**Machine learning e NLP**

Dal punto di vista storico, il testo sono dati, come l’approccio al data mining, e abbiamo delle features che catturiamo o che abbiamo gia in origine sui dati e poi le diamo in pasto ad un modello di apprendimento che le riceve in input e quel modello lo usiamo per classificare.

**Features -> modello -> valutazione**

Prima si identificano le features poi si fa apprendimento e si crea un modello e infine si usa.

Tecniche di ML più utilizzate in origine in ambito NLP:

* **Basate su probabilità**, quindi teorema di bayes
* **Support vector machine**, all’interno di uno spazio multidimensionale, l’SVM cerca di identificare un iperpiano nello spazio multidimensionale che cerca di suddividere al meglio le sottoaree associate a 2 classi... L’output è trovare i vettori di supporto che servono per scrivere un iperpiano di separazione di istanze di un Dataset. Lavora su classificazione binaria ma ci sono meccanismi per farlo funzionare su sistema multiclasse.

Quindi tutto quello che si faceva era, dato un testo, individuare tutta una serie di features come features lessicali, morfologiche, n-grammi, dipendenze sintattiche, sinonimi... Poi si usava SVM e si guardavano le performance con precision, recall, F1.

* **HMM:** per task di POS, NER
* **Conditional random fields**: usato in tutti i task di sequence label: POS, NER. Usano lo stato precedente e uno storico più lungo
* **Reti neurali**

**Deep learning e NLP**

Le reti più importanti in ambito NLP sono:

* **RNN:** reti ricorrenti.
* **LSTM:** long short term memory. Si basa su un concetto semplice: mentre leggi, alcune cose sono più rilevanti di altre e te le puoi ricordare il resto lo dimentica. Apprende cosa dimenticare. Molte delle cose che si sono lette non le tiene in memoria, riesce a leggere enormi sequenze di testi. In output la rete ricorda sia le cose più recenti lette sia quelle passate. Funzionano bene con il testo.
* **Gated recuret unit.** GRU. E’ un approccio più semplice di LSTM. Semplifica l’architettura. Con pochi dati funziona al pari di LSTM, con molti dati è meglio LSTM
* **Reti convoluzionali:** sono spesso trattate male perchè è un preprocessing dei dati. Arrivano dal mondo della classificazione delle immagini ma funzionano bene anche sul testo.
* **Transformer:** Si basano su un’architettura encoder – decoder, si ha una prima parte di encoding dove un set di features viene mappato su una dimensione più piccola. E’ usa specie di LSA, cerca di comprimere il feature set in meno dimensioni, quindi esprimere la stessa semantica ma con meno dimensioni. Poi c’è la parte di decoding che presa questa seantoca collassata cerca di riportarla in uno spazio n-dimensionale. **BERT** è il capostipite. Invece di avere un approccio LSA, guarda tutte le features in una sola volta.

Problemi aperti:

* **Overfitting**: Un modello che funziona benissimo sui dati di train ma quando lo sposto su dati non ancora visti si comporta malissimo.
* **Few-Shot-Learning**: Il learning con pochissimi dati di esempio. C’è bisogno sempre di più di algoritmi che funzionano bene ma con pochi dati in input.
* **Domain-adaptation**: Ciò che si tira fuori dai modelli è gia quasi pronto ma bisogna modellarli in un dominio specifico, quindi ad esepio degli embedding appresi sul brown corpus non funzionano bene per fare calssificazione su Twit, o su testi medici. Si adattano degli embedding e si danno in pasto ad una rete che fa nuovi cicli di learnin.
* **Interpretabilità**: spesso non si sa come interpretare milioni di classificazioni a caso.
* **Common**-**sense**: informazioni che spesso non sono codificate nell’input
* **Elevati costi**
* **Sviluppo si piccoli dispositivi**