Lezione 3: *LLM ed embedding*

Un veloce disclaimer: vorremmo cercare di raccontarvi qualcosa sulle tecniche utilizzate nei più recenti sviluppi nella ricerca sull'intelligenza artificiale, cercando magari di associarci qualche concetto matematico. Se a oggi cercate referenze su questi temi vi troverete spesso di fronte a due alternative: articoli estremamente tecnici, che richiedono molti prerequisiti per essere letti, oppure molto basilari, con il rischio che alla fine della lettura non si abbia imparato molto più di quello che si intuiva prima. Per mantenere un buon bilanciamento, vi racconteremo il significato di alcuni termini legati al mondo dell'intelligenza artificiale e vi lasceremo, verso la fine, intuire qualcosa sulla matematica che si nasconde dietro a queste prospettive. Vero, per farlo semplificheremo alcuni aspetti, ma questo non deve limitare la nostra curiosità, o la nostra soddisfazione per cercare di comprendere i misteri dietro a una delle più grandi rivoluzioni tecnologiche dei nostri giorni. Un'ultima cosa prima di iniziare: quando si trattano questi temi viene naturale dare un'idea di alcuni concetti tramite paragoni con situazioni e punti di vista "umani". Solitamente questi sono molto efficaci, ma non dobbiamo pensare a un paragone come a un punto di arrivo della nostra intuizione, ma come a un punto di partenza, per capire meglio un fenomeno complesso partendo da una situazione che ci fa sentire a nostro agio. Per ricordarci di questo isoleremo nel testo i paragoni con dei box. Buona lettura.

L'area di interesse sono i *Large Language Models*, spesso indicati con LLM. Questi costituiscono solo una piccola parte di tutta la tecnologia che va sotto il nome di Intelligenza Artificiale, ed è la parte che lavora con testi scritti.

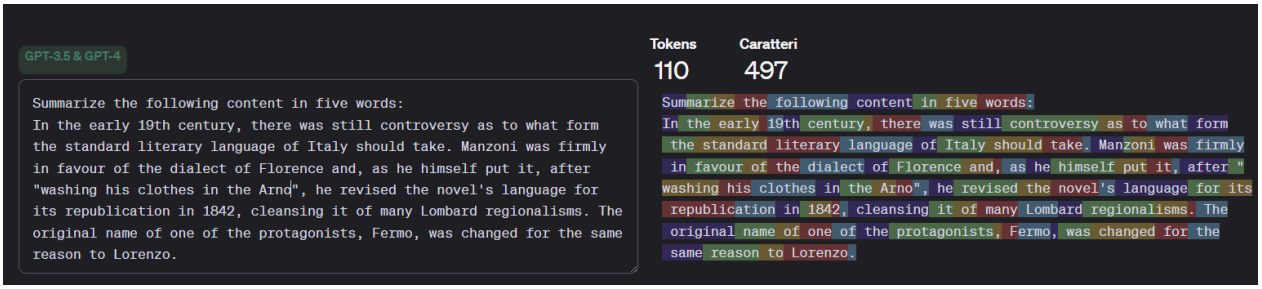
Uno sguardo dentro ChatGPT

Cerchiamo ora di capire cosa succede dentro un LLM (dentro ChatGPT, per esempio), partendo dal momento in cui l'utente inserisce un testo, e arrivando al momento in cui si legge una risposta.

Il primo step è una suddivisione del testo inserito in "token": gruppi di lettere contigue che possono spezzettare intere parole in parti. Per ChatGPT 3.5 i token sono circa un quarto rispetto al numero di caratteri inseriti in una domanda, ma questa è una scelta del modello. I token potrebbero essere singole lettere, oppure intere parole. Una stessa parola potrebbe, a seconda del modo in cui si trova in un testo, venire spezzettata in modi diversi. Questa operazione è la porta di ingresso in quello che succede negli LLM. Pensate che il "costo" di funzionamento di un LLM, viene spesso espresso tramite token. Al momento in cui scriviamo questo articolo, il costo di ChatGPT 4 è di 3 centesimi di dollaro ogni 1000 token in input e di 6 centesimi di dollaro ogni 1000 token in output.

La tokenization è un'operazione che facciamo anche noi: quando leggiamo un testo ci capita di soffermarci sulle singole parole per capire meglio il significato.

Leggendo "anticostituzionalmente", lo spezziamo in anti-costituzional-mente, dove i primi due token ci servono per capire il significato della parola e l'ultimo per capire che è un avverbio. Potremmo anche raggruppare più parole in un solo token, se in un testo ci fosse scritto "porta chiavi" potremmo leggerlo come porta-chiavi. Vi lasciamo immaginare come questa operazione già potrebbe dare via a problemi di interpretazioni (e lo fa: sia nel linguaggio umano, sia negli LLM).



Un testo inserito in un LLM e la sua suddivisione in token.

Ora che abbiamo suddiviso il testo di input in blocchi, dovremmo trasformarli in una forma facile da manipolare per il computer, e per la rete neurale che sta nel "cervello" del LLM. Lasceremo questa descrizione per la fine, perché è qui che vogliamo trovare un collegamento con la matematica. Immaginatevi solamente che ora il nostro testo verrà trasformato in sequenze di numeri, basandosi su parole, insiemi di parole, e singoli token. Queste sequenze numeriche prendono il nome di "embedding".

"Quel ramo del lago di Como, che volge a mezzogiorno" ⇒ [0.21, -0.13, 0.57, -0.42, ...]
"Quel ramo del lago di Como" ⇒[-0.35, 0.42, -0.28, 0.19, ...]
"Quel" ⇒[0.11, 0.27, -0.38, 0.15, ...]
"ramo" ⇒[0.52, 0.18, -0.21, -0.33, ...]
"del" ⇒[-0.25, 0.38, 0.67, -0.09, ...]
"lago" ⇒[0.33, -0.41, 0.23, -0.18, ...]
"di Como"⇒ [-0.12, 0.24, 0.31, 0.47, ...]
...
"mezzogiorno"⇒[0.43, -0.31, -0.19, 0.28, ...]
"mezzo"⇒ [0.23, 0.17, 0.47, -0.29, ...]
"giorno"⇒ [-0.55, 0.29, 0.15, -0.08, ...]

Un potenziale testo che viene trasformato in sequenze numeriche dal LLM. Potremmo avere sequenze per l'intero testo, o per parti di esso, sequenze per le singole parole e sequenze per i token.

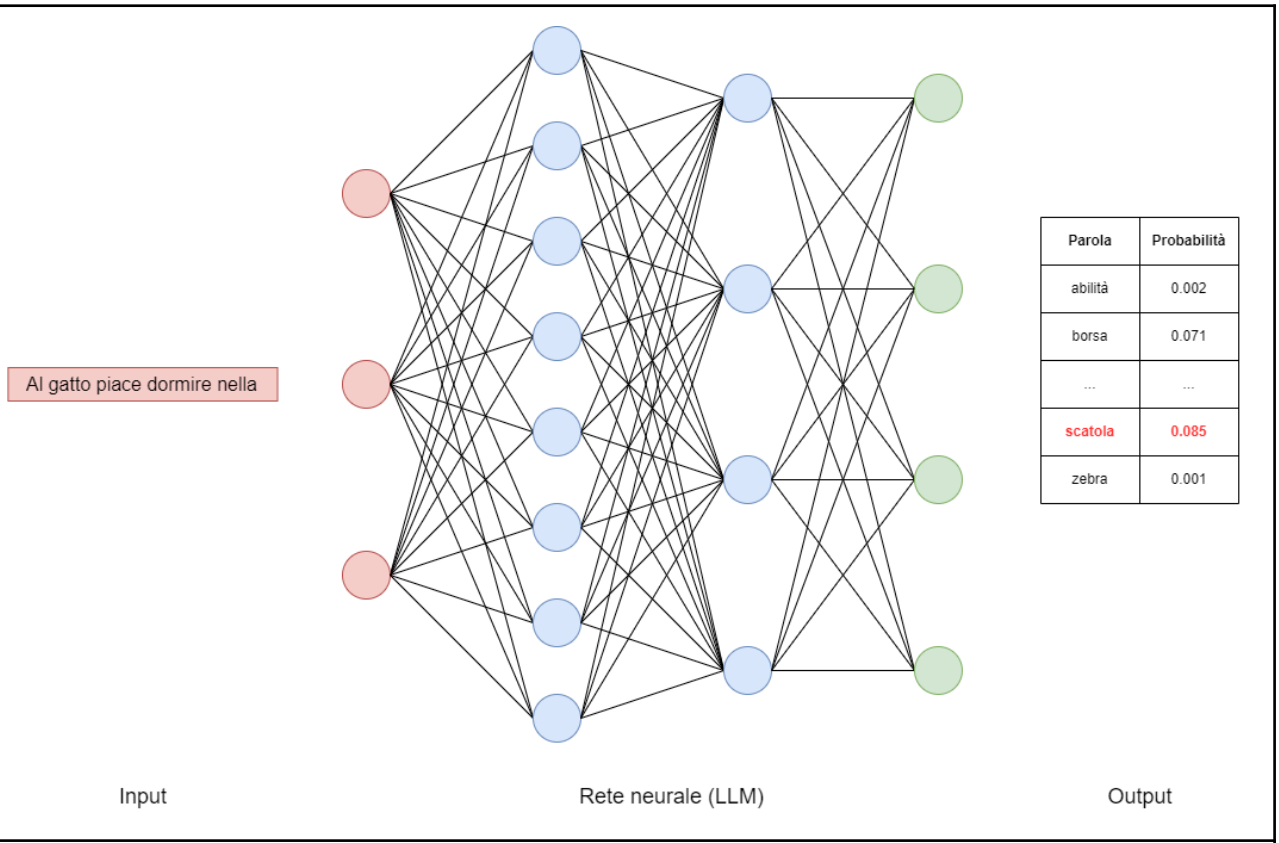
Ora che abbiamo a disposizione gli input, possiamo lavorare con una rete neurale che elabora questi input e li trasforma in altre sequenze di numeri che, in ultima analisi, produrranno la risposta del modello che stiamo considerando. Per darvi un'idea della dimensione di questa rete neurale, sicuramente dobbiamo immaginare una rete con diversi strati di neuroni.

Ogni tanto si usa il termine “rete con diversi strati di profondità”, da cui il nome “Deep learning”. ChatGPT 3 ha 96 layer e 175 miliardi di neuroni (o meglio, di parametri). Come confronto, il cervello umano ha circa 100 miliardi di neuroni. Che cosa può fare questa rete neurale?
Le reti neurali sono estremamente potenti per cercare di riconoscere strutture nei dati di input. Immaginate per esempio di partire da una frase e cercare di riconoscere se questa frase ha un significato “positivo” o “negativo”. Notate che le risposte dipendono dai dati con cui la rete è stata addestrata. "La passeggiata è stata piacevole" potrebbe quindi dare risultati di positività diversi da quelli che ci aspettiamo. Inoltre, l’interpretazione di una frase dipende dal contesto, e nel nostro linguaggio potrebbe essere presente dell’ironia, che potrebbe ribaltare il livello di positività e negatività della frase. D’altronde anche la letteratura è piena di frasi che possono essere intese in modo diametralmente opposto, pensiamo alla famoso aneddoto raccontato dal monaco Alberico, in cui un oracolo pronuncia la frase “*Ibis redibis non morieris in bello*”, che può essere letta come *Ibis, redibis, non morieris in bello (*“Andrai, ritornerai e non morirai in guerra”) oppure come *Ibis, redibis non, morieris in bello* (“Andrai, non ritornerai e morirai in guerra”).

"Il pranzo era accettabile." ⇒ Positivo = 40%, Negativo = 60% "La serata è stata discreta." ⇒ Positivo = 30%, Negativo = 70% "La passeggiata è stata piacevole." ⇒ Positivo = 65%, Negativo = 35% "La vista era mozzafiato." ⇒ Positivo = 90%, Negativo = 10%
Diverse frasi con diversi livelli di positività associati. Potrebbero esserci discordanze tra quello che è scritto e il nostro senso comune: questo perché la classificazione dell’IA dipende dai dati con cui è stata addestrata.

Queste possibilità non si fermano ovviamente al testo: si possono studiare suoni, immagini, filmati, sempre sfruttando la filosofia di spezzarli in parti più piccole, convertire in dati numerici e poi usare una rete neurale per riconoscere potenziali strutture che questi dati possono avere.

Abbiamo gli input, abbiamo la potenzialità di studiare e classificare gli input. Come facciamo a elaborare l’output? Come fa ChatGPT a scrivere? È il momento di parlare dei modelli di linguaggio (le ultime due lettere in LLM). Il problema che cerchiamo di risolvere è quello di “prevedere quale sarà la parola successiva in una sequenza di parole che vogliamo costituisca una frase. Osservate come questo problema si riesca a formulare nel linguaggio delle reti neurali. Abbiamo un insieme di input dato dalle prime parole della frase (ma anche da tutti gli altri dati che arrivavano dall’input dell’utente). Poi abbiamo un insieme possibile di output, dato per esempio dall’elenco di tutte le parole possibili con cui si potrebbe completare la frase. La rete neurale, per ciascun output, sceglierà la probabilità per cui quella parola potrebbe completare la frase.



Una schematizzazione di come può essere individuata la parola successiva a partire da un input

Una volta che il nostro modello ha scelto la parola successiva, si ricomincia, riprendendo come input la stessa sequenza di parole di prima a cui è stata aggiunta una parola in più, e calcolando il più probabile termine successivo. Una volta terminato questo processo abbiamo la frase finale, che costituirà la risposta del LLM alla nostra domanda, dopo essere stata opportunamente riconvertita da sequenze numeriche a linguaggio umano.

Se abbiamo provato a mandare un messaggio vocale tramite smartphone senza pensare in anticipo a cosa dire probabilmente abbiamo sperimentato un'esperienza simile. Nella mente abbiamo i concetti che vogliamo esprimere e abbiamo le parole che abbiamo già iniziato a pronunciare; man mano costruiamo la frase finale basandoci sia sull'obiettivo sia su cioè che abbiamo già detto.

Tutto sommato, dalla descrizione fatta fino a ora sembra che gli LLM non siano così tanto più complicati del completamento automatico di uno smartphone. Siamo sicuri che già molte persone che leggono hanno provato a cliccare più volte sulla parola automatica da completare, fino a costruire delle frasi di senso compiuto, anche se non particolarmente profonde. La grossa differenza sta sia nella grande complessità degli LLM (la prima "L" sta per "Large"), sia nel modo in cui queste reti neurali vengono addestrate. Questo ci permette di parlare di un altro acronimo, che è proprio l'acronimo GPT di "chat GPT". Conosciamo già il significato della G: sta per "Generative", cioè che *generano* la risposta prevedendo la parola che verrà subito dopo in una frase e completandola man mano. Non entreremo troppo nel dettaglio della T, che sta per

“Transformer”: questa parola descrive più nel dettaglio come è costruita la rete neurale utilizzata. La P, infine, significa “Pre-training”, che ci suggerisce proprio la maniera in cui questo modello viene addestrato.

Parlare di “pre-training” lascia intendere che ci siano più momenti nell’apprendimento, ed è proprio così. L’addestramento avviene in tre momenti: il primo (pre-training) è quello in cui la rete viene messa a confronto con moltissimi dati (milioni e milioni di frasi prese da internet e da libri) e la rete impara a formulare frasi grammaticalmente sensate e memorizza relazioni tra molti concetti. In questa fase, il modello non ha uno scopo: diventa solo molto abile a riconoscere strutture nei dati. Chiedendo a un LLM che ha svolto solo la fase di pre-training “Come ti chiami?” potrebbe rispondere con la frase “Quanti anni hai”, riconoscendo il fatto che in molti contesti incontrati nel pre-training la seconda domanda segue la prima.

Possiamo immaginare un LLM in questa fase come uno studioso che affronta una lingua sconosciuta, con caratteri e glifi sconosciuti. Analizzando testi in questa lingua riconosce alcune regole grammaticali (un glifo viene sempre dopo un altro), ma non sa davvero scrivere una frase precisa in quella lingua.

Per “allineare” una LLM con le aspettative che vogliamo dal modello si passa alla fase successiva, che è la fase di “fine-tuning”. Ora la “qualità” del materiale di apprendimento è migliore: non sono solo testi presi a caso, ma domande e risposte, istruzioni e contesto. In questo modo il modello, che ha già *imparato* a essere un algoritmo per completare testi, diventa più preciso, rispondendo in modo mirato seguendo le istruzioni.

Se vogliamo diventare provetti cuochi, la fase di pre-training potrebbe essere quella in cui si apprendono vaste nozioni di come sono chiamati gli oggetti in una cucina, dei diversi tipi di vegetali, e delle leggi fisiche che permettono la cottura. La fase di fine-tuning è quella di apprendere le operazioni necessarie seguendo le istruzioni che sono state compilate da un vero chef.
Ci sarebbe anche una terza fase, chiamata “RLHF”, che sta per apprendimento rinforzato da feedback umani. Lo scopo è simile a quello del fine-tuning, ma in questa fase siamo ancora più concentrati a fare sì che le risposte del nostro modello riflettano le aspettative umane.

Non è finita qui: ci sarebbe ancora molto da dire per comprendere meglio come funzionano questi LLM. Perciò, prima di passare alla matematica, vi lasciamo con delle domande, che possano servire come spunto per un approfondimento ulteriore, o anche solo per la consapevolezza che, anche in questo caso, c’è ancora molto da scoprire.

- Che cos’è la “temperatura” di una LLM, e come è legata alla tipologia di risposte che si ottengono?
- Come fa un LLM a riassumere un testo, quando il testo non è mai apparso nei dati di training?
- Come mai le LLM talvolta rispondono in modo sbagliato, oppure inventando risposte plausibili?

Spazi vettoriali ed Embedding: la matematica nascosta

Vogliamo ora tornare per un attimo al concetto che abbiamo chiamato “embedding”. Quella fase in cui frasi, parole, token vengono convertiti in dati numerici per poi essere interpretati dalla rete neurale. Abbiamo affermato che un testo viene convertito in una sequenza di numeri.

“Quel” ⇒[0.11, 0.27, -0.38, 0.15, ...]

La lunghezza di queste sequenze può variare a seconda del modello utilizzato. Usando l’algoritmo di embedding chiamato “text-embedding-ada-002” la lunghezza è per esempio di 1536. Da ora in avanti useremo sequenze lunghe 4, per semplicità. Queste sequenze numeriche hanno molte proprietà, ma in particolare possono essere sommate tra loro e riscalate:

[0.2, 0.11, 0.67, 0.4] + [1.1, 0.7, 0.01, 0.5] = [1.3, 0.81, 0.68, 0.9]

2·[0.1, 0.6, 0.91, 1.1] = [0.2, 1.2, 1.82, 2.2]

Queste due operazioni “regalano” agli embedding una struttura che in matematica è chiamata spazio vettoriale, una nozione che viene a volte trattata anche a scuola e in quelli che sono i corsi universitari di algebra lineare. La possibilità di sommare e di riscalare le sequenze apre le porte a moltissimi metodi di individuare le strutture tra i dati e di trovare i modi per rappresentarli (e per rappresentarli “velocemente” con algoritmi che permettano agli LLM di rispondere entro pochi secondi e non entro poche ore).

Una possibile struttura aggiuntiva è il concetto di “distanza”, che ci permette per esempio di affermare che i vettori (o gli “embedding”) [1, 0.4, 0.9, 2.1] e [1.01, 0.41, 0.89, 2.11] sono “vicini”. Ci sono molti modi possibili di calcolare la distanza tra due vettori, e chi fa matematica si trova spesso a studiarne le proprietà. Una volta trovato il modo di calcolare la distanza, si può trovare l’insieme di tutti gli embedding “vicini” (cioè che distano meno di una quantità fissata) a [0.2, 0.11, 0.67, 0.4], si può parlare di angoli e di perpendicolarità tra vettori e anche studiare come cambia questa relazione di vicinanza quando dei vettori vengono sommati o riscalati.

Il problema di realizzare embedding per un LLM gioca sul fatto di trovare un modo di associare a un testo dei vettori, in modo che la “distanza matematica” che si può costruire rispecchi una qualche forma di distanza semantica tra le parole o le frasi. Intuitivamente, possiamo richiedere che parole che sono sinonimi vengano mandate in vettori “vicini”, ma non solo. Vorremmo magari che il testo “Quel ramo del lago di Como” venga mandato in un vettore vicino al testo “I promessi sposi”, e al testo “Alessandro Manzoni”. Il fatto che la lunghezza dei vettori utilizzati (che viene chiamata “dimensione” dello spazio vettoriale in cui si trovano) sia molto alta, garantisce di avere tante possibilità per avvicinare i vettori rispecchiando significati semantici diversi.

In conclusione, chiedendo a un LLM una domanda tipo “Quali località mi consiglieresti di visitare in Veneto?”, non solo la domanda e i token associati, saranno convertiti in vettori, ma la rete neurale verrà informata anche di possibili vettori “vicini”, per esempio alla parola “Veneto”, fornendo un ampio contesto con cui costruire la risposta.

Crediamo che osservare come una parte di matematica molto astratta quale l'algebra lineare si possa vedere utilizzata nei più recenti sviluppi dell'intelligenza artificiale non sia solo una motivazione per approfondire la disciplina, ma anche una conferma del suo impatto tangibile nel mondo reale.

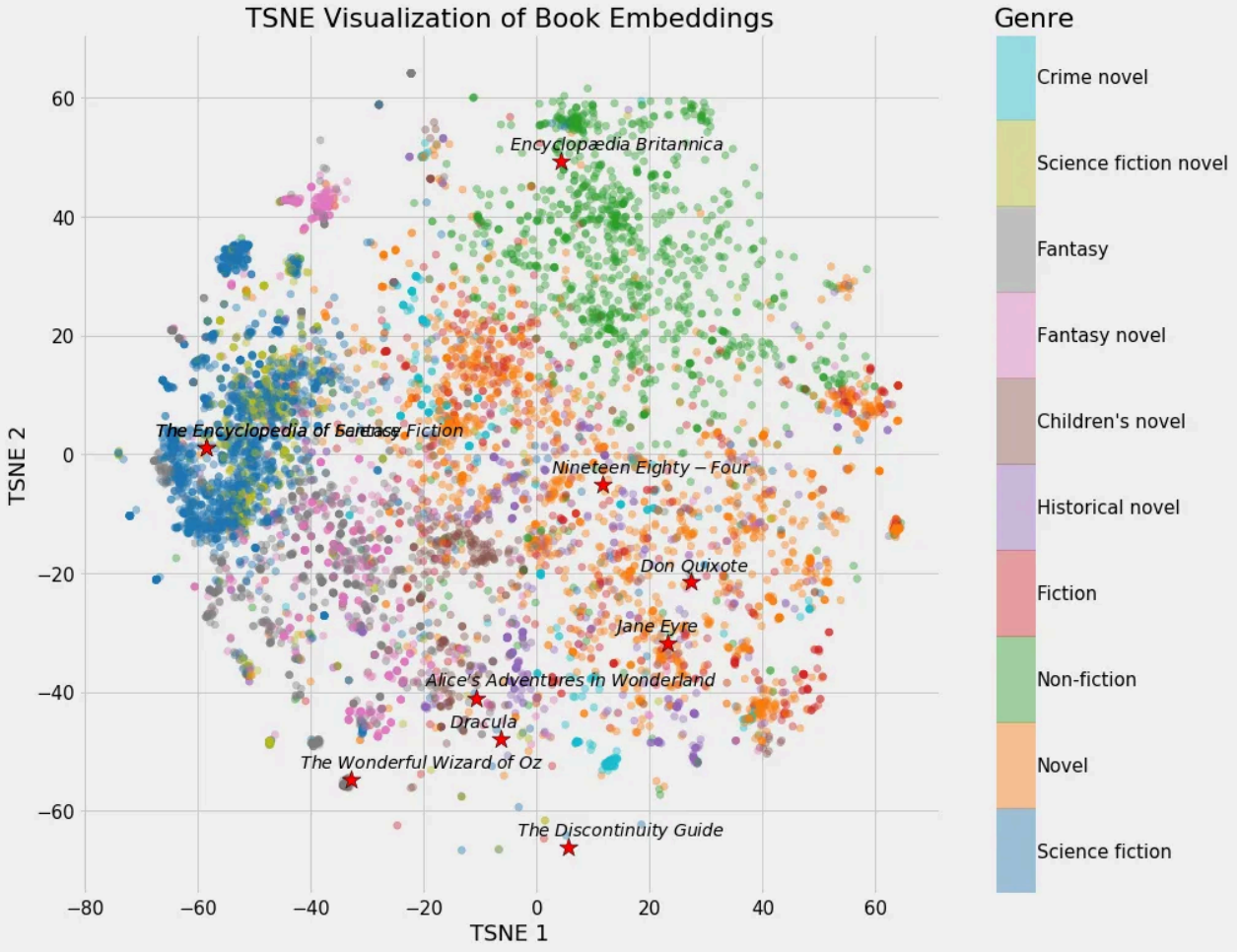


Immagine tratta dal progetto di "Book reccommendation System", che realizza un embedding di libri e visualizza qui la loro "vicinanza".

In classe: alla ricerca del colmo perfetto.

Proponete ai vostri studenti la seguente attività, da fare a gruppi. Assegnate a ciascun gruppo una domanda del tipo “Qual è il colmo per un pizzaiolo?” e chiedete loro di provare a rispondere nel “miglior” modo possibile seguendo i seguenti passaggi.

- Elencate alcune informazioni “di primo livello” che vi vengono in mente pensando al “pizzaiolo”. Per esempio “Impasta”, “Margherita è il nome della pizza più famosa”,... State selezionando informazioni “vicine” al concetto di “pizzaiolo”.
- Selezionate alcune di queste informazioni di primo livello e fate la stessa cosa a partire da queste (e senza tenere traccia del fatto che queste erano legate al concetto di “pizzaiolo”). Per esempio, “Margherita è un nome proprio”...
- Create un colmo - più o meno riuscito... - selezionando una o più informazioni di secondo livello. Per esempio “Avere una figlia che si chiama Margherita e che ogni quattro stagioni fa la capricciosa”.

Bibliografia

- https://www.tokenex.com/blog/what-is-tokenization/
- https://platform.openai.com/tokenizer
- https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f
- https://www.leewayhertz.com/what-is-embedding/
- https://github.com/WillKoehrsen/wikipedia-data-science/blob/master/notebooks/Book%20Recommendation%20System.ipynb
- https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526