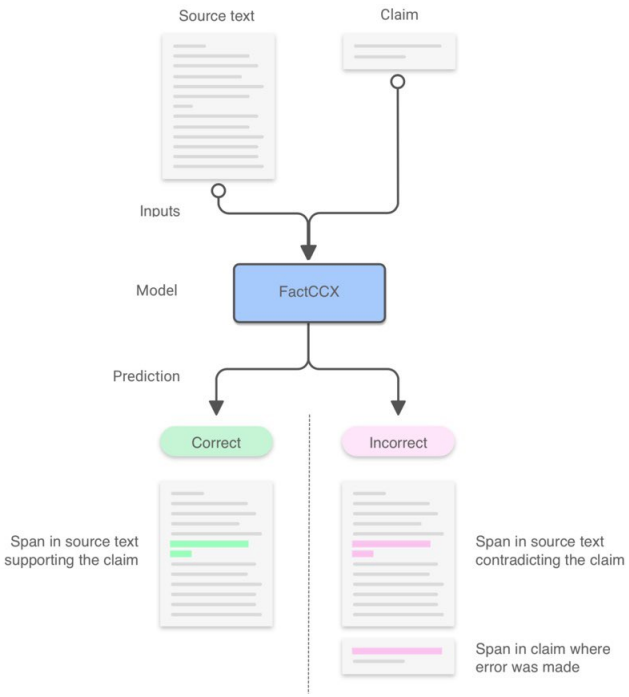

Figura 2.8: Valutazione della consistenza fattuale della frase su tutto il documento.

Fonte: [10]

corpo del documento. Il training set utilizzato per l'addestramento è artificiale, debolmente supervisionato, per costruirlo è stato preso come riferimento un dataset non etichettato e campionate un insieme di frasi denominate "claims". Queste claims sono state perturbate attraverso diverse tecniche, per poter creare esempi positivi e negativi per il modello. Le tecniche utilizzate nella trasformazione del testo generano sia variazioni semantiche che non, alcune di esse sono:

- `Parafrasi:` Il testo parafrasato viene generato attraverso la traduzione delle frasi testo originale, utilizzando sistemi neural machine translation.

- `Scambio di parole o numeri:` questa tecnica è utilizzata per addestrare il modello a riconoscere riassunti con parole o numeri sbagliati. Per generare una frase con entità di testo sbagliate, viene presa un'unità di testo dai documenti sorgente e sostituita nella claim in maniera casuale, ma assicurandosi di non selezionare due entità uguali nel testo. Queste entità estratte vengono suddivise in due gruppi: denominate(persone,luoghi, nomi di istituzioni) e numeriche (date o numeri).

- `Scambio di pronomi:` per verificare la consistenza fattuale, è necessario addestrare il modello a riconoscere pronomi errati nella frase. Questo metodo estrae tutti i pronomi vengono estratti dalle claim e scambiati rispettando i rispettivi gruppi, ad esempio un pronome possessivo puo essere scambiato con un altro possessivo. Queste modifiche alterano il significato semantico della frase.

- `Negazione di frasi:` per fornire al modello la capacità di gestire frasi negate utilizziamo questa tecnica. Le claim in un primo momento sono scansionate in cerca di verbi ausiliari, e casualmente vengono sostituite col rispettivo verbo negato. Ad esempio per negare un'affermazione viene aggiunto "not" o "n't", il contrario per le negazioni.

- `Inserimento di un elemento di rumore:` siccome si deve valutare la fattualità di un testo generato, ci si devono aspettare frasi con elementi di rumore.

Il modello viene addestrato sul datatset sintetico e utilizzato per la classificazione delle claim è BERT. Come si evince dalla Figura 2.8, BERT riceve in input i documenti e le claims e le classifica come consistenti/inconsistenti attraverso il classificatore a singolo strato basato sul token "CLS".

2.10 BARTScore

BARTscore la valutazione di testo generato come task di generazione del testo, come accennato in precedenza si avvale dell'idea che un testo di qualità può essere facilmente generato da un modello ad alte prestazioni come BART [5]. basandosi su un documento o un riassunto di riferimento. Utilizzando BART si sfrutta appieno l'architettura transformer, contrariamente a BERTScore che utilizza solamente il lato encoder. Bartscore sfrutta anche il decoder e calcola la probabilità di generare l'ipotesi passata partendo dal riferimento o dal documento sorgente.

$$\text{BARTSCORE} = \sum_{t=1}^{m} \omega_t \log p(\mathbf{y}_t | \mathbf{y}_{<t}, \mathbf{x}, \theta)$$

Figura 2.9: Formula per calcolare il punteggio di BARTScore.
Fonte: [12]

In Figura 2.9 abbiamo yt che è il token successivo da generare, x sono i token sorgenti. In questo caso ω sono i pesi idf dati ai diversi token, ma in

BARTScore sono tutti settati ad 1. Differenziando l'input dato alla metrica possiamo effettuare la valutazione del testo sfruttando quattro metodi diversi:

- `Faithfulness(s-> h):` dal documento sorgente all'ipotesi, si tratta quindi di calcolare la probabilità di generare il testo d'ipotesi conoscendo il testo sorgente e i relativi pesi dati a tale testo. É un metodo utile a capire inoltre la fluenza del testo e la coerenza rispetto al testo sorgente (Fedeltà).
- `Precision (r-> h):` dalla gold reference all'ipotesi, probabilità di ottenere il testo generato partendo dal testo prodotto da un umano.
- `Recall (h -> r):` dal testo generato dal modello al riferimento umano, questo metodo indica con quanta facilità una gold reference può essere generata dall'ipotesi, e utilizzabile per una valutazione a piramide nei task di summarization.
- `F-score (r<->h):` è la media tra precision e recall.

Gli autori di BARTScore hanno testato due varianti per la metrica, la prima consiste nel aggiungere frasi di prompt al riferimento o al testo generato, mentre la seconda consiste nel cambiare i pesi di riferimento per il testo, quindi riaddestrare il modello di generazione del testo BART per rendere il dominio di pre-train vicino allo specifico task da effettuare. Gli autori di BARTscore hanno implementato una versione della metrica che utilizza la tecnica di "promptine". Essa consiste nell'aggiungere brevi frasi di input per rendere i modelli preaddestrati più efficienti nell'esecuzione di un determinato task. La versione di BART che utilizza questa tecnica è detta BARTScore-PROMPT.

2.11 QuestEval

Tra i diversi task da valutare la summarization è uno dei più difficili. Per un dato documento da sintetizzare gli output possono essere innumerevoli. Inoltre, siccome un riassunto deve essere più breve rispetto al documento di partenza, è necessario verificare che contenga tutte le informazioni salienti. Le metriche classiche come Rouge, Bleu o Meteor non sono in grado di valutare aspetti come la consistenza fattuale. Si affianca dunque a FactCC un nuovo approccio di valutazione basato sul Question-Answering (QA), che verifica se le informazioni presenti nel documento sorgente sono rispettate dal riassunto. Thomas Scialom et al. [14] propongono QuestEval una metrica che non necessita obbligatoriamente dei riferimenti umani per valutare la fattualità e la rilevanza di un testo rispetto al documento sorgente. La metrica è composta da un componente **question generation** (QG) e **question answering** (QA).

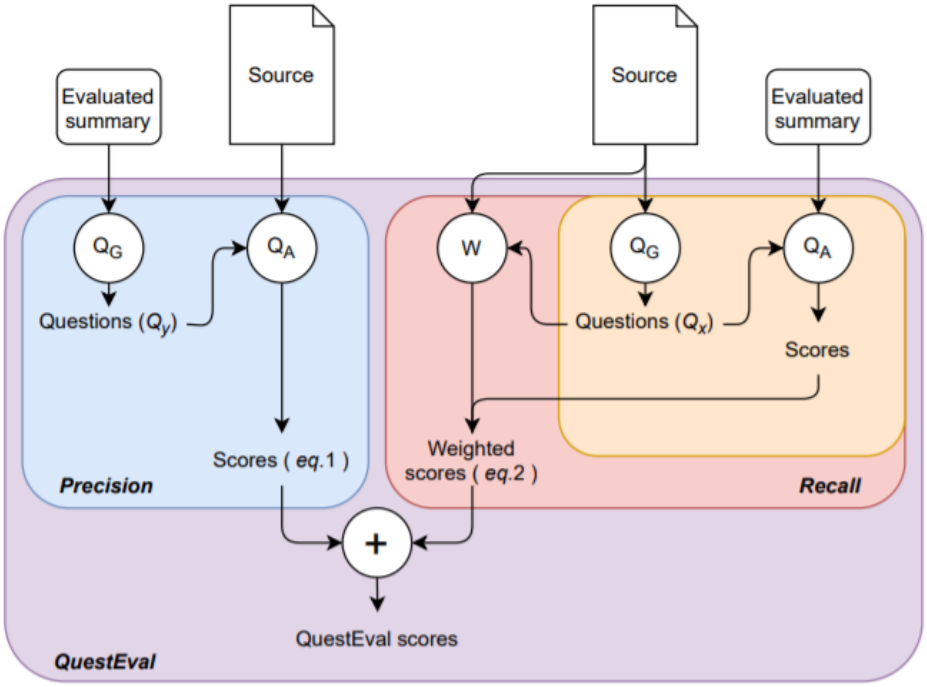

Figura 2.10: A sinistra abbiamo il calcolo della precisione quindi domande generate dal riassunto e risposte generate condizionando il modello sul documento sorgente. A destra viene calcolato il recall attraverso il sistema di query weghting.

Fonte: [14]

2.11.1 Componente QA

Il componente QA usato in questo caso è un modello T5 con il compito di estrarre risposte dal documento, dato il documento e una domanda a cui rispondere. Ci riferiremo a QA come $Q_A(r|T,q)$ probabilità di ottene una risposta r ad una domanda q nel testo T e $Q_A(T,q)$ la risposta ottima generata dal modello. Una volta valutato il riassunto non è detto che contenga effettivamente una risposta, è fondamentale che il modello QA sia in grado di predire quando una domanda è senza risposta. Il componente Q include perciò il *"unanswerable token"* (e).

2.11.2 Componente QG

Anche per il componente QG viene utilizzato un modello T5 addestrato per massimizzare la probabilità di ottenere domande quanto più vicine a quelle

umane, data la risposta e il documento sorgente. Vengono selezionati tutti i nomi ed entità nominate nel testo come risposte per condizionare il modello. Per ogni risposta viene generata una domanda attraverso l'algoritmo di "beam search". Le domande per cui il modello predice una risposta errata vengono eliminate. $Q_G(T)$ é il set di domande-risposte (q,r) per il testo T, $Q_G(T, q) = r$.

2.11.3 Precisione

$$Prec(D, S) = \frac{1}{|Q_G(S)|} \sum_{(q,r) \in Q_G(S)} F1(Q_A(D, q), r)$$

Se F1 restituisce 1 vi è un'esatta corrispondenza fra entrambe le risposte, 0 se non vi sono token in comune. Dove $Q_G(S)$ è il numero di coppie domande-risposte per la sequenza di token del riassunto generato. Tale risultato viene moltiplicato per la somma di tutti gli F1 calcolati sulla risposta predetta dal modello confrontata con quella ritenuta corretta.

2.11.4 Recall

$$Rec(D, S) = \frac{\sum_{(q,r) \in Q_G(D)} W(q, D)(1 - Q_A(e|S, q))}{\sum_{(q,r) \in Q_G(D)} W(q, D)}$$

$Q_G(D)$ è il set composto da tutte le coppie domande-risposte per il documento sorgente D, $W(q, D)$ sono i pesi delle domande. I modelli basati su question answering sono comunemente valutati utilizzando F1 score, misurando la sovrapposizione tra la risposta predetta e il riferimento. Tuttavia una risposta potrebbe essere corretta in tanti modi diversi, e F1 potrebbe ritornare 0 anche in questo caso. Per ovviare a questo problema si utilizza $1 - Q_A(e)$ invece di F1 Definendo il recall in questo modo è possibile misurare l'answerability di una domanda indipendentemente da come la risposta è espressa, non viene però tenuto conto di eventuali allucinazioni. Quando si valuta la consistenza fattuale di un documento, non è sufficiente che ad una domanda si possa associare una risposta dal documento sorgente. Le due risposte alle domande devono essere espresse con lo stesso significato per essere fattuali, usando l'answerability troviamo più "true positve", mentre per la precisione è importante rilevare i "true negative", per questo viene utilizzato F1 in quel caso.

2.11.5 Question weighter

Per la summarization Scialom introduce il query weighter,che è addestrato per distinguere le domande importanti da quelle superficiali. Per il calcolo del

punteggio finale sono stati unificati precisione e recall:

$$F_{SCORE} = 2\frac{Prec * Rec}{Prec + Rec}$$

2.12 FEQA

Durmus et. al [15] propongono un lavoro incentrato sul valutare come varia la fedeltà di un riassunto, all'aumentare dell'astrazione dal documento sorgente Gli autori giungono alla conclusione che all'aumentare dell'astrattività dei riassunti generati la loro fedeltà verso il documento sorgente diminuisce. Viene proposta poi una metrica, **FEQA** che similmente a QuestEval calcola la fattualità di un riassunto rispetto al suo documento sorgente. In in input accetta solo coppie <documento sorgente, ipotesi>, non lavora con i riferimenti al contrario di QuestEval. L'aspetto chiave della metrica è l'inserimento di maschere nel riassunto, le parole mascherate sono poi le risposte che il modello dovrà predire alle domande generate dall'ipotesi.

Per questa prima fase è stato utilizzato un modello BART addestrato sul dataset QA2D. Dall'altro lato invece si utilizza BERT affinato su Squad1.0/2.0 per rispondere alle domande generate dall'ipotesi, condizionando il modello sul documento sorgente. Il punteggio finale è un F1 tra le risposte.

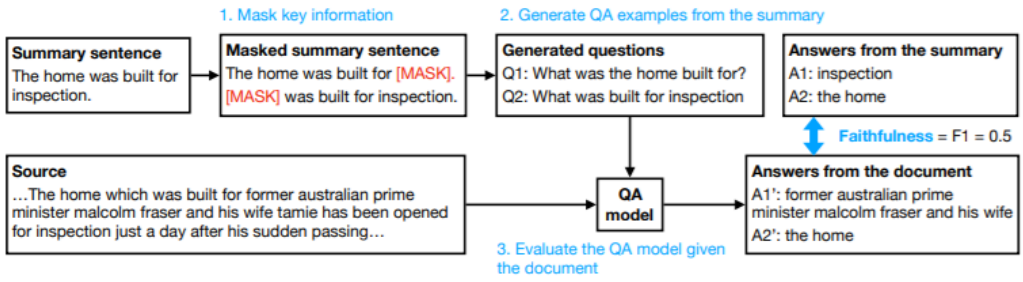

Figura 2.11: Passaggi basati su generazione di domande e risposte per calcolare il punteggio di FEQA.
Fonte: [15]

Capitolo 3

Blanche

In questo capitolo analizzo la libreria implementata da me e ed utilizzata per effettuare gli esperimenti descritti nel Capitolo 4.

3.1 Fase preliminare

La libreria si costituisce di tutte le seguenti metriche implementate in linguaggio Python:

- **Bleu** (Sezione 2.3).
- **Rouge** (Sezione 2.4).
- **Meteor** (Sezione 2.5).
- **BERTscore** (Sezione 2.6).
- **Bleurt** (Sezione 2.7).
- **Nubia** (Sezione 2.8).
- **BARTscore** (Sezione 2.10).
- **QuestEval** (Sezione 2.11).
- **FEQA** (Sezione 2.12).

In primo luogo mi sono appoggiato alla piattaforma Huggingface [47], la più grande community AI per il natural language processing, per studiare i sistemi di valutazione automatici più famosi e da lì partire per cercarne altri. Huggingface ha sviluppato una libreria chiamata "Datasets" che raccoglie le metriche più utilizzate, come ad esempio Rouge, Meteor e Bertscore. Per

caricarle in un proprio script python è necessario eseguire il comando "*datasets.load_metric(name_metric)* ", purtroppo però molte nuove metriche non sono presenti ed inoltre non vi é un'implementazione che gestisca le dipendenze. La mia idea è molto simile, ho voluto raccogliere sia quelle più famose e utilizzate sia quelle emergenti per poter cogliere diversi aspetti di valutazione, gestendo in automatico le dipendenze richieste. Negli ultimi anni, più precisamente dopo l'avvento dell'architettura Transformer e di BERT, sono nate metriche in grado di sfruttare approcci model-based per i diversi task di valutazione. Un esempio interessante è il question answering [48], un metodo promettente per rilevare la fattualità di un testo. In termini pratici, per le metriche emergenti ho direttamente reperito il codice sorgente dal relativo repository Github, in quanto l'unica implementazione disponibile è quella degli autori stessi. Diverso invece è stato per Bleu, Rouge e Meteor, che presentano diverse versioni sviluppate, mi sono quindi affidato ai riferimenti di Huggingface. L'obiettivo di questa libreria è togliere all'utente il compito di reperire le dipendenze python, e scaricare manualmente il modello pre-addestrato in caso servisse.

3.2 Struttura della libreria

La libreria è suddivisa in cartelle ognuna contenente l'implementazione della rispettiva metrica. Il modulo principale "blanche" si occupa di caricare in memoria il modello,ed eseguire lo score della metrica, mentre "utils" contiene funzioni utili a caricare i dati da file di testo.

3.2.1 Gestione delle dipendenze

La parte cruciale nello sviluppo della libreria è stata la gestione delle dipendenze, poichè rappresenta il problema più grande nell'utilizzo delle nuove metriche a cause dell'innumerevoli librerie da reperire per riuscire ad avviare la computazione (es. pythorch, transformer, tensorflow).

```python
def check_dependencies(r_file):
    print("Checking for dependencies...")
    import subprocess
    reqs = subprocess.check_output([sys.executable, '-m', 'pip', 'freeze'])
    installed_packages = [r.decode().split('==')[0] for r in reqs.split()]
    filed=open(r_file,"r")
    content=filed.read()
    dep_list=content.split("\n")
    filed.close()
    for dep in dep_list:
        if not dep in installed_packages:
            script="pip install " + dep
            os.system(script)
    print("Dependencies installed!")
```

Figura 3.1: Funzione per gestire le dipendenze di ogni metrica.
Fonte: https://github.com/marcoavagnano98/nlg-metrics

```python
def download_and_extract(url,storage_path,folder="./",unpackable=True,from_drive=False):
    """
    Parameters
    ---------i
    url: file to download
    storage_path: path to save compressed file
    folder: folder to extract tar/zip
    unpackable: if True extract file
    from_drive: file downloaded from drive
    """
    import tarfile
    filename, file_extension = os.path.splitext(storage_path)

    if from_drive:
        gdown.download(url,storage_path,quiet=False)
    else:
        wget.download(url,out=storage_path)
    if unpackable:
        print("Extracting compressed model...")
        if file_extension == '.zip':
            with zipfile.ZipFile(storage_path,'r') as zip_ref:
                zip_ref.extractall(folder)
        else:
            t_model=tarfile.open(storage_path)
            t_model.extractall(folder)
            t_model.close()
        os.remove(storage_path)
```

Figura 3.2: Funzione per scaricare ed estrarre i modelli.
Fonte: https://github.com/marcoavagnano98/nlg-metrics

Come si può vedere dalla Figura 3.1, la funzione “check_output()” lancia lo script “pip freeze” che restituisce una stringa contenente tutti i pacchetti installati nel sistema e le relative versioni. Successivamente vengono filtrati ed estratti i nomi delle singole librerie. Ogni metrica ha nella propria cartella un file denominato “requirements.txt” che elenca tutte le proprie dipendenze da installare e la versione minima o quella richiesta (es. $torch == 1.10.0$).

Ogni linea del file viene caricata in una lista e confrontata con i pacchetti presenti nel sistema, se la dipendenza non è presente viene installata. In fase di caricamento alcune metriche gestiscono internamente il download del loro modello addestrato, per altre invece è necessario implementare un metodo per farlo. La funzione mostrata in figura 3.2, permette di scaricare il file da Google Drive o da qualsiasi altra posizione in remoto. Per il primo tipo di download utilizzo la libreria "gdown", mentre per i restanti "wget". Una volta scaricato il modello è possibile specificare se estrarlo nel percorso passato alla funzione oppure salvarlo e basta, dipende da come gli autori hanno sviluppato la metrica.

3.2.2 Esecuzione delle metriche

La parte centrale della libreria si occupa dell'esecuzione di ogni singola metrica. Per ognuna di esse ho creato una funzione "run_{metric_name}", che riceve come argomenti la lista di riferimenti e le ipotesi da valutare. Non passando nulla i dati vengono caricati dalla cartella "data" attraverso il modulo "utils". Come vediamo dalla Figura 3.3, attraverso la funzione "load" vengono scaricati modello e dipendenze se necessario, inizializzata la classe della metrica ed eseguita. I punteggi ottenuti per ogni <riferimento, candidato> vengono trascritti in un file di testo all'interno dell'apposita cartella, la funzione restituisce la media degli score, mentre nel caso di Bleu il punteggio calcolato su tutto il corpus.

```
def run_questeval(references=[],candidates=[],sources=[],
                  f_preds="preds.txt",f_labels="labels.txt",task='text2text',do_weighter=False,no_cuda=True):
    """
    Parameters
    ----------
    references: list of references for each hypotesis
    candidates: list of claims to be evaluated
    sources: list of source documents for summarization task
    task: there is many type of task from questeval: summarization, text2text, data2text
    no_cuda: True use cpu, False use gpu, if available
    do_weighter: weight for summarization task

    """
    load("questeval")
    if not candidates and not references:
        candidates=utils.load_preds(f_preds)
        references=utils.load_preds(f_labels)
    from questeval.questeval_metric import QuestEval
    questeval = QuestEval(task=task, do_weighter=do_weighter,no_cuda=no_cuda)
    score = questeval.corpus_questeval(hypothesis=candidates,list_references=[references])
    return score
```

Figura 3.3: Esecuzione di QuestEval.

```python
def run_bleurt(references=[],candidates=[],f_preds="preds.txt",h_labels="labels.txt",metric="bleurt-20"):
    """
    Parameters
    ----------
    references: list of sources text for candidate
    candidates: list of texts to evaluate
    metric: specify which type of model is used for evaluation

    """
    load(metric)
    #checkpoint="bleurt/checkpoint/bleurt-base-128.zip"
    if not candidates:
        candidates=utils.load_preds(f_preds)
    if not references:
        references=utils.load_preds(h_labels)
    import bleurt.score as bs
    import numpy as np
    scorer = bs.BleurtScorer(BLEURT_MODEL_PATH)
    i=0
    all_scores=[]
    with open("bleurt/bart.txt", "a") as f:
        for i in range(0,len(references)):
            scores = scorer.score(references=[references[i]], candidates=[candidates[i]])
            all_scores.append(scores)
            f.write(str(scores))
            f.write("\n")
            assert type(scores) == list and len(scores) == 1
            print(i)
    print(np.mean(all_scores))
```

Figura 3.4: Esecuzione di Bleurt.

Capitolo 4

Esperimenti

In questo capitolo riporto i risultati di due diversi esperimenti effettuati su test-set ottenuti da due diversi modelli T5 e BART confrontati nel task di verbalizzazione di grafi evento. Le predizioni del modello T5 sono state computate attraverso l'algoritmo beam search con beam size fissato a 4, mentre per BART abbiamo solamente un'ipotesi generata per ogni input. Nel primo caso l'output è caratterizzato da quattro diverse frasi, ordinate per "log-likelihood" crescente, l'ultima frase dunque è la più probabile per il modello. Quindi viene fatta un'esplorazione in ampiezza del grafo e generati tutti i successori del nodo corrente, invece di scegliere volta per volta il token più probabile da generare, il modello seleziona le quattro frasi con log-likelihood complessivamente maggiore e le dispone in ordine crescente. In tutto abbiamo 13.752 predizioni per T5 e 3438 per BART. L'obiettivo di questi due esperimenti è dare una valutazione, basandoci su diversi aspetti che poi descriveremo, al testo generato dai due modelli utilizzando metriche automatiche per cercare quanto più possibile di riprodurre il punteggio che un umano darebbe a queste frasi e che al giorno d'oggi è considerato ancora come giudizio standard per i task NLP.

4.1 Task di verbalizzazione di grafi evento

I grafi sono strutture molto importante in campo NLP, possono rappresentare relazioni complesse fra un insieme di oggetti. Il task in particolare è un sotto-task del "data-to-text generation" che mira a creare testi fluenti che descrivono il grafo passato in input [49]. L'esperimento trattato in particolare riguarda la valutazione di testo generato da grafi evento nel dominio biomedico. Gli eventi biomedici sono interazioni complesse, strutturate, e semantiche coinvolgenti entità biomediche (es. proteine, geni, sintomi). Ogni entità coinvolta in un evento, vi partecipa con un preciso ruolo (es. tema, causa). Per "complesse" si intende la loro struttura n-aria e potenzialmente innestata, che li distingue dalle

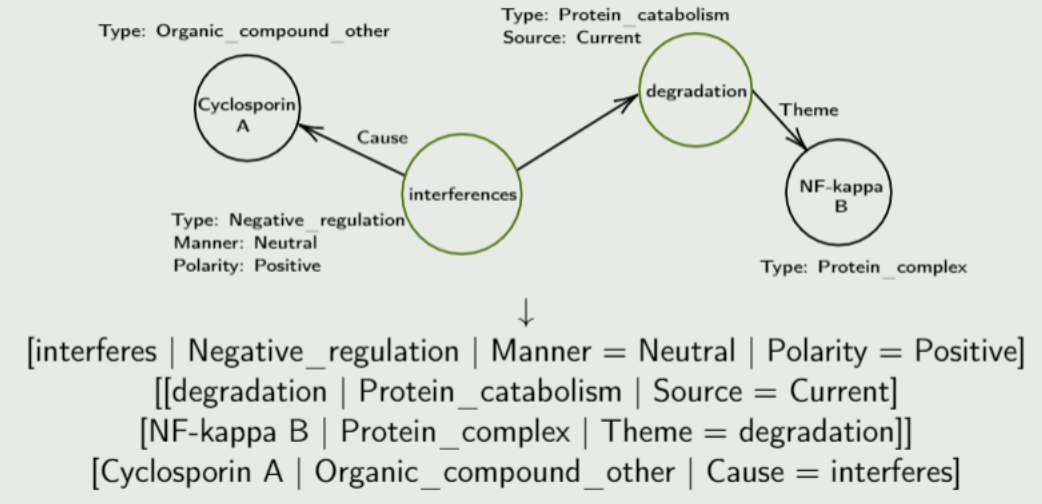

Figura 4.1: Input testuale ricavato dai 4 nodi per il modello
Fonte: slide tratta da "Verbalizzazione di eventi biomedici espressi nella letteratura scientifica: generazione controllata di linguaggio naturale da grafi di conoscenza mediante transformer text-to-text", Lorenzo Balzani

classiche relazioni binarie <soggetto, predicato, oggetto>. Infatti, i partecipanti di un evento possono essere a loro volta eventi, oltre che entità.

4.2 Modelli utilizzati

Il primo modello utilizzato è un T5-base pre-addestrato sul dataset C4 che ha subito un fine-tuning su un dataset specifico di verbalizzazione per 50 epoche con un batch-size di 16. Per T5-base si intende un modello con 220 milioni di parametri, riadattato attraverso la libreria FLAX per utilizzare un attention lineare anziché quadratica. Discorso leggermente differente per BART, è stata usata l'implementazione base presente su Huggingface. Entrambi i modelli sono stati addestrati nello stesso scenario, un dataset di 61869 esempi per 50 epoche.

4.3 Aspetti valutati dalle metriche

Le metriche utilizzate per gli esperimenti sono state ampiamente descritte nel Capitolo 2, andremo quindi solamente a spiegarne le prospettive di valutazione.

- `Bleu, Rouge, Meteor`: non riescono a carpire la contestualizzazione di una parola o il suo significato semantico, possono però dare una valutazione di quanto un testo risulti fluente ed adeguato rispetto al

riferimento, tralasciando le sinonimie fra parole che penalizzeranno di fatto il punteggio.

- `BERTscore`: effettua anch'esso un confronto fra due frasi, ma al contrario delle metriche precedenti crea dei vettori contestualizzati di parole, word embedding; riesce dunque, valutando gli stessi aspetti delle metriche precedenti, ad avere una correlazione più alta con il giudizio umano.

- `Bleurt`: gli organizzatori di WMT task, da cui BLEURT prende la maggior parte dei dati di addestramento, hanno chiesto ai propri annotatori: "Quanto l'output del sistema esprime adeguatamente il significato del riferimento?". In principio Bleurt potrebbe pertanto valutare l'adeguatezza. In pratica, gli autori di BLEURT hanno notato dalle risposte una forte correlazione con la fluenza del testo in inglese. Possiamo quindi considerare questi due aspetti.

- `Nubia`: al contrario delle precedenti, Nubia basandosi su modelli neurali preaddestrati come RoBERTa e GPT-2 riesce a catturare, oltre all'adeguatezza del testo, aspetti come variazione semantica ed errori grammaticali. Il punteggio è penalizzato in base alla percentuale di contraddizione, alla relazione semantica e all'irrilevanza fra le coppie di frasi.

- `QuestEval`: estrae le risposte attraverso il modello T5 dal riferimento e le confronta con quelle ottenute dal candidato utilizzando F1 e BERTscore. La metrica, considerando anche domande "unanswerable", rileva la presenza di allucinazioni, poiché parole inventate dal modello non sono in grado di rispondere alla domanda generata.

- `BARTscore`: come descritto nel Capitolo 2, ha delle prospettive di valutazione che variano in base all'input ricevuto. In questo caso, fornendo a BART i riferimenti "gold", possiamo avere informazioni riguardo la copertura semantica del testo, l'informatività e l'adeguatezza.

I punteggi delle metriche vanno tutti da 0 a 1, ad eccezione di BARTScore, il cui intervallo va da $-\infty$ a 0, più si avvicina a 0 più l'ipotesi è valida.

4.4 Esperimento 1

Per avere una copertura più ampia della valutazione sulle quattro frasi generate per ogni input, ho deciso di suddividere il primo esperimento in tre metodi.

- `Metodo 1`: il più semplice dei tre, attraverso ogni metrica ho calcolato il punteggio per ogni coppia <riferimento candidato> e riportato la media dei punteggi.
- `Metodo 2`: in questo caso ho scelto dalle frasi generate quella con log-likelihood maggiore, ottenendo un sottoinsieme quattro volte inferiore al principale.
- `Metodo 3`: per ottenere un confronto con il secondo metodo, quindi valutare se effettivamente la frase con log-likelihood maggiore fosse la migliore, ho selezionato il punteggio più alto tra le quattro frasi e riportato la media per ogni metrica.

Metrica	**Metodo 1**	**Metodo 2**	**Metodo 3**
Bleu	0,553	0,638	**0,696**
Rouge-1	0,616	0,688	**0,739**
Rouge-2	0,518	0,612	**0,667**
Rouge-L	0,583	0,547	**0,711**
Meteor	0,599	0,666	**0,721**
BERTScore	0,926	0,94	**0,951**
Bleurt-20	0,611	0,689	**0,735**
QuestEval	0,452	0.493	**0.529**
BARTScore	-2,362	-2,493	**-1,720**
Nubia	0,557	0,652	**0,731**

Tabella 4.1: Partendo da sinistra abbiamo la metrica utilizzata per valutare il testo e la media dei rispettivi punteggi ottenuti per tutti e 3 i metodi proposti.

4.4.1 Considerazioni

Complessivamente per quanto riguarda fluenza ed adeguatezza i punteggi sono molto simili, Bleurt nonostante utilizzi un modello pre-addestrato per predire il giudizio umano restituisce risultati praticamente identici a Rouge-1, e simili alle altre metriche che confrontano gli n-grammi. Questo può essere dovuto al fatto che in fase di pre-addestramento gli autori utilizzano dei "signals" costruiti su Bleu, Rouge e BERTscore, come descritto nel Capitolo 2 e che in questa particolare task in questo specifico dominio a fronte degli input ricevuti queste metriche colgono i medesimi errori. Per quanto riguarda BARTscore essendo una metrica basata sul task di generazione del testo proprio sfruttando BART, i risultati sono migliori nel secondo caso. Gli autori suggeriscono di sfruttare questa metrica effettuando un fine-tuning del proprio modello per

Metrica	Punteggi
Bleu	**0,559**
Rouge-1	**0,629**
Rouge-2	**0,535**
Rouge-L	**0,594**
Meteor	**0,586**
BARTScore	**-2.307**
Bleurt-20	**0,597**
QuestEval	**0,462**
BERTscore	**0,927**
Nubia	**0,562**

Tabella 4.2: Partendo da sinistra abbiamo la metrica utilizzata per valutare il testo e la media dei rispettivi punteggi ottenuti per tutti e 3 i metodi proposti.

avvicinare la valutazione al task e al dominio trattato ed ottenere risultati migliori. Per mancanza di tempo non ho potuto verificare quest'aspetto, ho testato invece la metrica utilizzando T5-large pre-addestrato su Pubmed (un dataset medico), ma ottenendo risultati largamente peggiori. Purtroppo non ho trovato un modello BART con fine-tuning su un dataset biomedico. *Analizzando i risultati ottenuti, valutando soprattutto metodo 2 e 3, possiamo affermare che non sempre generare il token successivo più probabile produce la frase più simile lessicalmente e/o semanticamente al proprio riferimento.*

4.5 Esperimento 2

Ogni frase generata in output è stata composta dalla verbalizzazione dei nodi del grafo. La quantità di nodi impiegati varia da 1 a più di 7, in questo secondo esperimento perciò ho voluto valutare quanto questo numero influisca sulla qualità del testo. Grazie al file di input dato al modello sono riuscito ad estrarre per ogni riga i nodi impiegatile ipotesi, e ricalcolato lo score aggregando i singoli punteggi per [1, 2, 3, 4, 5, 6, 7+] nodi.

Metrica	**1 n.**	**2 n.**	**3 n.**	**4 n.**	**5 n.**	**6 n.**	**7+ n.**
Bleu	0,184	0,389	0,521	0,628	0,738	0,779	**0,809**
Rouge-1	0,234	0,456	0,592	0,696	0,79	0,828	**0,852**
Rouge-2	0,104	0,331	0,486	0,606	0,726	0,773	**0,807**
Rouge-L	0.201	0.413	0,555	0,666	0,769	0,813	**0,841**
Bleurt-20	0,325	0,484	0,586	0,671	0,755	0,785	**0,802**
Meteor	0,233	0,433	0,571	0,675	0,786	0,821	**0,851**
BERTscore	0,857	0,896	0,921	0,941	0,959	0,967	**0,971**
QuestEval	0,271	0,366	0,441	0,505	0,545	0,559	**0,561**
Nubia	0,213	0,4	0,531	0,635	0,725	0,762	**0,791**
length	26,76	30,61	29,64	29,40	28,86	30,31	**31,29**
repetitiviness	28,64	34,16	33,45	32,91	32,82	34,41	**35,61**
abstractness	**0,914**	0.703	0.551	0.428	0,303	0,253	0,223

Tabella 4.3: Risultati ottenuti su ipotesi create con n numero di nodi dal modello T5.

Metrica	**1 n.**	**2 n.**	**3 n.**	**4 n.**	**5 n.**	**6 n.**	**7+ n.**
Bleu	0,184	0,386	0,541	0,639	0,761	0,759	**0,801**
Rouge-1	0,231	0,457	0,616	0,709	0,817	0,824	**0,857**
Rouge-2	0,079	0,331	0,516	0,626	0,764	0,776	**0,817**
Rouge-L	0,182	0,412	0,579	0,679	0,798	0,804	**0,841**
Meteor	0,201	0,416	0,569	0,665	0,786	0,788	**0,823**
BERTscore	0,856	0,896	0,925	0,942	0,962	0,963	**0,968**
Bleurt-20	0,293	0,463	0,583	0,665	0,755	0,761	**0,775**
QuestEval	0,266	0,368	0,456	0,519	0,561	0,561	**0,576**
Nubia	0,176	0,402	0,545	0,639	0,739	0,756	**0,779**
length	24,4	25,61	25,52	25,53	25,98	26,31	**27,21**
repetitiviness	27,16	27,79	28,15	27,92	28,62	29,34	**30,16**
abstractness	**0,931**	0,697	0,508	0,390	0,243	0,227	0,181

Tabella 4.4: Risultati ottenuti su ipotesi create con n numero di nodi dal modello BART.

4.5.1 Considerazioni

I risultati espressi per T5 in Tabella 4.3 e in Tabella 4.4 per BART ci dicono che mediamente la variazione del numero di nodi è direttamente proporzionale alla qualità dell'ipotesi generata. Notiamo che i punteggi aumentano all'aumentare del numero di nodi passati in input al modello. La ragione dietro a questi risultati potrebbe dipendere dal fatto che due nodi non danno sufficienti

informazioni contestuali per costruire una frase, quindi più informazioni si danno e più il modello riesce a generare frasi sintatticamente e semanticamente simili al riferimento umano. Non era scontato ottenere un risultato di questo tipo, ci si poteva aspettare difficoltà da parte dei modelli nell'incastrare così tante informazioni per creare una frase di senso compiuto. Visti i risultati ho deciso di inserire nuovi parametri di valutazione, la ripetitività e l'astrattività. La prima valuta quanti n-grammi si ripetono mediamente nelle frasi di ipotesi, il risultato è normalizzato per la lunghezza della frase, la seconda sicuramente più interessante misura quanti nuovi n-grammi sono presenti nell'ipotesi rispetto al riferimento. Come si può notare dalle Tabelle 4.3 e 4.4 la ripetitività si aggira mediamente tra i 3 e 4 n-grammi ripetuti a frase per entrambi i modelli, un ottimo risultato, mentre l'astrattività diminuisce al crescere del numero di nodi. *Quest'ultimo fenomeno è dato dal fatto che l'incremento dei nodi porta ad un avvicinamento lessicale dell'ipotesi al riferimento. Questa teoria è supportata anche dai risultati ottenuti dalle metriche basate su sovrapposizione di n-grammi.*

4.6 Confronto fra i modelli

In questa sezione riporto il confronto effettuato tra T5 e BART selezionando i punteggi per le metriche Bleurt e QuestEval. Per poterlo fare ho ricalcolato i punteggi di T5 come mostrato in Tabella 4.5 per le ipotesi con log-likelihood maggiore. Valutando i risultati senza l'output di beam search é ancora più chiara la superiorità di T5 su BART in questa specifica task. Si ipotizza che questo gap possa essere dovuto a una differenza architetturale e di pre-training. Nello specifico, T5-base segue un'architettura encoder-decoder completa con 220M di parametri, è addestrato su un corpus di grandi dimensioni paragonabile al risultato di un crawler su tutto il Web (C4), e adotta teacher forcing. Mentre, BART possiede 130M di parametri, è pre-addestrato su una quantità inferiore di documenti, e presenta semplici varianti rispetto alla controparte BERT (i.e., predizione di token mancanti anzichè mask-filling e ricostruzione dell'ordinamento completo delle frasi contenute all'interno di un documento anzichè next sentence prediction). Notiamo dalla Figura 4.2 che QuestEval a differenza di qualsiasi altra metrica analizzata, ha un leggero drop dei risultati per T5 quando passa da sei a sette o più nodi, mentre Bleurt é in continua crescita. Il motivo che sta alla base di questo comportamento é che il modello non in tutti i casi riesce a generare domande sensate, soprattutto in un task dove vengono forniti soltanto riferimenti. A mio avviso in un task di summarization dove in input si passa <documento, riferimento, ipotesi> avrebbe ottenuto risultati migliori. Queste affermazioni sono date dal fatto che rispetto alle altre

metriche QuestEval ha riportato risultati mediocri. Nel confronto fra i due modelli, la metrica rimane comunque coerente e mostra anche se più lievemente rispetto alle altre metriche la superiorità di T5 anche in termini di fattualità.

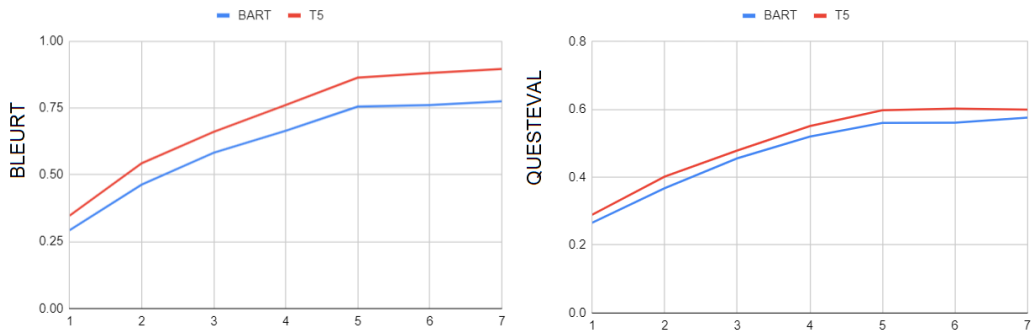

Figura 4.2: Confronto fra i punteggi ottenuti con Bleurt e QuestEval per i due modelli. Sull'asse X il numero di nodi in input, mentre sull'asse Y il punteggio della metrica.

Metrica	**1 n.**	**2 n.**	**3 n.**	**4 n.**	**5 n.**	**6 n.**	**7+ n.**
Rouge-1	0,233	0,518	0,661	0,773	0,884	0,905	**0,929**
Bleurt-20	0,348	0,544	0,661	0,761	0,864	0,881	**0,896**
QuestEval	0,289	0,402	0,479	0,551	0,598	**0,602**	0,599

Tabella 4.5: Risultati ottenuti su ipotesi create con n numero di nodi dal modello T5 considerando solo ipotesi con la più alta log-likelihood.

4.7 Tempi di valutazione

Le computazioni della maggior parte metriche sono state effettuate attraverso la piattaforma Google Colab. Per Bleurt ho utilizzato una GPU AMD r7 da 1 GB, in tutto per valutare 13752 (T5) + 4348 (BART) coppie di frasi, ho impiegato circa novanta ore. Da considerare che il modello BERT (BLEURT-20) é l'ultimo rilasciato da 579 milioni di parametri. Per le altre metriche model based i tempi si aggirano sulle 12 e 3 ore rispettivamente per T5 e BART.

Conclusioni

Questa tesi propone un'analisi dettagliata e aggiornata delle tecniche oggi disponibili per la valutazione di testo generato artificialmente. Nell'intento di offrire un contributo concreto alla comunità, si introduce una nuova libreria (BLANCHE) per la loro integrazione all'interno di un unico ambiente, risparmiando a qualsiasi possibile fruitore il compito di reperire le dipendenze necessarie. I risultati ottenuti sono stati utilizzati per contribuire alla fase sperimentale di un reale lavoro scientifico focalizzato sulla verbalizzazione di grafi, un trend di ricerca in forte crescita. Le predizioni rilasciate da T5 e BART si rivelano complessivamente comparabili, con punteggi F1 più alti nel primo caso. Per quanto concerne l'analisi rapportata al numero di partecipanti di un'interazione biomedica, si evince come le prestazioni tendano ad aumentare al crescere della grandezza del grafo, lasciando intendere come un maggior contesto offra al modello la possibilità di generare condizionatamente testo in modo più accurato In particolare, si è riscontrato un calo di astrattività della frase valutata rispetto al suo riferimento: al crescere del numero di nodi in input al modello l'ipotesi generata diventa via via meno astratta. Una metrica che ha destato particolare interesse durante la scrittura di questa tesi è QuestEval per la sua capacità di valutare efficacemente la fattualità di un testo in base al confronto di risposte – sia a livello lessicale che semantico – a domande automaticamente generate. Tuttavia, non è stato possibile sfruttarla al massimo del suo potenziale all'interno del caso di studio per via della mancanza di un documento corposo da cui estrarre correttamente le domande, in molti casi si è notata l'impossibilità del modello di generare risposte penalizzando di fatto il punteggio. Probabilmente QuestEval sarebbe più efficiente applicato ad un contesto di summarization. La presenza di implementazioni non mantenute o inutilizzabili a causa di errori in fase di compilazione, non ha permesso la sperimentazione e l'inclusione in BLANCHE di metriche quali QAGS e PEAK, rispettivamente basate su question answering e valutazione piramidale. La piattaforma HuggingFace ha agevolato la selezione di implementazioni standard per metriche altrimenti disponibili in molteplici varianti e determinanti un diverso risultato a fronte del medesimo input. Si segnala, inoltre, un tempo di computazione eccessivo per alcune metriche model-based, quali QuestEval e Bleurt. Tale aspetto, combinato al

quantitativo di risorse richieste – come numero di dipendenze e occupazione di memoria – è attualmente trascurato dagli autori, ma fondamentale nell'ottica della loro applicazione su vasta scala. Come questa tesi ha messo in luce, le metriche di valutazione automatica possono essere classificate in base al loro trade-off tra correlazione col giudizio umano ed efficienza. Proprio l'efficienza si prospetta come una delle principali ragioni che ancora limita il superamento di metodi semanticamente deboli come Rouge. Sviluppi futuri resi possibili dal lavoro in oggetto comprendono lo sviluppo di un framework per la comparazione massiva di metriche in termini di correlazione (es. Pearson e Kendall Tau) col giudizio umano, esaminando automaticamente un alto numero di metriche, dataset e task.

Ringraziamenti

Vorrei ringraziare il dott. Giacomo Frisoni che mi ha accompagnato in questo percorso di scrittura. Una passione e professionalità innata per il suo lavoro. Ringrazio la prof. Antonella Carbonaro che mi ha permesso di sviluppare questa tesi. Ringrazio la famiglia e gli amici che mi hanno sempre sostenuto nei momenti difficili.
In particolare, mia mamma Virginia, mio babbo Emilio, mia sorella Elena, mio fratello, Matteo, tutto il gruppo dei "periti": Viviano, Massimiliano, Riccardo, Nicholas, Elia, Matteo, Davide, Luca, Matteo, Lorenzo e i miei amici di corso: Simone e Thomas.

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. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[3] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[4] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

[5] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

[6] 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.

[7] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[8] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[9] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020.

[10] Wojciech Kryściński, Bryan McCann, Caiming Xiong, and Richard Socher. Evaluating the factual consistency of abstractive text summarization, 2019.

[11] Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. BLEURT: learning robust metrics for text generation. *CoRR*, abs/2004.04696, 2020.

[12] Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. *CoRR*, abs/2106.11520, 2021.

[13] Brian Thompson and Matt Post. Automatic machine translation evaluation in many languages via zero-shot paraphrasing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, November 2020. Association for Computational Linguistics.

[14] Thomas Scialom, Paul-Alexis Dray, Patrick Gallinari, Sylvain Lamprier, Benjamin Piwowarski, Jacopo Staiano, and Alex Wang. Questeval: Summarization asks for fact-based evaluation, 2021.

[15] Esin Durmus, He He, and Mona Diab. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070, Online, July 2020. Association for Computational Linguistics.

[16] Amirsina Torfi, Rouzbeh A. Shirvani, Yaser Keneshloo, Nader Tavaf, and Edward A. Fox. Natural language processing advancements by deep learning: A survey, 2021.

[17] Ehud Reiter and Anja Belz. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558, 2009.

[18] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[19] Felipe Almeida and Geraldo Xexéo. Word embeddings: A survey, 2019.

[20] Thomas Hofmann. Probabilistic latent semantic analysis, 2013.

[21] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[22] Y. Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. volume 3, pages 932–938, 01 2000.

[23] Kun Jing and Jungang Xu. A survey on neural network language models, 2019.

[24] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[26] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[27] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531, 2011.

[28] Hassan Kané, Muhammed Yusuf Kocyigit, Ali Abdalla, Pelkins Ajanoh, and Mohamed Coulibali. NUBIA: neural based interchangeability assessor for text generation. *CoRR*, abs/2004.14667, 2020.

[29] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[30] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[31] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[32] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *CoRR*, abs/2003.08271, 2020.

[33] Wilson L. Taylor. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433, 1953.

[34] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[35] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694, 2020.

[36] Vladimir Araujo, Andrés Carvallo, and Denis Parra. Adversarial evaluation of bert for biomedical named entity recognition. In *Proceedings of the The Fourth Widening Natural Language Processing Workshop*, pages 79–82, 2020.

[37] Jean-Baptiste Lamy, Alain Venot, and Catherine Duclos. Pymedtermino: An open-source generic api for advanced terminology services. *Studies in health technology and informatics*, 210:924–8, 05 2015.

[38] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders, 2019.

[39] Giuseppe Carenini and Jackie Chi Kit Cheung. Extractive vs. nlg-based abstractive summarization of evaluative text: The effect of corpus controversiality. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 33–41, 2008.

[40] Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. Understanding factuality in abstractive summarization with frank: A benchmark for factuality metrics. *arXiv preprint arXiv:2104.13346*, 2021.

[41] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. Evaluation of text generation: A survey. *CoRR*, abs/2006.14799, 2020.

[42] Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A Smith. All that's' human'is not gold: Evaluating human evaluation of generated text. *arXiv preprint arXiv:2107.00061*, 2021.

[43] Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822, Online, July 2020. Association for Computational Linguistics.

[44] Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *CoRR*, abs/1603.08023, 2016.

[45] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[46] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019.

[47] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[48] Lynette Hirschman and Robert Gaizauskas. Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300, 2001.

[49] Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. Investigating pretrained language models for graph-to-text generation, 2021.