DL-NLP

Iloc:	Institut d'Ensenyaments a Distància de les Illes Balears	Imprès per:	Carlos Sanchez Recio
Curs:	Models d'intel·ligència artificial	Data:	dilluns, 24 de febrer 2025, 20:20
Llibre:	DL-NLP		

Taula de continguts

1. Deep Learning per al processament del llenguatge natural

Al lliurament anterior hem vist com utilitzar gramàtiques per al processament del llenguatge natural. Els sistemes que es basen en l'anàlisi sintàctica i semàntica han demostrat èxit en moltes tasques, però el seu rendiment és limitat a causa de la complexitat enorme dels fenòmens lingüístics en el text real. Donada la gran quantitat de text que hi ha disponible en formats que les computadores poden llegir, té sentit considerar quins enfocaments basats en aprenentatge automàtic basat en dades poden ésser més efectius. En aquest lliurament treballarem aquesta hipòtesi amb les eines d'aprenentatge profund.

Començam veient com millorar l'aprenentatge representant les paraules com a punts en un espai d'alta dimensionalitat, més que no com a valors individuals. A continuació veurem com les xarxes neuronals recurrents poden capturar el significat i el context a llarga distància a mesura que es processa el text seqüencialment. Més endavant ens centram en la traducció automàtica, un dels èxits més grans de l'aprenentatge profund aplicat a l'NLP. Veurem models que es poden entrenar amb grans quantitats de text sense etiquetar i llavors aplicar a tasques específiques.

Per aquesta part més descriptiva, seguirem el capítol 25 del llibre Artificial Intelligence, A Modern Approach de Russell i Norvig. En aquest capítol del llibre hi han contribuït, a més, Jacob Devlin i Mei-Wing Chang.

Després veurem una panoràmica d'implementacions pràctiques basades en les idees d'aquest capítol, des de ChatGPT, Gemini, Claude i LlaMa fins a Perplexity i DeepSeek.

2. Word Embeddings

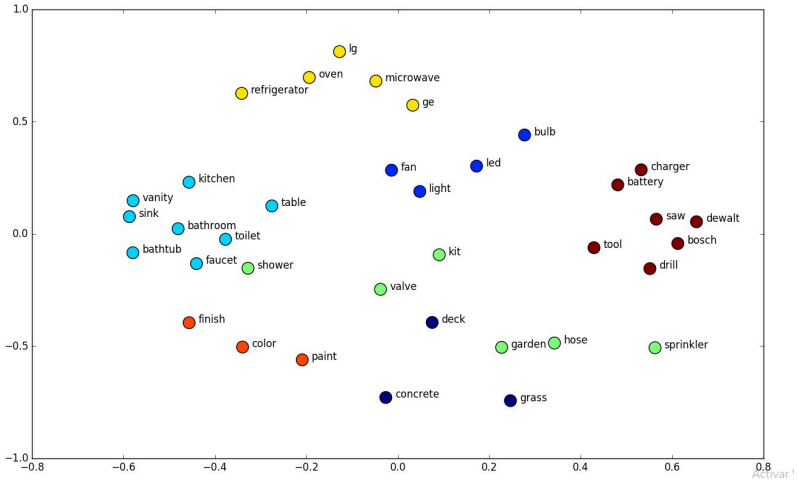
Convé representar les paraules sense la necessitat d'una enginyeria de característiques realitzada a mà, però que permeti la generalització entre mots relacionats. Les relacions entre paraules poden ser sintàctiques, semàntiques, de tema, de sentiment, o d'altres.

Com hauríem de codificar una paraula en un vector d'entrada x per usar-lo en una xarxa neuronal? Podríem usar un one-hot vector (la paraula i-èsima del diccionari amb un bit a 1 a la posició i i un 0 a tota la resta de posicions), però una representació així no capturaria la semblança entre les paraules.

Seguint la màxima del lingüista John Firth (1957) "coneixereu les paraules per la seva companyia" podríem representar cada paraula per un vector de recomptes de n-grams de totes les frases en què la paraula apareix. Però, els recomptes de n-grams es disparen. Amb un vocabulari de 100000 paraules (10^5), hi ha 10^{25} 5-grams a tenir en compte. S'aconsegueix una generalització millor si ho reduïm a un vector de dimensió més petita, potser amb només alguns centenars de dimensions. Aquest vector més petit s'anomena **word embedding**: un vector de dimensionalitat relativament baixa que representa una paraula. Els *word embeddings* s'aprenen automàticament a partir de les dades.

D'una banda, el *word embedding* de cada paraula és just un vector de nombres, on les dimensions individuals i els seus valors numèrics no tenen una interpretació fàcil.

D'una altra banda, l'espai de característiques té la propietat que paraules semblants acaben tenint vectors semblants, com podem veure a l'exemple següent.



Imatge: *Word embeddings mostrant la semblança entre paraules properes*

Per raons que no es comprenen totalment, els vectors de word embedding tenen propietats més enllà de la proximitat entre paraules semblants. Per exemple, suposem que miram el vector A de la paraula **Atenes** i el vector B de la paraula **Grècia**. Entre aquestes paraules, la diferència $B - A$ sembla que codifica la relació país/capital. Altres parells de paraules (França i París, Rússia i Moscou) tenen aproximadament la mateixa diferència de vectors.

Amb aquesta propietat es poden resoldre problemes d'analogia de paraules com "Atenes és a Grècia com Oslo és a què?". Si C representa Oslo i D la incògnita, podem suposar $B - A = D - C$ i aïllar $D = C + (B - A)$. Quan es calcula aquest nou vector D, resulta que és més a prop de Noruega que de cap altra paraula.

Això no obstant, no hi ha cap garantia que un algorisme de word embedding en particular sobre un corpus concret capturarà una relació semàntica específica. Els word embeddings són populars perquè han demostrat ser una bona representació per a les tasques lingüístiques com resposta de preguntes, traducció o resum, no perquè estigui garantit que poden respondre preguntes d'analogia en si.

Utilitzar word embeddings en comptes de codificacions one-hot de les paraules resulta útil en pràcticament totes les aplicacions de *deep learning* a NLP. En molts de casos es poden usar vectors preentrenats, obtinguts de diverses fonts. Els diccionaris de vectors habituals són Word2Vec, GloVe i FastText, que té embeddings per a 157 llengües, entre elles el català. L'ús de models preentrenats pot estalviar molt de temps i esforç.

També podem entrenar els nostres propis models. Això es fa al mateix temps que s'entrena una xarxa per a una tasca en particular. A diferència dels embeddings preentrenats, els word embeddings produïts per a una tasca específica es poden entrenar amb un corpus seleccionat amb cura i tendiran a emfatitzar aspectes de les paraules útils per a la tasca.

3. Xarxes neuronals recurrents

Amb els word embeddings tenim una bona representació per a les paraules isolades, però el llenguatge consisteix en una seqüència ordenada de paraules en la qual el **context** de les paraules del voltant és important. En tasques simples com l'etiquetatge de parts del llenguatge (anàlisi sintàctica), una finestra de mida fixa per ventura de cinc paraules sol donar prou context.

Tasques més complexes com la resposta de preguntes o la resolució de referències poden requerir dotzenes de paraules com a context. Per exemple, a la frase "N'Eduard em va dir que en Miquel estava molt malalt, així que **el** vaig dur a l'hospital", saber que el pronom **el** es refereix a en Miquel i no a n'Eduard requereix un context que va de la primera a la darrera paraula de la frase.

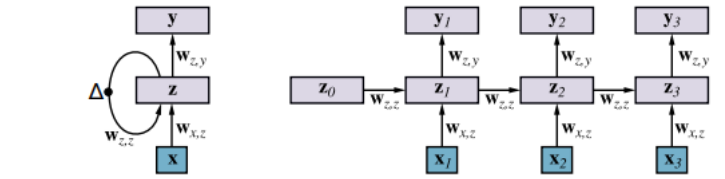
3.1. Models de llenguatge amb RNN

Començarem amb el problema de crear un model del llenguatge amb prou context. Un model de llenguatge és una distribució de probabilitat sobre seqüències de paraules. Permet predir la propera paraula d'un text donades totes les paraules prèvies, i sovint s'usa com a bloc per resoldre tasques més complexes.

Si es construeix un model de llenguatge amb un model de n-gram o una xarxa neuronal feedforward amb una una finestra fixa de n paraules podem trobar dificultats a causa del problema del context: el context passarà la mida de la finestra fixa, el model tendrà massa paràmetres, o totes dues coses alhora.

A més, la xarxa feedforward té el problema de la **asimetria**: qualsevol cosa que aprengui sobre l'aparició del pronom **el** com a dotzena paraula de la frase ho haurà de reaprendre per a qualsevol altra posició, perquè els pesos són diferents en cada posició de paraula.

La xarxa neuronal recurrent (*recurrent neural network*, RNN) està dissenyada per a processar sèries temporals, una dada en cada instant. Això suggereix que la RNN pot ser útil per a processar el llenguatge, una paraula en cada instant.



Imatge: *Xarxa neuronal recurrent. A l'esquerra, plegada; a la dreta, desplegada*

En un model de llenguatge amb RNN cada paraula d'entrada es codifica amb un vector de word embeddings. Hi ha una capa oculta que es passa com a entrada des d'un instant de temps al següent. Aleshores farem classificació multiclasse: les classes són les paraules del vocabulari. FInalment la sortida serà una distribució de probabilitat softmax sobre els possibles valors de la propera paraula de la frase.

L'arquitectura RNN resol tres problemes:

- Que hi hagi massa paràmetres
- L'asimetria, perquè hi ha els mateixos pesos per a qualsevol posició de les paraules
- De vegades, també el problema d'un context massa limitat

A la pràctica, els models RNN funcionen bé en una varietat de tasques, però no en totes. Pot ser difícil predir si aniran bé en un problema determinat.

Per entrenar un model de llenguatge amb RNN, les entrades, x_t, són les paraules del corpus de text d'entrenament, i les sortides observades són les mateixes paraules desplaçades en una posició. Per exemple, per al text d'entrenament "hello world", la primera entrada x_1 és el word embedding de "hello" i la primera sortida y_1 és el word embedding de "world". Estam entrenant el model per a predir la paraula següent, i esperam que per aconseguir-ho usarà la capa oculta per representar informació útil. Es calcula la diferència entre la sortida observada i la sortida real calculada per la xarxa, i es propaga cap enrere en el temps, mantenint els mateixos pesos en tots els instants.

Una vegada que s'ha entrenat el model, es pot usar per generar text aleatori. Donam al model una paraula d'entrada inicial x_1, a partir de la qual produirà una sortida y_1 que és una distribució de probabilitat softmax sobre les paraules. Mostrejam una sola paraula de la distribució, enregistram la paraula com a sortida per a l'instant t i la realimentam com a pròxima paraula d'entrada, x_2. Repetim mentre es vulgui. A l'hora de mostrejar la densitat de probabilitat, hi ha opcions: es podria prendre sempre la paraula més probable, es podria mostrejar d'acord amb la probabilitat de cada paraula, o fins i tot es podria sobrerepresentar les paraules més poc probables, per afegir més varietat a la sortida generada. El pes del mostratge és un hiperparàmetre del model.

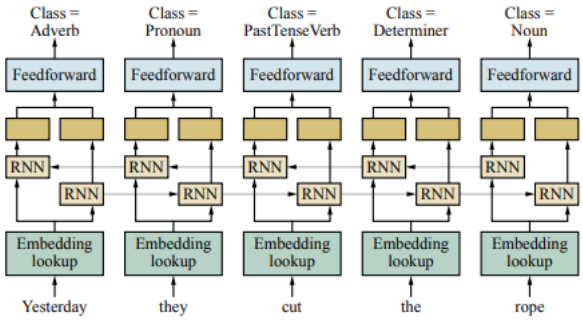
3.2. Classificació amb RNN

També es poden usar xarxes neuronals recurrents per a d'altres tasques de llenguatge, com l'etiquetatge de les parts de la parla (POS tagging) o la resolució de correferències. En els dos casos, les capa d'entrada i la capa oculta seran les mateixes, però en un POS tagger la sortida serà una distribució softmax sobre les POS, mentre que en la resolució de correferències la capa de sortida serà una distribució softmax sobre els possibles antecedents. Per exemple, quan la xarxa arriba a l'entrada el a la frase "n'Eduard em va dir que en Miquel estava molt malalt, així que el vaig dur a l'hospital" hauria de donar una probabilitat alta a "Miquel".

L'entrenament d'una RNN per a classificació és com el del model de llenguatge. L'única diferència és que les dades d'entrada necessitaran etiquetes, de categories lèxiques o indicacions de referència. Això complica molt l'adquisició de dades comparat amb el model de llenguatge, en què basta text sense etiquetar.

En un model de llenguatge volem predir la paraula n donades les paraules anteriors. Però en classificació, no hi ha cap raó per limitar-nos a les paraules anteriors. Pot ser molt útil mirar cap endavant en la frase. Per exemple, el referent de "el" seria diferent si la frase acabàs dient "a veure en Miquel" en comptes de "a l'hospital". A més se sap a través d'experiments de seguiment dels ulls que els lectors humans no van estrictament d'esquerra a dreta.

Per capturar el context de la dreta, es poden usar RNN bidireccionals, que concatenen un model diferent de dreta a esquerra al model d'esquerra a dreta.



Imatge: *RNN bidireccional per a POS tagging*

Les RNN també es poden usar en tasques de classificació a nivell de frase o de document, en què hi ha una única sortida al final, en comptes d'una cadena de sortides, una en cada instant. Per exemple, en **anàlisi de sentiment**, l'objectiu és classificar un text com a positiu o negatiu. Per exemple "aquesta pel·lícula està mal dirigida i mal interpretada" s'ha de classificar com a negatiu. Alguns sistemes d'anàlisi de sentiment tenen més de dues categories, o un valor escalar.

Usar RNN per a una tasca a nivell de frase és més complex, ja que s'ha d'obtenir una representació agregada per a tota la frase, y, a partir de les sortides de cada paraula y_t de la xarxa. La forma més simple d'aconseguir-ho és usar l'estat ocult que correspon a la darrera paraula de l'entrada, ja que la RNN haurà llegit la frase completa en aquell instant. Però això pot esbiaixar el model a fer més atenció cap al final de la frase.

Una altra tècnica habitual és fer un *pool* amb tots els vectors ocults. Per exemple, amb average pooling es calcula el promig element a element sobre tots els vectors ocults.

$$\hat{z} = \frac{1}{s}\sum_{t=1}^{s} z_t$$

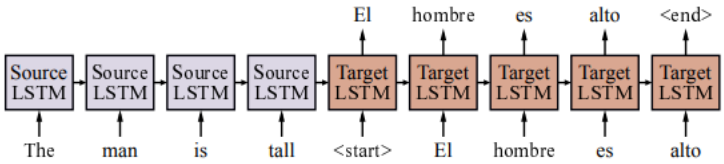
El vector d-dimensional agregat $\hat{z}$aleshores es pot introduir a una o més capes feedforward abans d'entrar a la capa de sortida.

3.3. LSTM per a tasques de llenguatge

Les RNN de vegades resolen el problema de la limitació del context. En teoria, qualsevol informació pot passar d'una capa oculta a la següent qualsevol nombre d'intervals de temps. A la pràctica, però, la informació es pot perdre o distorsionar. El problema a les RNN és semblant al **gradient evanescent**, llevat que ara tenim capes al llarg del temps en comptes de capes profundes.

El model LSTM (long short-term memory) és un tipus de RNN amb unitats de *gating* que no presenten el problema de reproduir de forma imperfecta un missatge d'un instant de temps a l'altre. En canvi, les LSTM poden recordar algunes parts de l'entrada, i copiar-la cap al següent instant, i oblidar-ne altres parts.

Una LSTM pot aprendre a crear una característica latent per a la persona i el nombre del subjecte d'una frase, per exemple, i copiar aquesta característica endavant sense alterar fins que és necessària per triar la forma verbal. Una RNN regular sovint es confon quan hi ha moltes paraules entre el subjecte i el verb.



Imatge: *LSTM per a traducció automàtica*

L'article Long short-term memory de Sepp Hochreiter i Jürgen Schmidhuber ha tengut un gran impacte d'ençà de la seva publicació el 1997. Acumula més de 100000 citacions, d'acord amb Google Scholar.

4. Models sequence-to-sequence

Una de les tasques més estudiades en NLP és la traducció automàtica (*machine translation*, MT). El seu objectiu és traduir una frase des d'una llengua d'origen (*source*) fins a una llengua de destinació (*target*). Els models de traducció automàtica s'entrenen amb un gran corpus de parells de frases en les llengües d'origen i de destinació. L'objectiu és traduir amb precisió frases noves que no formen part de les dades d'entrenament.

Es poden usar RNN per crear un sistema de traducció automàtica? Certament es pot codificar la frase d'origen amb una RNN. Si hi hagués una correspondència de les paraules d'una a una entre origen i destinació, es podria tractar com un problema d'etiquetatge, però la correspondència no és d'una en una i a més l'ordre pot variar. Aleshores, com es genera una frase en la llengua de destinació?

Sembla que podem generar una paraula cada instant, però seguir el context de forma que es recordin parts de la frase que encara no s'han traduït, i saber quines parts ja s'han traduït per no repetir-les. Algunes frases s'hauran de processar senceres abans no es puguin començar a traduir. Per tant, la generació de cada paraula de destinació dependrà de tota la frase d'origen i de les paraules de destinació ja generades.

Això lliga estretament la generació de text en MT al model de llenguatge amb RNN estàndard. Si hem entrenat una RNN amb text en anglès, serà més probable que generi "big dog" que no "dog big". Això no obstant, no volem generar qualsevol frase aleatòria, sinó la frase que correspon a la frase d'entrada. La forma més simple d'aconseguir-ho és usar dues RNN, una per a l'origen i una altra per a la destinació. S'executa la RNN d'origen sobre tota la frase d'entrada i després s'usa el darrer estat ocult de la RNN d'origen com a estat ocult inicial de la RNN de destinació. D'aquesta forma, cada paraula de destinació està condicionada implícitament a tota la frase d'entrada i a les paraules prèvies de destinació.

Aquesta arquitectura de xarxa neuronal és un **model sequence-to-sequence** bàsic, i s'utilitza a més de traducció automàtica, en altres tasques com generació de descripcions d'imatges, o resum, reescrivint un text llarg en un de més breu mantenint el significat.

Els models de seqüència a seqüència bàsics varen suposar un gran avanç en la traducció automàtica. Aconseguiren una reducció dels errors en un 60% respecte dels sistemes anteriors. Així i tot, aquests models tenen tres inconvenients principals:

Biaix cap al context proper: qualsevol cosa que una RNN hagi de recordar sobre el passat, s'ha de representar en el seu estat ocult. Per exemple, considerem que una RNN està processant la paraula 57 d'una frase de 70 paraules. L'estat ocult segurament contendrà més informació sobre la paraula 56 que no sobre la 5, ja que en cada instant l'estat ocult s'actualitza substituïnt informació que ja té per nova informació. Aquest comportament és part del disseny intencionat del model i sovint té sentit en NLP, ja que el context proper sol ser més important. Però el context llunyà també pot ser crucial, i es perd en un model RNN. Fins i tot les LSTM tenen dificultats amb aquesta tasca.

Mida fixa del context: En un model RNN per a traducció tota la frase d'origen es comprimeix en un únic vector d'estat ocult de dimensió fixa. Una LSTM típicament té 1024 dimensions i si hi hem de representar per exemple una frase de 64 paraules, això només dona 16 dimensions per paraula, insuficient en frases complexes. Si s'incrementa la mida del vector d'estat ocult, això alenteix l'entrenament i pot dur a sobreajust.

Processament seqüencial lent: les xarxes neuronals aprofiten el suport de hardware eficient per a l'aritmètica de matrius processant les dades d'entrenament en blocs (*batches)*. Les RNN, en canvi, estan restringides a manipular les dades una paraula en cada instant.

4.1. Atenció

I si la RNN de destinació es condiciona sobre tots els vectors ocults de la RNN d'origen, en comptes de només el darrer?

Això mitigaria el problema del biaix cap al context proper i els límits d'una mida finita del context, permetent al model d'accedir a qualsevol mot previ igualment.

Una forma d'aconseguir--ho és concatenar tots els vectors ocults de la RNN d'origen. Però això incrementaria molt el nombre de pesos, i per tant el temps de computació així com un possible sobreajust (*overfitting*). En lloc d'això, podem aprofitar que quan la RNN genera una paraula en cada instant, és probable que només una petita part de la seqüència d'origen sigui rellevant per a la paraula actual.

La RNN de destinació ha de fer atenció a cada part de l'origen per a cada mot. Suposem que tenim una xarxa entrenada per traduir d'anglès a espanyol. L'entrada és "the front door is red" seguida d'un marcador de final de frase. Així, hauria de fer atenció a "The" i generar "La", després a "door" per generar "puerta", i així successivament. Notem que per decidir que "the" tradueix com a "la" també cal fer atenció a "door", que tradueix en espanyol a un substantiu amb gènere femení i nombre singular. Així es pot decidir a quina de les quatre opcions d'article ("el", "la", "los", "las") correspon en aquest cas l'anglès "the".

Aquest concepte es pot formalitzar amb un component de xarxa neural anomenat atenció, que crea un resum contextual de la frase d'origen en una representació de dimensionalitat fixa. El vector de context c_i conté la informació més rellevant per generar la següent paraula de la seqüència de destinació, i s'utilitzarà com a entrada addicional a la RNN de sortida. Un model sequence-to-sequence que usa atenció s'anomena **model de seqüència a seqüència atencional**. Si la RNN destinació estàndard s'escriu com

$$h_i = RNN(h_{i-1}, x_i)$$

la RNN destinació en models sequence-to-sequence atencionals s'escriu

$$h_i = RNN(h_{i-1}, [x_i; c_i]),$$

afegint-hi el vector de context c_i. El vector combinat $[x_i; c_i]$ és la concatenació dels vectors d'entrada i de context, amb les definicions següents.

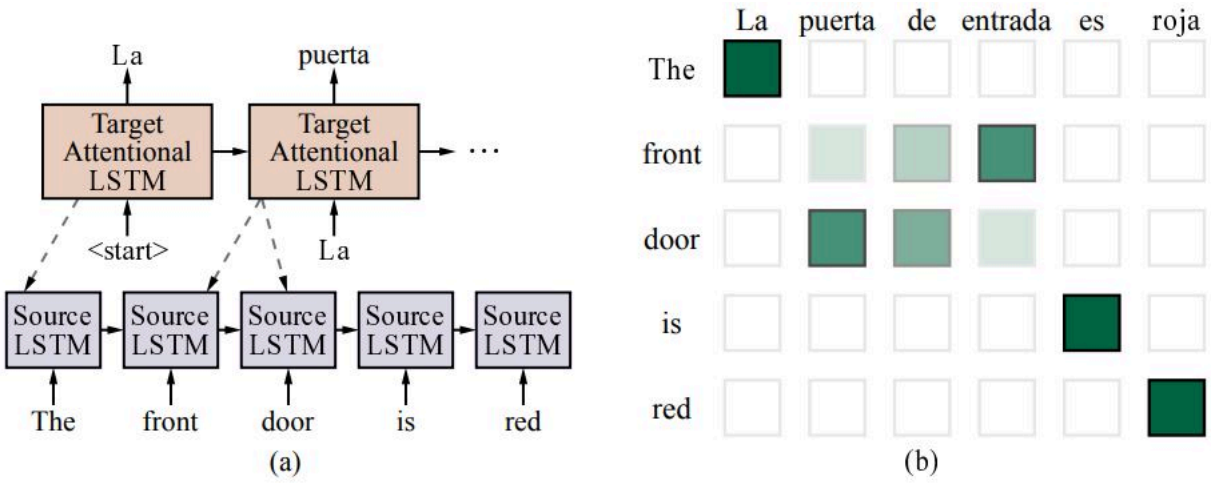
$$r_{ij} = h_{i-1} \cdot s_j$$

$$a_{ij} = \frac{e^{r_{ij}}}{\sum_k e^{r_{ik}}}$$

$$c_i = \sum_j a_{ij} \cdot s_j$$

En aquestes definicions, h_{i-1} és el vector de la RNN destí que s'usarà per predir la paraula a l'instant i, i s_j és la sortida a l'instant j de la RNN font (*source*).Tant h_{i-1} com s_j són vectors d-dimensionals, on d és la mida oculta. El valor r_{ij} és aleshores la puntuació d'atenció entre l'estat de destinació actual i la paraula d'origen j. Aquestes puntuacions es normalitzen a una probabilitat a_{ij} usant una *softmax* sobre totes les paraules d'origen. Finalment, aquestes probabilitats s'usen per generar un promig ponderat dels vectors de la RNN font, c_i, un altre vector d-dimensional.

A continuació donam un exemple de model de seqüència a seqüència atencional.



Imatge: *a) Model atencional per a una traducció anglès-espanyol. Les línies discontínues representen l'atenció. b) Exemple de la matriu de probabilitat d'atenció per a un parell de seqüències bilingüe. Les capses més fosques representen valors més alts de α_{ij}. Les probabilitats sumen la unitat en cada columna.*

Hi ha alguns detalls importants d'entendre. Primer, la component d'atenció mateixa no té pesos apresos i suporta seqüències de longitud variable tant a l'entrada com a la sortida. Segon, l'atenció és un mecanisme latent. El programador no fixa quina informació s'usa i quan, el model ho aprèn. El mecanisme d'atenció també es pot combinar amb RNN multicapa. En aquest cas l'atenció se sol aplicar a cada capa.

La formulació probabilística softmax per a l'atenció té tres objectius. En primer lloc, fa que l'atenció sigui derivable, cosa que és necessària per poder usar-la amb retropropagació. Tot i que l'atenció no té pesos apresos, els gradients van cap enrere a través de l'atenció a les RNN d'origen i destinació. En segon lloc, la formulació probabilística permet al model capturar certs tipus de contextualització que poden no haver estat capturats per la RNN font, ja que l'atenció pot considerar la seqüència d'origen completa de cop, i aprendre a mantenir el que és important i ignorar la resta. En tercer lloc, l'atenció probabilística representa la incertesa: si la xarxa no sap exactament quina paraula d'origen traduir a continuació, pot distribuir les probabilitats d'atenció entre diverses opcions, i després triarà la paraula la xarxa RNN de destinació.

A diferència d'altres components de les xarxes neuronals, les probabilitats d'atenció sovint són interpretables i significatives intuïtivament. Per exemple, en el cas de la traducció automàtica, les probabilitats d'atenció sovint corresponen als alineaments paraula a paraula que una persona generaria. Això està il·lustrat a la part b) de la figura.

Els models de seqüència a seqüència són una solució natural per a la traducció automàtica, però gairebé qualsevol tasca de llenguatge natural es pot codificar com un problema de seqüència a seqüència. Per exemple, un sistema de resposta de preguntes es pot entrenar amb una entrada formada per una pregunta seguida d'un delimitador i seguida per una resposta.

A diferència de la majoria de components de les xarxes neuronals, les probabilitats d'atenció sovint poden interpretar-se i tenen un significat intuïtiu.

Els models de seqüència són una representació natural per a la traducció automàtica, però gairebé qualsevol tasca de llenguatge natural es pot codificar com un problema de seqüència a seqüència. Per exemple, un sistema de respondre preguntes (*question answering*) es pot entrenar sobre una entrada que consistesqui en una pregunta seguida d'un delimitador seguida de la resposta.

4.2. Descodificació

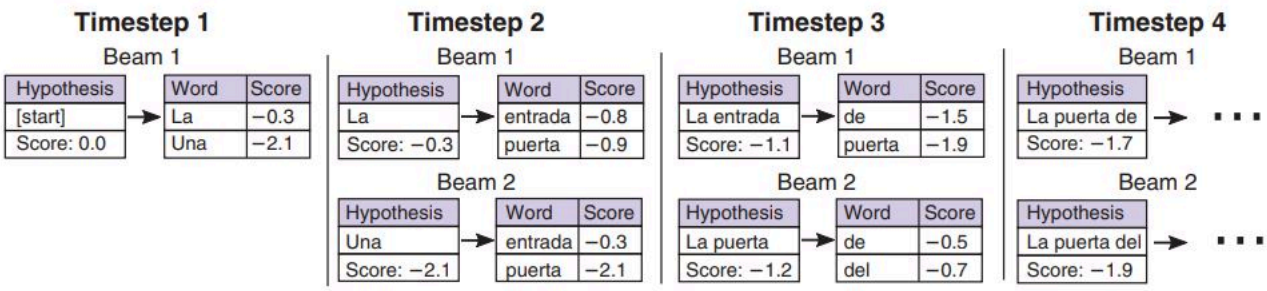
A l'etapa d'entrenament, un model de seqüència a seqüència mira de maximitzar la probabilitat de cada paralua en la sentència d'entrenament objectiu (*target*), condicionada a l'origen (*source*) i a totes les paraules anteriors objectiu. Una vegada que l'entrenament s'ha acabat, a partir d'una sentència origen el nostre objectiu és generar la sentència objectiu. Podem generar l'objectiu paraula per paraula, i després realimentar la paraula generada per al següent instant de temps. Aquest procediment rep el nom de **descodificació**.

La forma més simple de descodificació consisteix a seleccionar la paraula de probabilitat més alta en cada instant i realimentar aquesta paraula com a entrada per al següent instant. Aquesta variant s'anomena **descodificació cobdiciosa** (*greedy decoding*) perquè després que s'ha generat una paraula, el sistema està totalment compromès amb la seqüència que ha generat fins aquell moment, i ja no modificarà la seva decisió. Però el problema és que l'objectiu de la descodificació és maximitzar la probabilitat de la seqüència objectiu completa, cosa que la descodificació cobdiciosa pot no aconseguir. Per exemple, considerem un descodificador *greedy* que ha de traduir a l'espanyol la frase anglesa *The front door is red*.

La traducció correcta és *La puerta de entrada es roja*. Suposem que la RNN genera correctament la primera paraula *La* per a *The*. Notem que això ja presenta un problema, perquè *The* correspon a *El*, *La*, *Los* o *Las* segons el gènere i el nombre del substantiu que acompanya. A continuació, el traductor podria proposar *entrada* per a *front*. Però això ja seria una errada, perquè en espanyol primer hi hauria d'anar el substantiu, *puerta*, que ja no s'hi podria inserir. La descodificació cobdiciosa és ràpida, però el model no té cap mecanisme per corregir errades.

Es podria mirar de millorar el mecanisme d'atenció de forma que sempre atengués a la paraula de la dreta per mirar de realitzar una predicció correcta cada vegada. Però en moltes frases és impossible traduir correctament les primeres paraules d'una frase sense saber amb quines paraules acaba l'oració.

Una aproximació més bona és mirar de trobar una descodificació òptima, o almanco una de bona, usant un algorisme de cerca. Una opció habitual és la **cerca en feix** (*beam search*). En el context de descodificació en traducció automàtica, la cerca en feix típicament manté les k hipòtesis més bones en cada etapa, estenent-ne cada una amb una paraula usant les k paraules més bones, i a continuació se seleccionen les k més bones d'entre les k^2 noves hipòtesis. Quan totes les hipòtesis del feix generen el token especial <end>, l'algorisme dona com a resultat la hipòtesi de puntuació més alta.



Imatge: *Cerca en feix amb una amplada b=2. La puntuació de cada paraula és la log-probabilitat generada per la softmax de la RNN objectiu, i la puntuació de cada hipòtesi és la suma de les puntuacions de les paraules. A l'instant 3 (timestep 3) la hipòtesi "La entrada" només pot generar continuacions de baixa probabilitat, així que "cau del feix".*

A mesura que els models d'aprenentatge profund van tornant més precisos, es poden permetre usar una amplada de feix més petita. Els millors models neuronals de traducció automàtica actuals usen un feix entre 4 i 8, mentre que que els models estadístics anteriors podien usar un feix de 100 o més.

5. L'arquitectura Transformer

L'article **Attention is all you need** (Vaswani et al., 2018) va introduir l'arquitectura **transformer**, que fa servir un mecanisme d'**autoatenció** (*self-attention*) i modelitza el context a llarga distància sense una dependència seqüencial.

Es pot descarregar l'article en aquest enllaç.

5.1. Autoatenció

Anteriorment, en els models de seqüència a seqüència, l'atenció s'aplicava de la RNN destinació a la RNN origen. L'**autoatenció** (*self-attention*) estén aquest mecanisme de forma que cada seqüència d'estats ocults també fa atenció a ella mateixa (l'origen a l'origen, la destinació a la destinació). Això permet que el model capturi el context a llarga distància (i també proper) en cada seqüència.

La manera més directa d'aplicar l'autoatenció és aquella en què la matriu d'atenció es forma directament amb el producte escalar dels vectors d'entrada. Tanmateix, això és problemàtic. El producte escalar d'un vector amb ell mateix sempre és alt, de forma que cada estat ocult estarà esbiaixat a atendre cap a ell mateix. El transformer resol aquest problema projectant l'entrada en tres representacions diferents usant tres matrius de pesos diferents.

- El **vector consulta** (*query vector*) $q_i = W_q x_i$ és aquell des del qual es fa atenció, com l'objectiu (target) en el mecanisme estàndard d'atenció.
- El **vector clau** $k_i = W_k x_i$ és aquell al qual es fa atenció, com l'origen (*source*) en el mecanisme d'atenció bàsic.
- El **vector valor** $v_i = W_v x_i$ és el context que es genera.

En el mecanisme d'atenció estàndard, les xarxes de la clau i el valor són idèntiques, però té sentit que tenguin representacions diferents. Els resultats de codificar la paraula i-èsima, c_i, es poden calcular aplicant un mecanisme d'atenció als vectors projectats:

$$r_{ij} = (q_i \cdot k_j)\sqrt{(d)}$$

$$a_{ij} = \frac{e^{r_{ij}}}{\sum_k e^{r_{ik}}}$$

$$c_i = \sum_j a_{ij} \cdot v_j$$

on d és la longitud dels vectors k i q. Notem que i i j són índexs a la mateixa oració. ja que estam codificant el context usant auto-atenció. En cada capa del transformer, l'auto-atenció usa els vectors ocults que venen de la capa anterior, que al principi és la capa d'*embedding*.

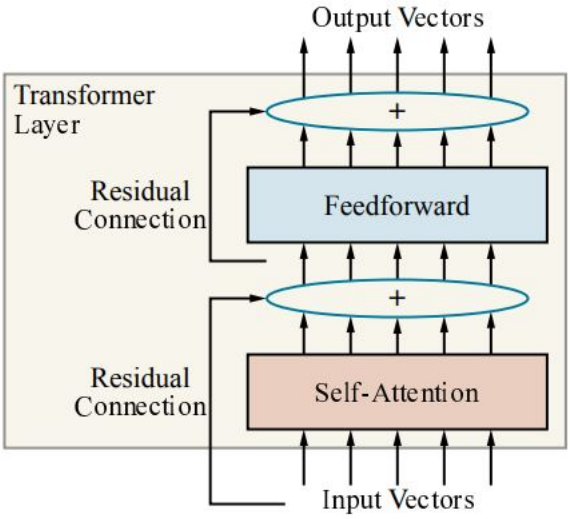
Aquí hi ha diversos detalls que s'han de tenir en compte. En primer lloc, el mecanisme d'auto-atenció és asimètric, ja que r_{ji} és diferent de r_{ij}. En segon lloc, s'ha afegit el factor d'escala $\sqrt{d}$ per millorar l'estabilitat numèrica. En tercer lloc, la codificació de totes les paraules d'una oració es pot calcular simultàniament, ja que les equacions de dalt es poden expressar amb operacions de matrius que es poden calcular eficientment en el maquinari especialitzat modern.

La tria de quin context s'ha d'utilitzar s'aprèn completament dels exemples d'entrenament, no s'especifica prèviament. El resum basat en el context, c_i, és una suma sobre totes les posicions prèvies a la sentència. En teoria, qualsevol informació de l'oració pot aparèixer a c_i, però a la pràctica, de vegades es perd informació important, perquè se'n fa el promig al llarg de tota la seqüència. Una forma de resoldre-ho és l'**atenció multi-cap** (*multiheaded attention*). Dividim la sentència en m fragments iguals i aplicam el model d'atenció a cada una de les m peces. Cada fragment té el seu propi conjunt de pesos. Després els resultats es concatenen per formar c_i. Concatenant en comptes de sumant, és més fàcil que una part important destaqui.

5.2. De l'autoatenció al transformer

L'autoatenció només és un element del model transformer. Cada capa d'un transformer conté diverses subcapes. A cada capa del transformer, primer s'hi aplica l'autoatenció. La sortida del mòdul d'atenció s'injecta a través de capes feedforward, on s'apliquen les matrius de pesos independentment a cada posició. Després de la primera capa feedforward, s'aplica una funció d'activació no lineal, típicament la ReLU. Per adreçar el problema potencial del gradient evanescent (*vanishing gradient*), s'afegeixen dues connexions residuals a la capa del transformer.

A continuació mostram un transformer d'una sola capa.

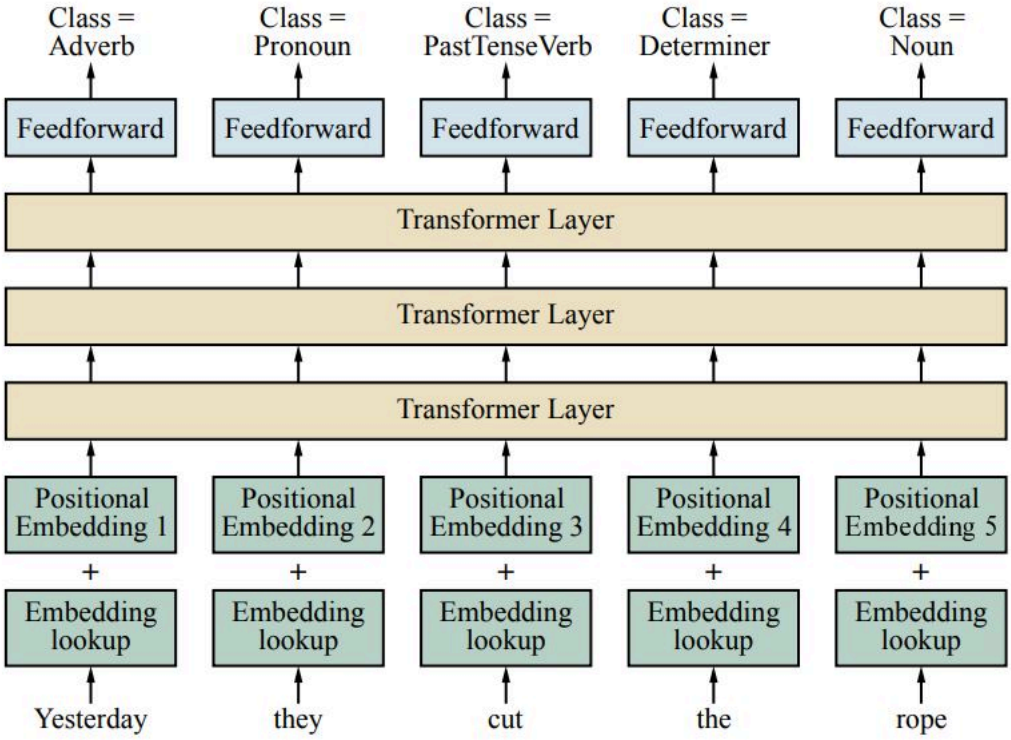


Imatge: *Un transformer d'una sola capa conté auto-atenció, una xarxa feedforward i connexions residuals.*

A la pràctica, els transformers solen tenir sis o més capes. Com en la resta de models, la sortida de la capa i s'usa com a entrada de la capa $i+1$.

L'arquitectura transformer no captura explícitament l'ordre de les paraules a la frase, ja que el context es modelitza només a través de l'autoatenció, que és agnòstica a l'ordre de les paraules. Per capturar l'ordenació de les paraules, el transformer usa una tècnica anomenada **embedding posicional**. Si la nostra oració té una longitud màxima de n, s'aprenen n vectors d'embedding nous, un per a cada posició. L'entrada al primer transformer és la suma de l'embedding de la paraula a la posició t més l'embedding posicional que correspon a la posició t.

La figura següent il·lustra l'arquitectura transformer aplicada a POS tagging.



Imatge: *Ús de l'arquitectura transformer per a POS tagging.*

A la part de baix, el *word embedding* i el *positional embedding* se sumen per formar l'entrada d'un transformer de tres capes. El transformer produeix un vector per paraula, com al POS tagging basat en RNN. Cada vector s'introdueix a una capa de sortida final i una capa softmax per produir una distribució de probabilitat sobre les etiquetes.

En aquesta secció, realment només hem explicat la meitat del transformer: el model que hem descrit aquí és el transformer codificador. És útil per a tasques de classificació de text. L'arquitectura transformer completa es va dissenyar originalment com un model de seqüència a seqüència per a la traducció automàtica. Per tant, a més del codificador, també inclou un transformer descodificador. El codificador i el descodificador són gairebé idèntics, llevat que els descodificador utilitza una versió de l'autoatenció en què cada paraula només atén a les paraules anteriors, ja que el text es genera d'esquerra a dreta. El descodificador també té un segon mòdul d'atenció en cada capa del transformer que atén a la sortida del codificador del transformer.

6. Pretraining i transfer learning

Aconseguir prou dades per construir un model robust pot ser un repte. En visió per computadora, aquest repte es va adreçar recollint grans col·leccions d'imatges, com ImageNet, i etiquetant-les a mà.

En llenguatge natural, se sol treballar a partir de text no etiquetat. La diferència ve en part de la dificultat d'etiquetar. Un treballador sense formació especialitzada pot etiquetar fàcilment una imatge com un moix o una posta de sol, però cal formació específica per anotar una frase amb etiquetes lèxiques o sintàctiques. La diferència ve també de l'abundància de text: Internet afegeix cent mil milions de paraules de text cada dia, incloent-hi llibres digitalitzats, recursos curats com la wikipèdia i posts de mitjans socials sense supervisió.

Projectes com Common Crawl donen accés fàcil a aquestes dades. Qualsevol text es pot usar per construir models n-gram o word embedding, i una part del text ve acompanyat d'una estructura que pot ser útil per a una varietat de tasques. Per exemple, hi ha molts de llocs de FAQs amb parells pregunta-resposta que es poden fer servir per entrenar sistemes de resposta de preguntes. Molts de llocs web publiquen traduccions paral·leles de textos, que poden servir per entrenar sistemes de traducció automàtica. També hi ha textos que venen etiquetats, com els dels llocs de revisió amb un sistema de valoració de cinc estrelles.

Per no haver de generar un nou conjunt de dades cada vegada que desenvolupem una aplicació de NLP, se sol treballar amb models que parteixen d'un **preentrenament** (*pretraining*), una forma d'**aprenentatge per transferència** (*transfer learning*). Una gran quantitat de dades lingüístiques generals s'usen per entrenar una versió inicial del sistema d'NLP. A partir d'aquí, es pot usar una quantitat més petita de dades específíca del domini (possiblement incloent-hi dades etiquetades) per refinar el model (*fine-tuning*). El model refinat pot aprendre el vocabulari, expressions, estructures sintàctiques i d'altres fenòmens lingüístics especifics del nou domini.

6.1. Word embeddings preentrenats

Al primer apartat d'aquest lliurament hem introduït els *word embeddings*. Vàrem veure que paraules semblants acaben amb representacions vectorials semblants, que permeten de resoldre problemes d'analogia mitjançant subtracció de vectors. Això indica que els *word embeddings* capturen molta d'informació sobre les paraules.

En aquesta secció abordarem com es creen els *word embeddings* amb un procés completament no supervisat a partir d'un gran corpus de text.

Ens centrarem en un model concret de word embeddings, el model **GloVe** (**Glo**bal **Ve**ctors). El model comença recollint recomptes de quantes vegades una paraula apareix dins una finestra entorn d'una altra paraula. Primer es tria la mida de la finestra (potser 5 paraules), es considera X_{ij} el nombre de vegades que les paraules i i j co-ocorren dins una finestra i X_i el nombre de vegades que la paraula i co-ocorre amb qualsevol altra paraula. P_{ij} serà la probabilitat que la paraula j aparegui en el context de la paraula i. E_i serà el *word embedding* de la paraula i.

Una part de la idea sota el model GloVe és que la relació entre dues paraules es pot capturar bé comparant-les a d'altres paraules. Considerem les paraules *gel* i *vapor*, i la raó de les probabilitats de coocurrència amb una altra paraula w:

$$\frac{P_{w,ice}}{P_{w,steam}}$$

Quan w sigui la paraula *sòlid* la raó serà alta (vol dir que *sòlid* s'aplica més a *gel*) i quan w sigui la paraula *gas* la raó serà baixa (vol dir que *gas* s'aplica més a *vapor*). Quan w sigui una paraula sense contingut, com ara el, una paraula com $aigua$ que és igual de rellevant per als dos termes, o una paraula irrellevant per als dos, com per exemple $moda$, la ràtio serà propera a 1.

El model GloVe comença amb aquestes consideracions i a través de raonaments matemàtics que converteixen raons de probabilitats en diferències de vectors i productes escalars arriba a la restricció

$$E_i \cdot E'_k = \log(P_{ij})$$

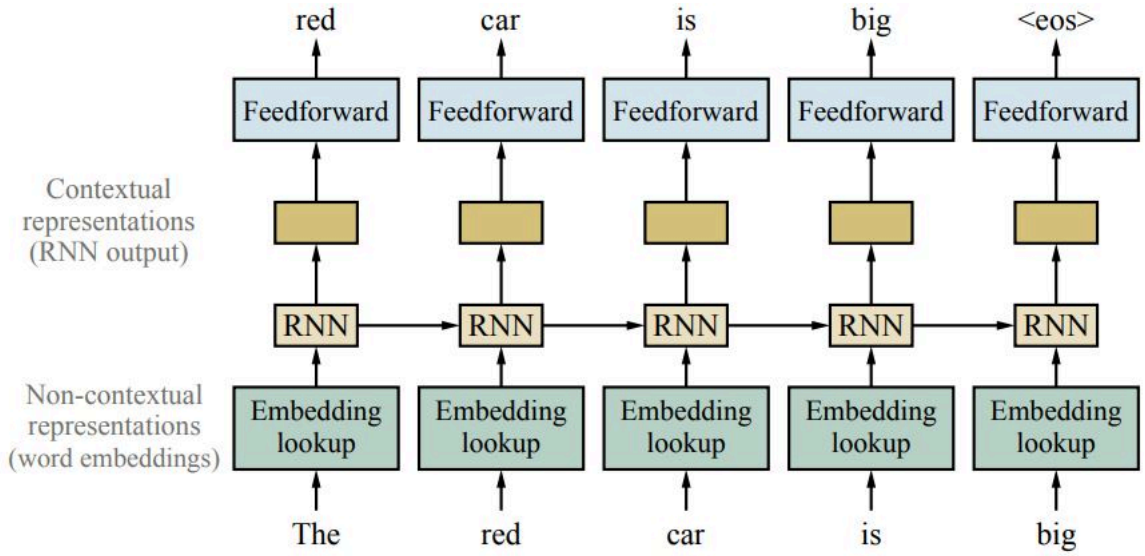
Amb altres paraules, el producte escalar de dos vectors de paraula és igual al logaritme de la probabilitat de la seva coocurrència. Això té sentit intuïtivament: dos vectors gairebé ortogonals tenen un producte escalar proper a 0, i dos vectors normalitzats gairebé idèntics tenen un producte escalar proper a 1. Hi ha una complicació tècnica que fa que el model GloVe hagi de crear dos vectors d'embedding per a cada paraula, E_i i E'_i. Calcular-los i llavors sumar-los al final ajudar a limitar el sobreajust.

Entrenar un model com GloVe és molt menys costós que entrenar una xarxa neuronal estàndard: es pot entrenar un nou model a partir de milers de milions de paraules en algunes hores amb una CPU de sobretaula estàndard.

6.2. Representacions contextuals preentrenades

Els word embeddings són representacions més bones que no els tokens de paraules individuals, però hem d'encarar el problema de la polisèmia. Per exemple, la paraula *porta* tant pot ser un substantiu com una forma del verb *portar*. Per tant, podem esperar clústers ben diferents de contextos: un de semblant a substantius com *finestra*, *pany*, o *clau*, i un altre semblant a formes verbals com *du*, *mena*, *trasllada* o *transporta*. *Porta* és un exemple clar d'una paraula amb dos significats distints, però altres paraules tenen matisos de significat segons el context. Les frases fetes també s'entenen més bé en conjunt que com a paraules individuals.

Per això, en lloc de simplement aprendre una taula que transformi cada paraula en el seu embedding, interessa entrenar un model per generar **representacions contextuals** de cada paraula en una frase. Una representació contextual transforma una paraula i el seu context en un vector de *word embedding*.



Imatge: *Entrenament de representacions contextuals amb un model de llenguatge d'esquerra a dreta*

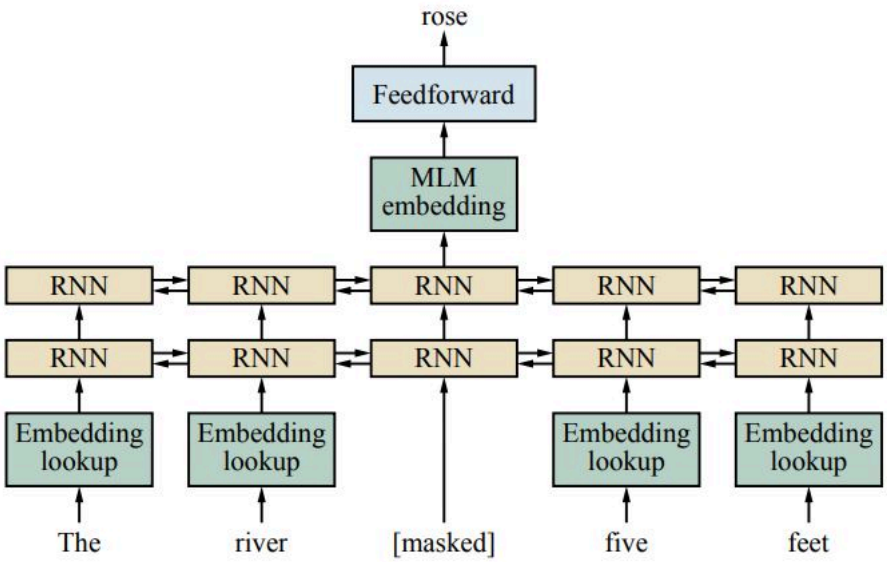
6.3. Models de llenguatge emmascarats

Una feblesa dels models estàndard de llenguatge com ara els n-gram és que la contextualització de cada paraula només depèn de les paraules anteriors de la frase. Les prediccions es realitzen d'esquerra a dreta. Però de vegades, el context posterior a l'oració ajuda a clarificar paraules anteriors.

Per això, podem usar models de llenguatge emmascarats (*Masked Language Model*, **MLM**). Els MLM s'entrenen ocultant paraules individuals a l'entrada i demanant al model que predigui la paraula amagada. Per aconseguir-ho, es poden fer servir tant RNN profundes bidireccionals com transformers sobre l'oració emmascarada.

L'elegància d'aquesta aproximació és que no necessita dades etiquetades: la frase dona la seva pròpia etiqueta per a la paraula amagada.

Si aquests models s'entrenen damunt un gran corpus de text, genera representacions preentrenades que ofereixen bons resultats en una àmplia gamma de tasques de NLP (traducció automàtica, resposta de preguntes, resum, valoracions de gramaticalitat, entre d'altres).



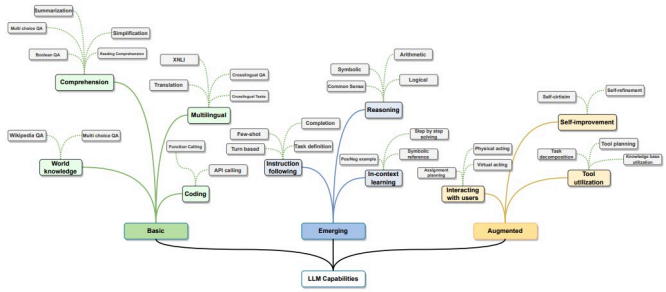
Imatge: *Models de llenguatge emmascarats*

7. Tècniques usades als LLM

Vegem amb més detall algunes característiques dels darrers models extensos de llenguatge, les tècniques d'entrenament i refinament que s'hi apliquen, així com les seves extensions i una classificació d'acord amb diversos criteris.

Tota aquesta informació la tenim a l'article del 2024 Large Language Models: A Survey.

7.1. Capacitats



Els models extensos de llenguatge tenen aplicació en un ventall ampli de camps.

Observem la branca de **raonament** (*reasoning*). Inclou els elements **lògic** (sil·logismes), **aritmètic**, **simbòlic**, i el gran repte no resolt d'incloure-hi el **sentit comú**.

Dins aquest arbre, el node que té potencial per a l'aparició de superintel·ligència, que s'expandís il·limitadament, és el de l'**automillorament** (*self-improvement*), amb els elements d'autocrítica (*self-criticism*) i refinament iteratiu (*self-refinement*).

Aquestes habilitats emergents inclouen:

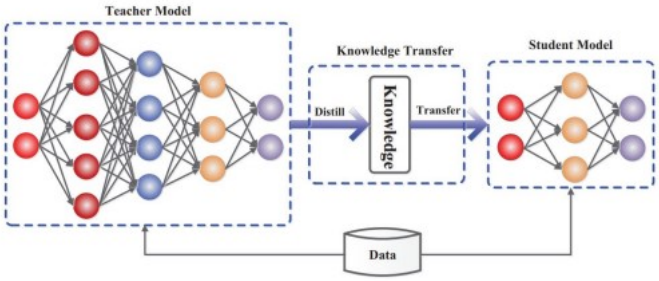
1. Aprenentatge en context, on els LLM aprenen una tasca nova a partir d'un petit conjunt d'exemples presentats a l'indicador en el moment de la inferència,
2. Seguiment d'instruccions, on els LLM poden realitzar les indicacions per a nous tipus de tasques sense utilitzar exemples explícits
3. Raonament en diversos passos, on els LLM poden resoldre una tasca complexa desglossant aquesta tasca en passos de raonament mitjançant la cadena de pensament (Chain of Thought)

Els LLM també es poden augmentar mitjançant l'ús de coneixements i eines externes, perquè puguin interactuar eficaçment amb els usuaris i l'entorn, i millorar contínuament mitjançant les dades de retroalimentació recollides a través d'interaccions, per exemple, mitjançant l'aprenentatge de reforç amb feedback humà (RLHF).

Mitjançant l'ús avançat i les tècniques d'augment, els LLM es poden desplegar com els anomenats agents d'IA: entitats artificials que perceben el seu entorn, prenen decisions i executen accions.

Fins fa poc, les investigacions s'han centrat a desenvolupar agents per a tasques i dominis específics. Les habilitats emergents que demostren els LLM permeten construir agents d'IA de propòsit general basats en LLM. Tot i que els LLM estan entrenats per produir respostes en entorns estàtics, els agents d'IA han de prendre accions per interactuar amb l'entorn dinàmic. Per tant, els agents basats en LLM sovint necessiten augmentar els LLM per obtenir informació actualitzada de bases de coneixement externes, verificar si una acció del sistema produeix el resultat esperat o fer front a la situació quan les coses no surten com s'esperava.

7.2. Destil·lació



La destil·lació (*distillation* en anglès) consisteix en l'entrenament d'un model a partir de la interacció amb un model més gran. El model petit es considera un estudiant i el model més gran el seu mestre.

El coneixement destil·lat del model gran es transfereix al model petit.

La destil·lació del coneixement és el procés d'aprenentatge d'un model més gran [143]. Els primers dies de llançament de models de millor rendiment han demostrat que aquest enfocament és molt útil fins i tot si s'utilitza en un enfocament de destil·lació API.

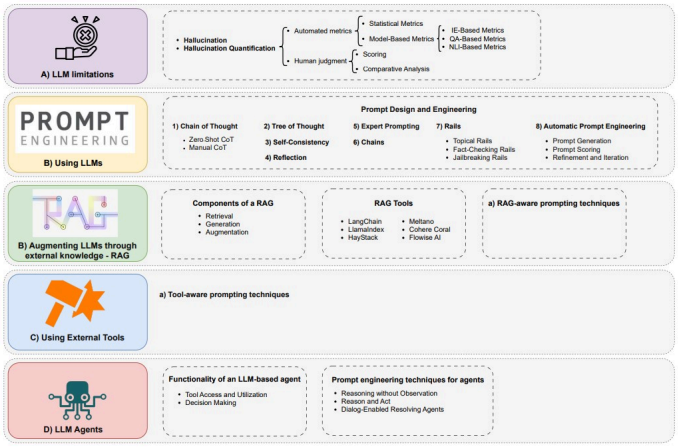
També es coneix com un enfocament per destil·lar el coneixement no d'un sol model, sinó de múltiples models en un de més petit. La creació de models més petits amb aquest enfocament produeix mides de models més petites que es poden utilitzar fins i tot en dispositius finals. La destil·lació del coneixement, tal com es mostra a la figura, il·lustra una configuració general d'aquest esquema d'entrenament.

El coneixement es pot transferir mitjançant diferents formes d'aprenentatge: destil·lació de resposta, destil·lació de característiques i destil·lació API. La destil·lació de la resposta només s'ocupa dels resultats del model del professor i intenta ensenyar al model de l'estudiant com actuar exactament o almenys de manera similar (en el sentit de predicció) com el professor. La destil·lació de característiques no només utilitza l'última capa, sinó també capes intermèdies per crear una millor representació interna del model d'estudiant. Això ajuda al model més petit a tenir una representació similar al model del professor.

La destil·lació d'API és el procés d'utilitzar una API (normalment d'un proveïdor de LLM com OpenAI) per entrenar models més petits. En el cas dels LLM, s'utilitza per entrenar el model a partir de la sortida directa del model més gran, cosa que el fa molt similar a la destil·lació de resposta. Aquest tipus de destil·lació suscita moltes preocupacions perquè en els casos en què el model en si no està disponible obertament, s'exposa una API de pagament (normalment) per als usuaris finals. D'altra banda, mentre els usuaris paguen per cada trucada, la manera d'utilitzar les prediccions és limitada, per exemple, OpenAI prohibeix l'ús de la seva API per crear LLM que més tard s'utilitzaran per competir amb ella. El principal valor en aquest cas són les dades de formació.

Se sospita que DeepSeek pot haver fet ús de la destil·lació a partir d'algun model d'OpenAI per aconseguir els seus resultats impressionants.

7.3. Ús i augmentació



La principal limitació dels LLM són les anomenades al·lucinacions. Aquests models tenen tendència a respondre sempre, sense distingir si el text que produeixen correspon a una informació factual acurada o no. A més, pel fet que redacten amb una competència molt notable, les seves respostes fan sensació d'autoritat, i poden induir fàcilment una reacció humana de credibilitat.

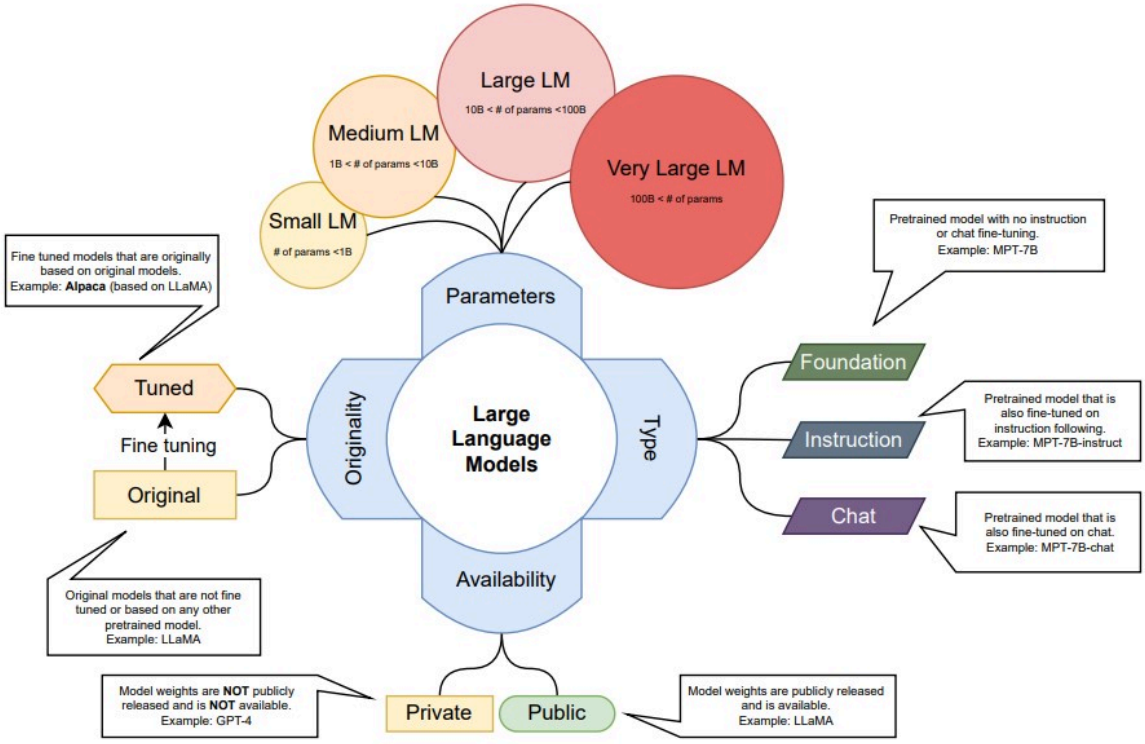
Per obtenir les respostes més ajustades als nostres interessos, podem utilitzar una varietat de tècniques d'indicació o *prompting*. Les cadenes de raonament (*Chain of Thought*, CoT) organitzen la seqüència del pensament d'una forma comprensible per a nosaltres. A més de cadenes lineals, també s'utilitzen arbres (*Tree of Thought*) i grafs (*Graph of Thought*), que permeten un desplegament més complex de les opcions de deducció de conclusions.

Es pot augmentar la capacitat dels LLM combinant-los amb fonts externes de coneixement, en el que s'anomena *Retrieval Augmented Generation*. D'aquesta forma, les respostes en llenguatge natural oferides pel model de llenguatge es basen en informació contrastada i s'evita el problema de les al·lucinacions.

Moltes vegades la utilització d'eines externes dona informació precisa d'interès per a l'usuari, que convindrà redactar amb l'LLM. Aquestes eines externes poden ser des de calculadores fins a navegadors web que permetin accedir a una infinitat de serveis d'obtenció d'informació: meteorològica, de disponibilitat de productes...

Més amunt en l'escala d'autonomia, els agents basats en LLM poden realitzar seqüències d'accions complexes de forma autònoma, cosa que els dona una gran utilitat alhora que planteja riscos importants de seguretat. La utilització de la computadora per part d'un agent és un procediment vulnerable a intrusions i filtració d'informació confidencial.

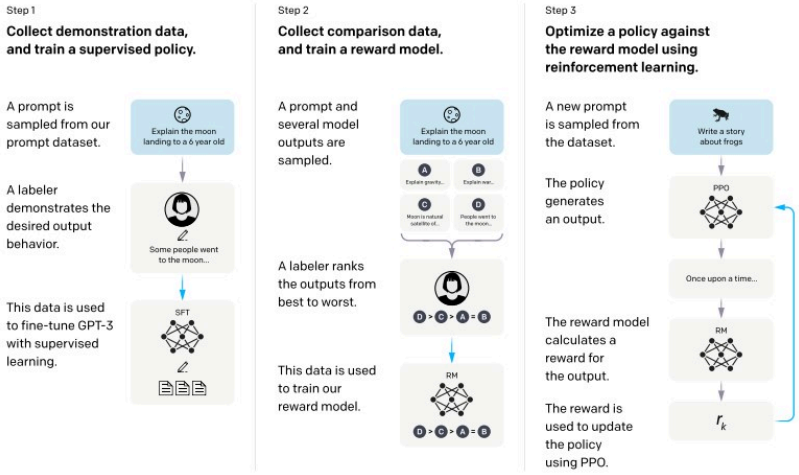
7.4. Categorització



Podem classificar els LLM d'acord amb diversos criteris: nombre de paràmetres, tipus, disponibilitat i originalitat.

- Segons la seva **mida**: des dels petits, per sota de 1000 milions de paràmetres, fins als molt grans, per damunt de 100000 milions de paràmetres.
- Segons el seu **tipus**: fundacional, instruccional o de xat.
- Segons la seva **disponibilitat**: públics, amb els pesos publicats com LlaMA o privats, amb pesos no publicats ni disponibles, com GPT-4.
- Segons la seva **originalitat**: si el model és directament el que s'ha obtingut després d'entrenar (original) o si s'ha refinat per aplicar a una tasca particular (*fine-tuned*).

7.5. RLHF



La figura il·lustra un sistema d'aprenentatge per reforç a través de retroacció humana (RHLF), en tres etapes:

1. A partir de les dades recollides, s'entrena una política supervisada.
2. Es recullen diferents respostes a un *prompt*, un etiquetador humà les ordena i se n'obté un model de recompensa.
3. S'optimitza una política contra el model de recompensa usant aprenentatge per reforç

Mitjançant RHLF, es pot ajustar el comportament dels MLE de forma que les seves produccions s'ajustin a les preferències humanes.

Training language models to follow instructions with human feedback

8. Casos d'ús

En aquesta secció oferim una selecció d'exemples de codi per aplicar Transformers a diverses aplicacions.

- Classificació de text
- Question Answering with Hugging Face Transformers
- Traducció amb Transformer
- Named Entity Recognition

8.1. Character-level recurrent sequence-to-sequence model

Aquest exemple mostra com implementar un model bàsic de seqüència a seqüència recurrent a nivell de caràcter. S'aplica a la traducció de frases breus en anglès a frases curtes en francès, caràcter a caràcter. Hem de tenir en compte que és força inusual fer traducció automàtica a nivell de caràcters, ja que els models a nivell de paraula són més habituals en aquest domini.

Resum de l'algorisme

Comencem amb seqüències d'entrada d'un domini (per exemple, frases en anglès) i les seqüències de destinació corresponents d'un altre domini (per exemple, frases en francès).

Un codificador LSTM converteix les seqüències d'entrada en 2 vectors d'estat (mantenim l'últim estat LSTM i descartem les sortides).

Un descodificador LSTM està entrenat per convertir les seqüències objectiu en la mateixa seqüència però compensada per un pas de temps en el futur, un procés d'entrenament anomenat "forçament del professor" en aquest context. Utilitza com a estat inicial els vectors d'estat del codificador. Efectivament, el descodificador aprèn a generar objectiu[t+1...] donat objectiu[...t], condicionat a la seqüència d'entrada.

En mode d'inferència, quan volem descodificar seqüències d'entrada desconegudes, seguim les passes següents:

- Codificar la seqüència d'entrada en vectors d'estat
- Començar amb una seqüència objectiu de mida 1 (només el caràcter d'inici de la seqüència)
- Introduir els vectors d'estat i la seqüència objectiu d'1 caràcter al descodificador per produir prediccions per al següent caràcter
- Mostrar el caràcter següent utilitzant aquestes prediccions (només utilitzant argmax).
- Afegir el caràcter mostrat a la seqüència objectiu.
- Repetir fins que generem el caràcter de final de seqüència o bé arribem al límit de caràcters.

https://keras.io/examples/nlp/lstm_seq2seq/

8.2. Classificació de text

Els transformers es poden usar per classificar text, com per exemple en valoracions positives o negatives de pel·lícules, com a la base de dades Imdb en aquest cas.

L'exemple té els blocs següents:

- Implementació d'un bloc *transformer* com a capa (*layer*)
- Implementació de la capa d'*embedding*
- Descàrrega i preparació de les dades d'Imdb.
- Creació del model classificador usant la capa *transformer*
- Entrenament i avaluació final

Classificació de text

8.3. Question Answering With Hugging Face Transformers

La resposta a preguntes és una tasca habitual de PNL amb diverses variants. En algunes variants, la tasca és d'elecció múltiple: s'ofereix una llista de possibles respostes amb cada pregunta i el model simplement ha de retornar una distribució de probabilitat sobre les opcions. Una variant més difícil de respondre a preguntes, que és més aplicable a les tasques de la vida real, es dona quan no es proporcionen les opcions. En lloc d'això, el model rep un document d'entrada (anomenat context) i una pregunta sobre el document, i ha d'extreure l'abast de text del document que conté la resposta. En aquest cas, el model no calcula una distribució de probabilitat sobre respostes, sinó dues distribucions de probabilitat sobre les fitxes del text del document, que representen l'inici i el final de l'abast que conté la resposta. Aquesta variant s'anomena "resposta extractiva de preguntes".

La resposta extractiva de preguntes és una tasca de PNL molt difícil, i la mida del conjunt de dades necessària per entrenar aquest model des de zero quan les preguntes i les respostes són en llenguatge natural és prohibitivament enorme. Com a resultat, la resposta a preguntes (com gairebé totes les tasques de PNL) es beneficia enormement de partir d'un model de base sòlid pre-entrenat: partir d'un model de llenguatge pre-entrenat fort pot reduir la mida del conjunt de dades necessària per assolir una precisió determinada en diversos ordres de magnitud, cosa que permetrà assolir un rendiment molt fort amb conjunts de dades sorprenentment raonables.

Començar amb un model preentrenat afegeix dificultats, però, d'on treu el model? Com us assegureu que les vostres dades d'entrada es processen prèviament i es representen de la mateixa manera que el model original? Com modifiqueu el model per afegir un bloc de sortida que coincideixi amb la vostra tasca d'interès?

En aquest exemple, veurem com carregar un model de la biblioteca Hugging Face Transformers per afrontar aquest repte. També carregarem un conjunt de dades de resposta a preguntes de referència de la biblioteca Datasets: aquest és un altre dipòsit de codi obert que conté una àmplia gamma de conjunts de dades en moltes modalitats, des de la PNL fins a la visió i més enllà. Tanmateix, no hi ha cap requisit que aquestes biblioteques s'hagin d'utilitzar juntes. Si volem entrenar un model de Transformers amb les nostres pròpies dades, o volem carregar dades des de Datasets i entrenar-hi els nostres propis models totalment no relacionats, això és possible.

https://keras.io/examples/nlp/question_answering

8.4. Traducció amb Transformer

En aquest exemple, es construeix un model de transformador seqüència a seqüència, que entrenarem en una tasca de traducció automàtica de l'anglès a l'espanyol.

Veurem com:

- Vectoritzar el text amb la capa Keras TextVectorization.
- Implementar una capa TransformerEncoder, una capa TransformerDecoder i una capa PositionalEmbedding.
- Preparar dades per entrenar un model de seqüència a seqüència.
- Utilitzar el model entrenat per generar traduccions de frases d'entrada mai vistes (inferència de seqüència a seqüència).

https://keras.io/examples/nlp/neural_machine_translation_with_transformer/

8.5. Named Entity Recognition

El reconeixement d'entitats amb nom (NER, Named Entity Recognition) és el procés d'identificació d'entitats amb nom al text. Alguns exemple d'entitats anomenades són: "Persona", "Ubicació", "Organització", "Dates", etc. NER és essencialment una tasca de classificació de tokens on cada token es classifica en una o més categories predeterminades.

En aquest exercici, entrenarem un model senzill basat en Transformer per realitzar NER. Utilitzarem les dades de la tasca compartida de CoNLL 2003, presa de la biblioteca de conjunts de dades de HuggingFace, que conté una versió processada d'aquest conjunt de dades.

https://keras.io/examples/nlp/ner_transformers/

9. Estat de l'art

L'aprenentatge profund i l'aprenentatge per transferència han avançat l'estat de l'art de l'NLP d'una forma espectacular. L'impuls principal d'aquest punt de no retorn és l'observació que l'aprenentatge per transferència funciona bé en els problemes de llenguatge natural: un model de llenguatge general es pot descarregar i afinar (*fine-tuning*) per a una tasca específica.

1. **What will best separate a mixture of iron filings and black pepper?**
   (a) magnet (b) filter paper (c) triple beam balance (d) voltmeter
2. **Which form of energy is produced when a rubber band vibrates?**
   (a) chemical (b) light (c) electrical (d) sound
3. **Because copper is a metal, it is**
   (a) liquid at room temperature (b) nonreactive with other substances
   (c) a poor conductor of electricity (d) a good conductor of heat
4. **Which process in an apple tree primarily results from cell division?**
   (a) growth (b) photosynthesis (c) gas exchange (d) waste removal

Imatge: *Qüestionari d'un examen de ciències que el sistema ARISTO pot respondre correctament usant un conjunt de mètodes, el més important un model de llenguatge RoBERTa. Respondre aquestes qüestions necessita coneixement sobre el llenguatge natural, l'estructura dels tests d'elecció múltiple, sentit comú i ciència.*

It is a truth universally acknowledged that the earth is not the center of the universe. There are those who assert there is. I do not accept them, but others I consider to be of the same opinion. The truth is, however, that if there are other than the center, and if there are any other living things in the universe and if they are not human, then we do not yet have our answers. We have to go on. This page gives a simplified, simplified answer to the problem. We don't have all the answers. The truth is, however, that the truth is out there.

When Gregor Samsa woke up one morning, he did not notice anything strange. "When my wife is looking at me, I feel like she is looking at a piece of art," he said. "I think she is admiring something I have created." The idea is that by looking at your own life, you learn something important and become a better person. It is a theory that emerged from psychologist Daniel Goleman's work, in which he asked "How do you know you're not a loser?"

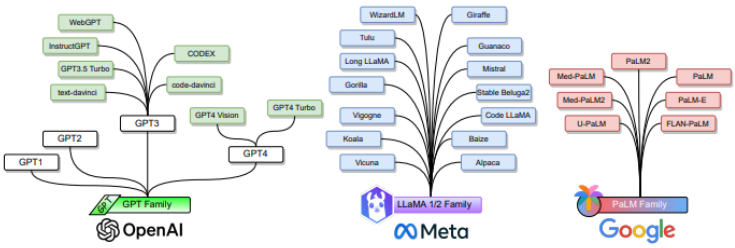
Alice was beginning to get very tired of sitting with her sister on the bank. She sat up, yawned, and said, with a loud little scream, "I hope you don't mind if I keep on doing what I should like to do, and if someone asks me which of us will do more, don't tell them that I won't do much, my dear sister."

All happy families are alike; each happy family is like a garden of paradise. The only difference between happy families and unhappy families, is that the unhappy family doesn't have any flowers or trees.

Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Please fill out the following details. Thank you... Thank you for your interest in this interview. Please wait...

Imatge: *Compleció de textos generada pel model GPT-2, donats els prompts en negreta. La major part dels textos són prou fluïds, almanco localment. El darrer paràgraf mostra que el model de vegades fa figa.*

Els models grans de llenguatge estan en expansió constant. En la següent imatge hi veim algunes de les branques més representatives.



Imatge: *Famílies de LLM. Font: Large Language Models: A Survey (feb 2024)*

Ens podem preguntar per què hem estudidat gramàtiques, anàlisi sintàctica i interpretació semàntica en el lliurament anterior, simplement per ignorar-ho ara en favor dels models basats purament en dades? Ara mateix, la resposta és simplement que els models basats en dades són més fàcils de desenvolupar i mantenir, i puntuen més bé en ens tests estàndard, comparats amb els sistemes construïts a mà que es poden construir usant una quantitat raonable d'esforç humà. Pot ser que els models transformer i els seus parents estiguin aprenent representacions latents que capturen les mateixes idees bàsiques que les gramàtiques i la informació semàntica; o pot ser que estigui passant qualque cosa totalment diferent dins aquests models; simplement no ho sabem, com expliquen Russell i Norvig al llibre Artificial Intelligence: A Modern Approach. Sabem que un sistema que s'entrena amb dades de text és més fàcil de mantenir i d'adaptar a nous dominis i nous llenguatges que un sistema basat en característiques definides a mà.

També pot ser que avenços futurs en modelització gramàtica i semàntica facin oscil·lar el pèndol en sentit contrari. Potser és més probable l'emergència de sistemes híbrids que combinin les millors característiques dels dos enfocaments.

És clar que hi ha espai de millora: no només els sistemes de NLP encara són per darrere la precisió de les persones en moltes tasques, sinó que aconsegueixen el seu bon funcionament després de processar milers de vegades el text que una persona pot llegir en tota la seva vida. Això suggereix que hi ha molt de camp per a aportacions des de la lingüística, la psicologia i la recerca en NLP.

10. Sistemes disponibles

A més de les grans empreses tecnològiques, altres companyies més petites treballenen grans models de llenguatge. Aquí en presentam una llista, sempre subjecta a noves incorporacions i substitucions de sistemes per altres versions que els superen.

- ChatGPT d'OpenAI
- LlaMa de Meta (matriu de Facebook)
- Gemini de Google
- Claude d'Anthropic
- Coral de Cohere
- Mistral

En català podem provar el sistema FLOR, derivat de BLOOM dins l'àmbit del Projecte Aina.

A la web ~~chat.lmsys.org~~ lmarena.ai podem contribuir a comparar parells de models de llenguatge extensos (MLE, en anglès *LLM*).

A la imatge següent oferim una vista del rànquing obtingut el juny de 2024.

Rank* (UB)	Model	Arena Elo	95% CI	Votes	Organization
1	GPT-4o-2024-05-13	1287	+4/-4	32181	OpenAI
2	Gemini-1.5-Pro-API-0514	1267	+5/-4	25519	Google
2	Gemini-Advanced-0514	1266	+5/-5	27225	Google
4	Gemini-1.5-Pro-API-0409-Preview	1257	+3/-3	55731	Google
4	GPT-4-Turbo-2024-04-09	1256	+2/-2	59891	OpenAI
5	GPT-4-1106-preview	1251	+2/-3	80067	OpenAI
6	Claude 3 Opus	1248	+2/-2	123645	Anthropic
6	GPT-4-0125-preview	1246	+3/-2	73286	OpenAI
9	Yi-Large-preview	1239	+4/-3	34567	01 AI
9	Gemini-1.5-Flash-API-0514	1232	+4/-4	23797	Google
11	Bard (Gemini Pro)	1208	+7/-5	11853	Google
11	Llama-3-70b-Instruct	1208	+3/-2	124645	Meta
12	Claude 3 Sonnet	1201	+3/-2	96209	Anthropic

Imatge: *Rànquing de MLE obtingut per valoració humana col·lectiva. Font: lmsys.org*

10.1. Empreses líder

Quan investigadors de Google publicaren el famós article anomenat dels transformers (**Attention Is All You Need**), ni ells mateixos no varen ser conscients de l'impacte que aconseguiria la seva publicació.

Qui sí que va veure ràpidament el potencial d'aquesta nova arquitectura va ser **Ilya Sutskever**, un dels fundadors d'**OpenAI** i en aquell moment el seu CTO. Sutskever va comanar a **Alec Radford** la primera implementació de transformers generatius pre-entrenats (Generative Pretrained Transformers), que va posar les bases de l'actual ChatGPT i el seu gran impacte el novembre de 2022.

Molt abans de ChatGPT, **Google** feia temps que treballava en sistemes de xarxes neuronals aplicades al processament del llenguatge natural. Bard, LaMDA o T5 són exemples dels esforços de l'empresa en aquesta àrea.

Tanmateix, després dels avanços accelerats d'OpenAI en l'aplicació dels transformers a productes d'impacte massiu, Google s'ha vist obligat a reaccionar ràpidament i posar-se al màxim nivell mundial. El seu darrer model, no només de text sinó també multimodal, és **Gemini**.

Dins Facebook, que forma part de **Meta**, la divisió d'IA, liderada per **Yann LeCun**, portava el nom de FAIR (Facebook Artificial Intelligence Research), ara Meta AI. Destaca la seva aposta pel codi obert amb sistemes com LlaMA.

Dins Europa, l'empresa més destacada en models de llenguatge extensos és la francesa **Mistral**. Han tengut èxit en la implantació de models MoE (Mixture of Experts), entre els quals Mixtral. Juntament amb HuggingFace, formen part d'un ecosistema francès capdavanter en IA.

Un grup d'investigadors que va sortir de Google, crítics amb la manca de mesures de seguretat en IA responsable de l'empresa, varen constituir **Anthropic,** liderada per **Dario Amodei**.

Destaquen pel sistema **Claude**. Té característiques interessants com els recents *artifacts*, que permeten obtenir molt ràpidament esbossos de pàgines web, per exemple, simplement a partir de prompts o indicacions que es poden iterar algunes vegades per anar aconseguint un resultat més afinat i ajustat a la intenció del dissenyador.

Un servei interessant que integra la cerca a la web amb diversos MLE subjacents és **Perplexity**. El seu mode de funcionament, citant les fonts que utilitza, ha estat després imitat per altres sistemes d'empreses més grans.

DeepSeek és una empresa xinesa d'intel·ligència artificial que ha desenvolupat models competitius amb els líders del sector, com OpenAI i Google. Utilitzant pesos de codi obert de LLaMA i tècniques d'optimització avançades, han aconseguit maximitzar l'eficiència de les GPU de Nvidia disponibles, malgrat les restriccions d'exportació dels EUA a la Xina. El seu model R1, conegut per les seves capacitats avançades de raonament, ha tingut un impacte significatiu en el mercat, provocant una caiguda del 17% en el valor de les accions de Nvidia en un sol dia.

- https://www.vilaweb.cat/noticies/deepseek-quan-la-intelligencia-artificial-comenca-a-raonar/

11. Estratègia d'Intel·ligència Artificial 2024

Com llegim al pròleg de l'estratègia espanyola d'IA del 2024, "la Intel·ligència Artificial representa una de les revolucions més transcendentals dels darrers temps, que per les seves característiques té la capacitat d'abordar els desafiaments més complexos del món contemporani. Aquesta revolució tecnològica, el desenvolupament i l'expansió s'està accelerant substancialment en molt pocs anys, promet ser el catalitzador de grans transformacions econòmiques i socials. Des d'una perspectiva econòmica, genera noves possibilitats, que les converteixen en el potencial impulsor d'un augment notable de productivitat i del creixement, amb un efecte que s'estén de manera transversal a un gran nombre de sectors econòmics. Vora aquests impactes, la IA té a més la capacitat d'influir en un ampli conjunt d'àmbits que van més enllà del purament tecnològic o econòmic, afectant també als comportaments i a les relacions humanes."

Per això, "a més dels seus beneficis evidents, la IA planteja inevitablement reptes i incerteses sobre els seus usos i les seves implicacions en totes les esferes de l'economia, la societat i les persones, només comparables al que s'ha esdevingut a les grans revolucions industrials al llarg de la història."

En aquest context, l'estratègia d'IA aposta pel desenvolupament d'un model de llenguatge fundacional per a les llengües oficials d'Espanya: castellà, català, gallec i basc.

La base fonamental per a l'entrenament dels models de llenguatge són la quantitat i la qualitat de les dades disponibles.

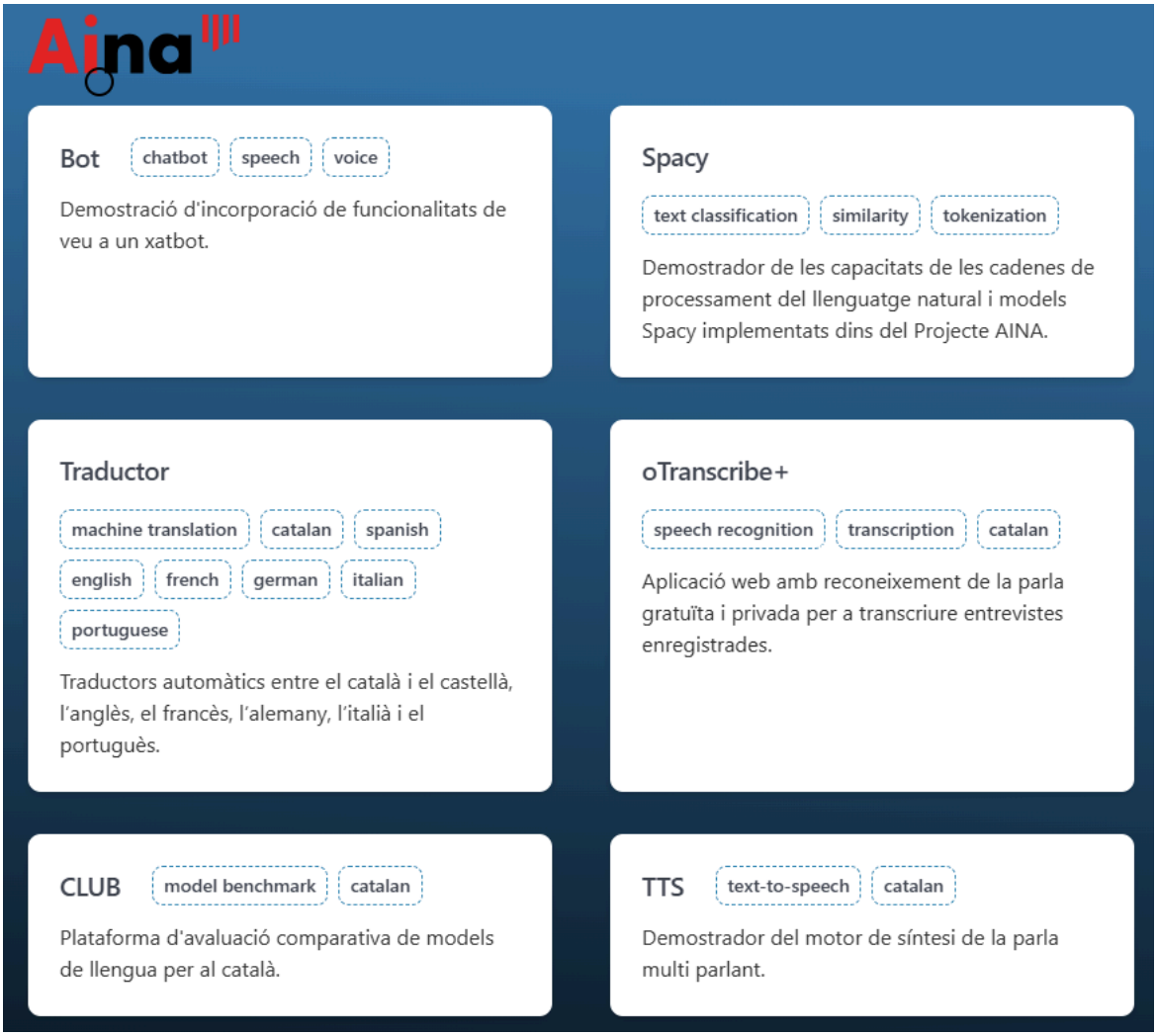
En aquest moment, comptam amb un corpus massiu multilingüe (36 llengües europees) amb un total de 10 trilions (americans, corresponen a 10 bilions) de tokens. A més, hi ha material amb més de 40 corpus anotats per a l'afinació i avaluació de models. També hi ha creats conjunts i instruccions en diverses llengües per a l'alineació de models.

Després d'obtenir les dades, es realitzaran tasques com la generació d'instruccions i anotació de dades. També cal desenvolupar conjunts d'avaluació, especialment per a models generatius, capaços d'avaluar el rendiment dels models i comparar-los.

- https://cervantes.org/es/sobre-nosotros/publicaciones/informe-estado-actual-corpus-espanol-lenguas-cooficiales-variantes

12. Projecte Aina

El Projecte Aina se centra en el desenvolupament de recursos de tecnologies del llenguatge i la parla per al català.



Combinant diversos d'aquests elements, Softcatalà ha fet públic un sistema de doblatge automàtic de vídeos de l'anglès o el castellà al català.

13. Incidents amb xatbots

Pel fet d'usar el llenguatge natural humà, la interacció de les eines d'IA basades en MLE, com els xatbots, poden generar reaccions intenses a les persones.

Per exemple, l'investigador de Google Blake Lemoine va declarar que pensava que el xatbot amb qui havia estat conversant durant una llarga temporada havia adquirit consciència. Aquesta afirmació va ser rebatuda per Google i Lemoine va ser acomiadat.

https://www.scientificamerican.com/article/google-engineer-claims-ai-chatbot-is-sentient-why-that-matters/

A Bèlgica, un home profundament preocupat pel canvi climàtic, n'havia estat conversant llargament amb un xatbot, amb qui establí una relació íntima. La darrera conversa que hi tengué pràcticament es pot interpretar com una invitació al suïcidi, que va ser el desenllaç de la relació.

https://www.euronews.com/next/2023/03/31/man-ends-his-life-after-an-ai-chatbot-encouraged-him-to-sacrifice-himself-to-stop-climate-

Un altre cas de suïcidi atribuït a un xatbot afectà un jove de catorze anys a Florida.

https://www.nytimes.com/2024/10/23/technology/characterai-lawsuit-teen-suicide.html

A més, hi ha hagut altres interaccions amb xatbots totalment fora de lloc, com per exemple una suggerència de matar els pares, feta a un adolescent.

https://www.bbc.com/news/articles/cd605e48q1vo

La gran quantitat de persones i la varietat de circumstàncies en què poden interactuar amb sistemes conversacionals fa imprescindible una cura extrema en els converses que s'hi puguin produir. Aquesta situació és especialment delicada quan els usuaris són menors d'edat, més vulnerables als missatges rebuts.

14. Per a saber-ne més

Acabam el lliurament amb una llista de recursos i noms clau en el camp de l'NLP.

- El transformer il·lustrat: https://jalammar.github.io/illustrated-transformer/
- Catàleg de transformers: https://amatriain.net/blog/transformer-models-an-introduction-and-catalog-2d1e9039f376/
- NLP a HuggingFace: https://huggingface.co/learn/nlp-course

Alguns investigadors destacats són els següents.

- **Jacob Devlin** i **Ming-Wei Chang** són els primers autors de l'article sobre el transformer BERT. També són els autors del capítol sobre Deep Learning aplicat al processament del llenguatge natural de la quarta edició del llibre Artificial Intelligence, A Modern Approach. Els apunts d'aquest lliurament segueixen de prop aquesta referència.
- **Ilya Sutskever** és cofundador d'OpenAI, un investigador fonamental en els GPT, entre ells ChatGPT.
- **Andrej Karpathy** va ser cofundador d'OpenAI i ara treballa a EurekaLabs.
- Chris Manning és un investigador de referència en l'àrea de NLP.

Cada un dels autors de l'article dels Transformer ha posat en marxa les seves pròpies iniciatives.

https://www.wired.com/story/eight-google-employees-invented-modern-ai-transformers-paper/

- Illia Polosukhin va fundar Near, una empresa de blockchain.
- **Niki Parmar** i Ashish Vaswani varen engegar **Adept AI**, i més endavant, la seva segona empresa, **Essential AI**.
- **Llion Jones** fundà Sakana AI.
- Noam Shazeer fa cofundar Character AI.
- Aidan Gomez fa cofundar Cohere.
- Jakob Uszkoreit va fundar la companyia de biotecnologia Inceptive.
- Lukasz Kaiser treballa a OpenAI.

Llevat de la primera, basada en *blockchain*, la resta d'empreses es basen en l'aplicació de la tecnologia dels *transformers*.

A més, l'equip dels vuit autors de l'article sobre Transformers ha rebut el premi NEC C&C el 2024.

https://www.nec.com/en/press/202410/images/1501-01-02.pdf