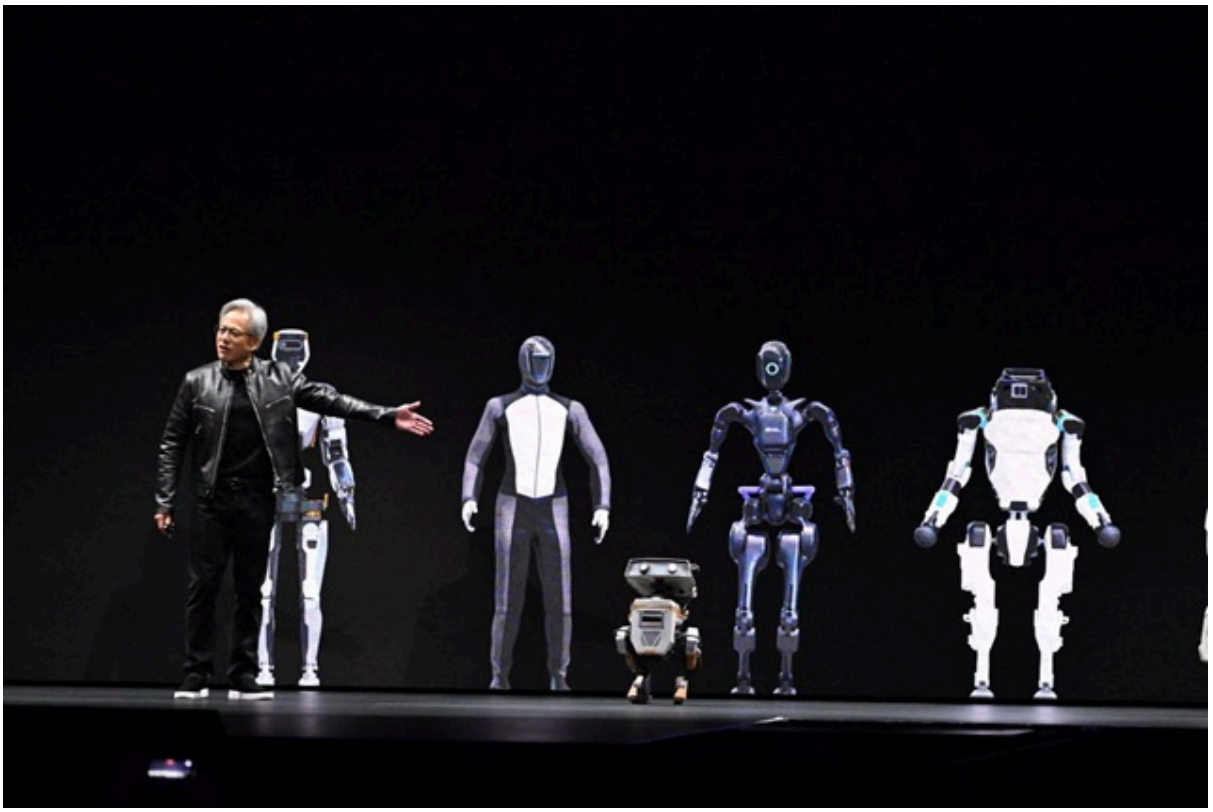
Università degli Studi di Padova
Corso di Ingegneria del Software 2024/2025



Artificial QI

Valutare i risultati di sistemi che utilizzano i Large Language Models.

ZUCCHETTI
IL SOFTWARE CHE CREA SUCCESSO

Oggetto dell'appalto

Le nuove possibilità offerte dai metodi dell' Intelligenza Artificiale stanno raggiungendo le applicazioni destinate agli utenti finali. Si vedono sempre più programmi che offrono accesso a sistemi basati su Large Language Models per aiutare l'utilizzatore in compiti che sono universalmente riconosciuti come "intelligenti".
Per una introduzione agli LLM segnaliamo questa survey: https://arxiv.org/pdf/2402.06196

Se questo è il comportamento visibile dall'esterno, all'interno della procedura informatica gli sviluppatori hanno costruito un sistema che deve per forza utilizzare delle componenti che non sono come i tradizionali sistemi di programmazione: certi, ripetibili, algoritmici.

Da una attenta osservazione possiamo in effetti notare come l'ultima ondata di Intelligenza Artificiale sia fortemente basata sulle capacità dei Large Language Models di comprendere il linguaggio naturale. Ma questa abilità è dovuta a "capacità emergenti", non a deliberate scelte ingegneristiche. All'aumentare della dimensione delle reti neurali utilizzate, verso i 70 Miliardi di neuroni (indicati usualmente come di parametri), sono inaspettatamente emerse tre capacità: la capacità di imparare dal contesto della domanda posta al sistema, la capacità di seguire le istruzioni fornite e la capacità di eseguire il compito richiesto passo a passo.

Per la loro natura statistica, queste grandi reti neurali contenute negli LLM danno risposte che non sono sempre uguali e ben determinate. L'algoritmo di calcolo è preciso e definito, ma l'introduzione di alcune variazioni casuali rende la risposta sempre diversa e molto difficile da prevedere.

Molti esperti parlano di "scoperta" anziché di "invenzione": nel secondo caso il risultato è ottenuto come logica conseguenza dei metodi utilizzati, nel primo ci si trova di fronte a qualcosa di inaspettato ma non per questo meno utile.

Forse la scoperta più importante degli ultimi mille anni è stata la scoperta dell'America.
Guardiamo cosa succede di fronte ad un evento così importante ma non costruito, intuendone i risultati: Cristoforo Colombo è morto convinto di essere arrivato in India, per 100 anni è stato cercato l'oro in quelle terre al punto che ne è crollato il valore, poi sono partite bolle speculative come la "Compagnia dei Mari del Sud".
Eppure oggi il nostro mondo è caratterizzato da mais, patate, pomodori … tutte cose che non sono state considerate per moltissimo tempo, essendo tutti quanti accecati dall' oro e dalle pietre preziose.

Trovandoci di fronte a sistemi statistici e non deterministici, è molto difficile per gli sviluppatori verificare quali siano le conseguenze sul risultato finale di ogni singola scelta che viene applicata nel processo di sviluppo, quando le procedure utilizzano queste tecniche.

Vediamo con degli esempi: sto sviluppando un sistema che raccoglie tutte le leggi del diritto fallimentare e data una domanda dell'utente devo estrarre tutti i documenti rilevanti, come cambiano le risposte cambiando il modello LLM utilizzato?
E' strettamente necessario "ChatGPT-4-turbo", modello più potente attualmente offerto da OpenAI e dal numero di parametri non noto ma si favoleggia sia superiore al trilione, o "Mistral-7B" di Mistral AI con soli 7 miliardi di parametri fornisce risultati accettabili?
Le capacità di ragionamento richieste dal prodotto in sviluppo sono presenti solo in modelli superiori a 70B (miliardi di parametri) o bastano modelli più contenuti come i 7B che si possono usare in locale con una GPU dotata di almeno 12 GigaByte di RAM e dal costo relativamente basso?
Come cambia la risposta del sistema dopo aver effettuato una attività di "fine-tuning" di un modello, adattandolo ad uno specifico contesto applicativo?
Come influisce nel sistema di preparazione del prompt, la "Retrieval Augmented Generation", la spezzatura e l'indicizzazione della base di conoscenza? Per una introduzione alla RAG rimandiamo a: https://arxiv.org/pdf/2312.10997
Quanto deve essere grande e precisa la base di conoscenza per poter dare all'utente risposte utili?
Quante volte il sistema produce risposte senza senso (allucinazioni) pur partendo da dati corretti?
Riesco ad ottenere risultati decenti anche se per ragioni di privacy non faccio mai uscire i dati dalla macchina?

A tutte queste domande non si può dare risposta per "calcolo": non siamo nei casi in cui "2+2 fa 4"; l'indeterminazione e la natura statistica dei Large Language Models costringe ad una verifica a posteriori delle prestazioni del sistema.

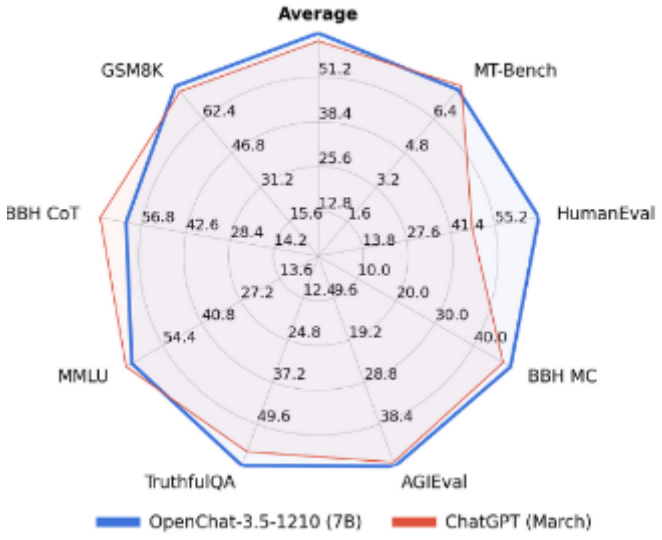


Grafico dalla pagina di Open Chat 3.5 su Hugging Face con dati di fine 2023
https://huggingface.co/openchat/openchat-3.5-1210

Il problema è rilevante, al punto che per valutare le capacità degli LLM sono stati inventati dei test di varia natura in cui i modelli danno risposte a varie domande per capire come performano in un dato compito.
Nella figura, ad esempio, si vede il confronto tra OpenChat 3.5 e ChatGPT nei test MT-Bench, HumanEval, BBH MC, AgiEval, TruthfulQA, MMLU, BBH CoT e GSM8K.
Questi test sono delle liste di domande e risposte e viene valutato se la risposta fornita dal modello in analisi è corretta o meno confrontandola con la risposta nota o con altri metodi che a loro volta coinvolgono altri modelli LLM.

Anche questi sistemi di valutazione possono essere imprecisi, per cui alcuni siti come https://chat.lmsys.org danno una classifica basata su giudizi umani per valutare la bontà dei modelli.

Rank* (UB)	Model	Arena Score	95% CI	Votes	Organization	License	Knowledge Cutoff
1	o1-preview	1355	+12/-11	2991	OpenAI	Proprietary	2023/10
2	ChatGPT-4o-latest (2024-09-03)	1335	+5/-6	10213	OpenAI	Proprietary	2023/10
2	o1-mini	1324	+12/-9	3009	OpenAI	Proprietary	2023/10
4	Gemini-1.5-Pro-Exp-0827	1299	+5/-4	28229	Google	Proprietary	2023/11
4	Grok-2-08-13	1294	+4/-4	23999	xAI	Proprietary	2024/3
6	GPT-4o-2024-05-13	1285	+3/-3	90695	OpenAI	Proprietary	2023/10
7	GPT-4o-mini-2024-07-18	1273	+3/-3	30434	OpenAI	Proprietary	2023/10
7	Claude 3.5 Sonnet	1269	+3/-3	62977	Anthropic	Proprietary	2024/4
7	Gemini-1.5-Flash-Exp-0827	1269	+4/-4	22264	Google	Proprietary	2023/11
7	Grok-2-Mini-08-13	1267	+4/-5	22041	xAI	Proprietary	2024/3
7	Gemini Advanced App (2024-05-14)	1267	+3/-3	52218	Google	Proprietary	Online
7	Meta-Llama-3.1-405b-Instruct-fp8	1266	+4/-4	31280	Meta	Llama 3.1 Community	2023/12
7	Meta-Llama-3.1-405b-Instruct-bf16	1264	+6/-8	5865	Meta	Llama 3.1 Community	2023/12

Classifica su valutazione umana dal sito https://chat.lmsys.org al 24/09/2024

Nella figura vediamo la classifica stilata da richieste verificate da persone reali presente nel sito indicato precedentemente.
Un altro sito che presenta molte classifiche di questo tipo sui più svariati casi d'uso degli LLM e https://huggingface.co

I test basati sulla verifica umana possono essere applicati in casi particolari, come ad esempio una valutazione finale, proprio perché lunghi e costosi.
Invece durante il processo di sviluppo dovrebbe essere possibile verificare l'effetto di ogni singola scelta senza dover coinvolgere una persona "intelligente" per controllare centinaia di risposte.
Spesso infatti lo sviluppatore verifica i benefici che il suo intervento ha portato per i casi specifici per cui lo ha immaginato, ma non si accorge dei peggioramenti che sono avvenuti in altri casi che non sono direttamente sotto osservazione.

Le risposte date dai sistemi "intelligenti" non sono facili da controllare: possono cambiare di volta in volta in base alla "temperatura" impostata, possono deviare con "allucinazioni" da quanto atteso, a

volte sono simili ma il senso della risposta non è esatto, a volte vengono mescolate parti che non sono pertinenti tra loro (es: viene generata una nuova legge che non esiste a partire dalla descrizione di due leggi diverse).

Vi sono vari metodi per verificare la risposta di un LMM:

1. Chiedere ad un altro LMM (di capacità superiore) la risposta e confrontarla con quella ottenuta.
2. Specificare la risposta assieme alla domanda e quindi confrontarla con quella ottenuta dall' LLM.
3. Chiedere all' LLM la risposta in un formato trattabile da un programma (JSon, XML, …) e quindi usare metodi tradizionali di confronto.

L'appalto che proponiamo è la realizzazione di un programma che permetta di valutare la capacità di rispondere ad una lista di problemi/domande di un sistema che usa tecniche di Intelligenza Artificiale.
Il programmatore che usa il sistema di test potrà così comprendere in modo esaustivo gli effetti delle varie scelte che farà di volta in volta, pervenendo al sistema più performante.

Caratteristiche e Requisiti Obbligatori

L' azienda Zucchetti Spa chiede di sviluppare una applicazione che svolga i seguenti compiti:

1. Archiviazione di una lista di domande e delle risposte attese.
2. Programma di esecuzione del test che attraverso una API ponga le domande ad un programma esterno e ne registri la risposta.
3. Programma di valutazione della correttezza / verosimiglianza delle risposte ricevute.
4. Procedura di presentazione dei risultati dell'esecuzione del test.
5. I punti precedenti devono essere integrati in un unico sistema che permetta di utilizzarli come parte di un insieme e non forniti come utility separate.

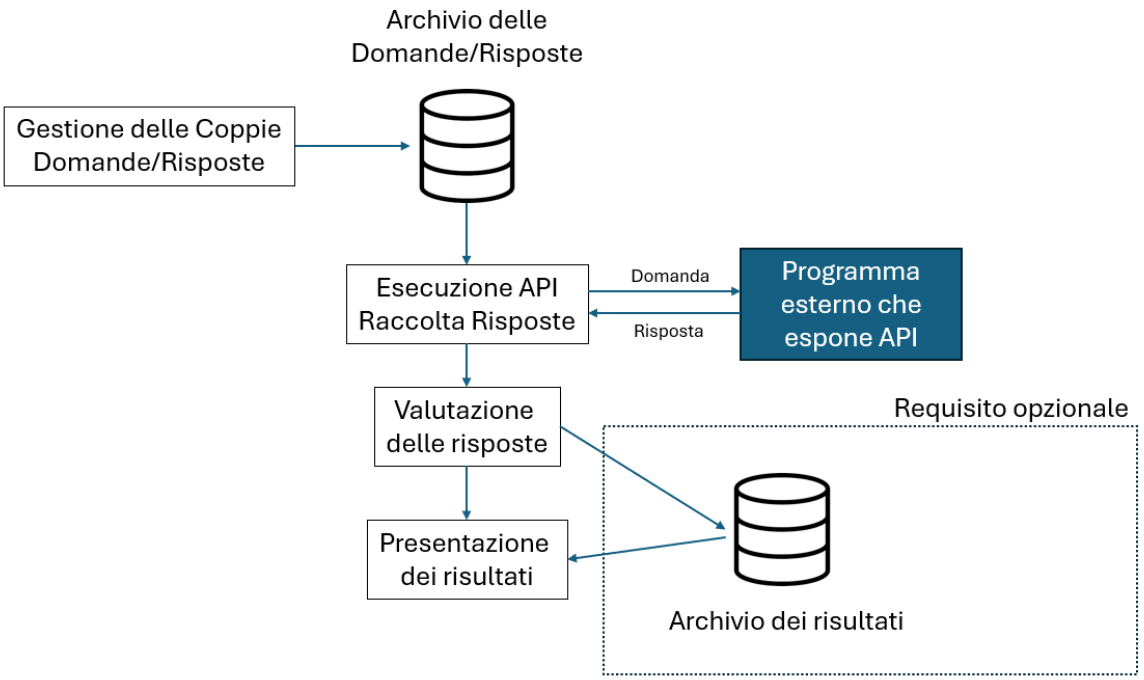
Il primo punto richiede di organizzare una raccolta di domande-risposte. Lasciamo agli sviluppatori la scelta del metodo per archiviare queste informazioni: può essere un database relazionale o un semplice file su disco con un formato predefinito.
Deve però essere sviluppata un'interfaccia "user friendly" per caricare, modificare e cancellare i dati contenuti nell'archivio.

Il secondo punto richiede di sviluppare un ciclo di esecuzione che legga i dati delle coppie domande/risposte e quindi sottoponga ad un sistema esterno attraverso una API Rest secondo lo standard OpenAPI 3.1 la domanda e raccolga la risposta data dal programma chiamato.
I risultati possono essere utilizzati uno ad uno o registrati su un apposito archivio.
Tra i requisiti opzionali è descritta una proposta per gestire in modo completo questa fase.

Il terzo punto è il cuore del problema proposto: valutare se la risposta ottenuta dal programma chiamato via API sia effettivamente compatibile con la risposta attesa.Per questo compito possono

essere utilizzati vari metodi come già suggerito, ma si tratta di un problema aperto e vorremmo ricevere proposte innovative.
Il primo sistema, il più naturale, è far valutare ad un LLM di grandi capacità se i due testi, quello atteso e quello ottenuto, dicono la stessa cosa. Pur essendo molto efficace, questo metodo continua ad utilizzare un LLM che a sua volta potrebbe avere allucinazioni o non capire correttamente quanto proposto.
Si potrebbe quindi utilizzare un metodo più tradizionale, per esempio un BM25 basato su keywords, anche se diventa necessario trovare un sistema per gestire sinonimi e polisemie. Questo metodo potrebbe anche essere in aggiunta e non solo in sostituzione del precedente.
Per analizzare il contenuto si potrebbe anche chiedere ad un LLM di estrarre delle parti significative in formato JSon e quindi operare sui dati estratti con metodi tradizionali.

Un ulteriore problema è dato dal fatto che a volte le risposte contengono anche informazioni aggiuntive che però possono cambiare il senso al resto del discorso.
Bisogna quindi anche controllare che il testo verificato sia la totalità del testo e non seguano altri discorsi non correlati. Questo, che apparentemente è un problema marginale, è invece molto importante quando si tenta di usare gli LLM per guidare le procedure: se l'utente espone due concetti alle volte ne viene colto uno solo, il più evidente, ed il secondo viene scartato con ovvi problemi nel risultato finale.



Schema di massima dell'applicazione "Artificial QI"

L'ultimo punto della procedura è la presentazione dei risultati. Ci attendiamo che ogni esecuzione debba valutare decine se non centinaia di domande, per cui dovrà essere costruito un sistema di aggregazione dei risultati con un indice sintetico della bontà generale di quanto ottenuto, una evidenziazione dei casi migliori e peggiori e la possibilità di verificare tutte le risposte ottenute.

Segue il requisito che richiede che il sistema sia organico e non in parti separate. L'applicazione dovrà essere unica e composta delle varie parti che sono state descritte.
L'uso immaginato è che un operatore specifichi le domande che vuole porre al sistema in esame, dia una indicazione delle risposte che ritiene corrette e quindi a partire dallo stesso ambiente possa eseguire un test e valutare i risultati. Il tutto senza dover uscire dal sistema e lanciare programmi separati.

L'azienda Zucchetti si rende disponibile a fornire un programma esterno che espone delle API da testare. Il programma contiene il manuale di alcune procedure e dovrebbe rispondere a domande dell'utente sul funzionamento delle stesse.
Questa è da considerarsi una opportunità per avere una parte del sistema già pronta, se gli studenti dispongono di altri sistemi da testare l'azienda Zucchetti non vincola questo aspetto.
Si richiede però che le domande e le risposte siano in linguaggio naturale e non sistemi codificati o espressi in formalismi analizzabili direttamente da un programma come XML o JSon.
Può essere accettato che alcune coppie di Domande / Risposte siano con metodi formalizzati (soprattutto risposte), ma il sistema deve poter valutare anche domande / risposte libere.

Il progetto ha una parte centrale di esplorazione delle tecnologie, mentre il resto del programma è relativamente basato su sistemi noti. L'azienda quindi vuole dare dei consigli rispetto alla sequenza di risultati attesi.

Nella prima fase, dove verrà sviluppato il PoC (Proof of concept) consigliamo di esplorare soprattutto gli algoritmi di confronto tra risposte attese e risposte ottenute. Senza un buon risultato in questo punto il resto del programma ha un valore minore.

La seconda fase, quella dove viene sviluppato l'MVP (Minumum Viable Product) deve realizzare i requisiti indicati come obbligatori, di fatto è il sistema minimo per poter usufruire dei risultati ottenuti dalla capacità di confronto.

I requisiti opzionali portano ad un prodotto completo che può appunto essere sviluppato qualora i tempi a disposizione fossero compatibili con la realizzazione delle potenzialità indicate.

Per ogni fase, ma soprattutto a partire dall'MVP, è importante che alla realizzazione si accompagni una adeguata copertura con test automatici. Ci attendiamo infatti che il sistema sia soggetto a revisioni e quindi verificabile in modo adeguato solo attraverso la ripetizione automatizzata dei casi d'uso principali.

Lo sviluppo per passaggi successivi, PoC, MVP e prodotto finale, richiede l'adozione di pattern architetturali di comprovata duttilità e robustezza. Ad ogni iterazione il codice dovrà essere revisionato, aumentato ed esteso, quindi sistemi fragili o senza una validità certa non sono adatti. Inoltre lo spiccato carattere esplorativo del tema proposto aumenterà il numero delle revisioni necessarie proprio per le scoperte che verranno fatte sviluppando le prime interazioni.

Requisiti Opzionali

Il cuore del problema proposto è la valutazione di verosimiglianza delle risposte ottenute e i requisiti obbligatori portano ad un sistema minimo per esplorare il problema.
I requisiti opzionali tendono a fornire un sistema utilizzabile come ambiente di test nel contesto di una pipeline di produzione del software.

Archiviazione dei risultati di test: questo è il requisito opzionale più importante, dopo una esecuzione del test è interessante poter archiviare il risultato per poi confrontarlo con altre esecuzioni. Questo richiede una gestione delle esecuzioni con la possibilità di salvare, ricercare, cancellare il documento prodotto da una esecuzione.

Confronto automatico tra run di test diverse: il primo confronto sarà dato dall'indice sintetico e dalle altre valutazioni, disponendo dei dati di una esecuzione precedente può essere sviluppato un programma di confronto per evidenziare quali parti sono migliorate e quali sono peggiorate. In questo modo si ha una valutazione dei risultati che permette di capire meglio come ogni scelta nel programma che espone le API ha influenzato il risultato finale.

Archiviazione di set di domande/risposte: per come sono stati proposti i requisiti obbligatori viene richiesto un unico set di domande / risposte che viene eseguito nella sua interezza.
Un'interessante estensione è avere un archivio che possa gestire più set di domande / risposte per verifiche di aspetti diversi del sistema o di più di un sistema.

Gestione della configurazione della chiamate alle API esterne: i programmi espongono API secondo le loro regole, per poter testare vari sistemi è opportuno poter configurare il punto di chiamata delle API in modo da potersi adeguare a prodotti diversi.

Suggerimenti

L'interfaccia web del sistema ChatGPT è disponibile a questo indirizzo web:
https://chat.openai.com/

Un buon sito dove trovare modelli alternativi è:
https://chat.lmsys.org/

La più grande raccolta di modelli LLM per i più svariati compiti è:
https://huggingface.co/

Un sistema che permette di eseguire un modello LLM in locale con API identiche a OpenAI è:
https://github.com/ggerganov/llama.cpp

Un’ottima libreria per la ricerca semantica e con molti esempi di fine-tuning di modelli o di creazione di modelli da zero è:
https://github.com/neuml/txtai

Parte di questo documento è stato revisionato da ChatGPT.

Variazioni ai requisiti

In corso d'opera non sarà possibile variare/modificare i requisiti minimi (obbligatori per accettare il prodotto). Sarà invece possibile variare i requisiti opzionali, in quanto saranno i gruppi vincitori dell'appalto a modificarli / eliminarli / aggiungerli.

Documentazione

Il progetto dovrà essere supportato dalla documentazione minima richiesta per il corso di Ingegneria del software e dovrà essere fornito un manuale per l'utilizzo ed un manuale per chiunque voglia estendere l'applicazione.

Garanzia e Manutenzione

L'azienda Zucchetti SPA è interessata a questo progetto come dimostrazione della fattibilità dell'obiettivo utilizzando le tecnologie web. Costituirà titolo preferenziale nella valutazione delle proposte la pubblicazione del progetto sul sito “github.com” o altri repository pubblici, in conformità con i relativi requisiti di natura open-source, per favorire la continuità del prodotto risultante.

Rinvio

Per tutto quanto non previsto nel presente capitolato, sono applicabili le disposizioni contenute nelle leggi e nei collegati per la gestione degli appalti pubblici.

Contatti

Per qualsiasi informazione la persona di riferimento in Zucchetti è Gregorio Piccoli all’indirizzo email gregorio.piccoli@zucchetti.it