0 : Cosa c'è in questo notebook

Scaricamanto ed utilizzo modelli di Natural Language Processing da huggingface.com per fare un po di esperimenti ed esempi con dati testuali per il progetto **geografia**.

In particolare si esplorano un po di metodi e strumenti per la comprensione e annotazione dei testi, sfruttando modelli già allenati su task che ci posssono interessare come ad esempio :

- **Mask-Filling** (con huggingface transformer): predice un token mancante in una frase
- **Zero-Shot classification** (con huggingface transformer): classifica una frase rispetto a delle classi variabili (utile per topic analysis, sentiment retrival e menate generiche)
- **Pos-Tagging** (con la libreria spacy) : analisi grammaticale e sintattica del testo

0.1 Transformer Models da hugging face

I modelli pre-allenati sono stati sottoposti ad estensivi training con tantissimi dati, in genere macinati per giorni. i task non-supervisionati tipici utilizzati per l'allenamento sono in genere:

- next sentence prediction (prendi un paragrafo, lo splitti sulla base del carattere '.' ed usi la prima frase come dato e la seconda come label da predirre)
- fill-mask: predirre la parola mancante di una frase (nel training si prendono frasi complete e si maschera un termine a random per frase)

Accanto a questi task usati per allenare la struttura della rete neurale (aka "il grosso" dei nodi profondi), i modelli di huggingface possono essere già "fine-tunati" rispetto ad ulteriori downstream task (tipo sentiment analysis o zero shot) e quindi utilizzabili direttamente. per alcune coppie di modelli-task invece è necessario un ulteriore sessione di fine-tuning con un dataset rilevante per il task d'interesse. Qua c'è una lista delle tipologie di task piu comuni: https://huggingface.co/tasks

Infine un altra differenza sostanziale che ci interessa è se il modello sia:

- monolingua
- multilingua

i modelli considerati per questo girovagare sono:

- il modello **dbmdz/bert-base-italian-cased** è piu leggero e solo in italiano, ma di task ready-to-go disponibili c'è solo fill-mask e pos-tag
- il modello **facebook/bart-large-mnli** è molto pesante, ma è multilingua ed ha implementato lo zero-shot-classification
- il modello **bert-base-cased** solo inglese ma con praticamente tutti i downstream task già pronti all'utilizzo

0.2 Spacy Library

https://spacy.io/ libreria python piu semplice ed efficiente rispetto ai modelli transformer di hugging face, ma meno versatile per quantità di task possibili. ottima per pulire testi e tokenization (preprocessing), molto buono il pos-tagging

In [1]:
```python
from transformers import AutoModel, AutoTokenizer
from transformers import pipeline
multilingual_model_name = "facebook/bart-large-mnli" # 1.5 GB
italian_model_name = "dbmdz/bert-base-italian-cased" # 422 MB
alltask_model_name = "bert-base-cased"

## Download model and configuration from huggingface.co and cache.
#muli_tokenizer = AutoTokenizer.from_pretrained(multilingual_model_name)
#muli_model = AutoModel.from_pretrained(multilingual_model_name)
```

```
2022-05-03 17:55:08.489149: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open shared object file: No such file or directory
2022-05-03 17:55:08.489192: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
```

1 : Fill-Mask task example

In [2]:
```python
fill = pipeline('fill-mask',model=italian_model_name)
masked_sentence = 'Umberto Eco è [MASK] un grande scrittore'
fill(masked_sentence)
```

```
Some weights of the model checkpoint at dbmdz/bert-base-italian-cased were not used when initializing BertForMaskedLM: ['cls.seq_relationship.bias', 'cls.seq_relationship.weight']
- This IS expected if you are initializing BertForMaskedLM from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequence Classification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertForMaskedLM from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
```

Out[2]:
```
[{'score': 0.9144052267074585,
  'token': 482,
  'token_str': 'stato',
  'sequence': 'Umberto Eco è stato un grande scrittore'},
 {'score': 0.02569987252354622,
  'token': 5801,
  'token_str': 'diventato',
  'sequence': 'Umberto Eco è diventato un grande scrittore'},
 {'score': 0.022715188562870026,
  'token': 409,
  'token_str': 'anche',
  'sequence': 'Umberto Eco è anche un grande scrittore'},
 {'score': 0.006274290382862091,
  'token': 1402,
  'token_str': 'oggi',
  'sequence': 'Umberto Eco è oggi un grande scrittore'},
 {'score': 0.004773843102157116,
  'token': 14743,
  'token_str': 'divenuto',
  'sequence': 'Umberto Eco è divenuto un grande scrittore'}]
```

In [3]:
```python
masked_sentence = 'a che ora ci [MASK]?'
fill(masked_sentence)
```

Out[3]:
```
[{'score': 0.1162104457616806,
  'token': 7607,
  'token_str': 'troviamo',
  'sequence': 'a che ora ci troviamo?'},
 {'score': 0.06763437390327454,
  'token': 17567,
  'token_str': 'aspettiamo',
  'sequence': 'a che ora ci aspettiamo?'},
 {'score': 0.06530724465847015,
  'token': 1303,
  'token_str': 'sarà',
  'sequence': 'a che ora ci sarà?'},
 {'score': 0.05703260377049446,
  'token': 4238,
  'token_str': 'vediamo',
  'sequence': 'a che ora ci vediamo?'},
 {'score': 0.053022969514131546,
  'token': 17387,
  'token_str': 'vedete',
  'sequence': 'a che ora ci vedete?'}]
```

In [4]:
```python
masked_sentence = 'a che [MASK] ci troviamo?'
fill(masked_sentence)
```

Out[4]:
```
[{'score': 0.3658718466758728,
  'token': 510,
  'token_str': 'cosa',
  'sequence': 'a che cosa ci troviamo?'},
 {'score': 0.0469609797000885,
  'token': 739,
  'token_str': 'modo',
  'sequence': 'a che modo ci troviamo?'},
 {'score': 0.043758004903793335,
  'token': 212,
  'token_str': 'non',
  'sequence': 'a che non ci troviamo?'},
 {'score': 0.023330725729465485,
  'token': 1302,
  'token_str': 'livello',
  'sequence': 'a che livello ci troviamo?'},
 {'score': 0.02314302697777748,
  'token': 1711,
  'token_str': 'condizioni',
  'sequence': 'a che condizioni ci troviamo?'}]
```

2 : Zero-Shot classification task example

uso modello multilingua (unico disponibile su huggingface in grado di performare l'inferenza zero-shot in italiano)

link generico in italiano sullo **zero-shot** learning e inference: https://zephyrnet.com/it/Zero-shot-learning-puoi-classificare-un-oggetto-senza-vederlo-prima/

In [5]:
```python
zs = pipeline("zero-shot-classification",model=multilingual_model_name)
```

In [6]:
```python
zs('che bella giornata di sole', candidate_labels=['positivo','negativo']) #multiclass
```

Out[6]:
```
{'sequence': 'che bella giornata di sole',
 'labels': ['positivo', 'negativo'],
 'scores': [0.9892858266830444, 0.010714183561503887]}
```

In [7]:
```python
zs(sequences='dove vai stasera?',hypothesis_template='questa frase è una {}',candidate_labels=['affermazione','domanda'])
```

Out[7]:
```
{'sequence': 'dove vai stasera?',
 'labels': ['domanda', 'affermazione'],
 'scores': [0.606775164604187, 0.3932248651981354]}
```

In [8]:
```python
zs(sequences='where are you going?',hypothesis_template='this phrase is a {}',candidate_labels=['question','affirmation'])
```

Out[8]:
```
{'sequence': 'where are you going?',
 'labels': ['question', 'affirmation'],
 'scores': [0.8955954909324646, 0.10440445691347122]}
```

In [9]:
```python
zs(sequences='Voglio uscire con voi stasera, ma devo pulire casa prima',candidate_labels=['dubbio','invito','carro','positivo','negativo'], multiclass=True)
```

Out[9]:
```
{'sequence': 'Voglio uscire con voi stasera, ma devo pulire casa prima',
 'labels': ['invito', 'carro', 'negativo', 'dubbio', 'positivo'],
 'scores': [0.7679879069328308,
  0.08098021149635315,
  0.06931985914707184,
  0.053561460226774216,
  0.028150461614131927]}
```

In [10]:
```python
#https://www.google.it/search?q=thayer+mood+plane+italia&biw=1528&bih=788&tbm=isch&source=iu&ictx=1&vet=1&fir=idr_CWra3dTA3M%252C3vr2PPAMir9TdM%252C_%253BEhLHcfw0fkhWHM%252CEh6xJHl
moods=['fastidio','rabbia','sconforto','tristezza','noia','pace','relax','calma','felicità','gioia','indifferenza','interesse','dubbio','eccitazione']
zs(sequences='Sono felice di uscire con voi stasera',candidate_labels=moods, multiclass=True)
```

Out[10]:
```
{'sequence': 'Sono felice di uscire con voi stasera',
 'labels': ['felicità',
  'relax',
  'calma',
  'interesse',
  'eccitazione',
  'fastidio',
  'pace',
  'indifferenza',
  'gioia',
  'noia',
  'dubbio',
  'rabbia',
  'sconforto',
  'tristezza'],
 'scores': [0.4919441342353821,
  0.1856914758682251,
  0.11093546450138092,
  0.05410005897283554,
  0.03366483747959137,
  0.03167872875928879,
  0.022022755816578865,
  0.01747749000787735,
  0.015628598630428314,
  0.013213596306741238,
  0.00887034647166729,
  0.006537520792335272,
  0.005070512183010578,
  0.0031645605340600014]}
```

In [11]:
```python
sentenza = 'non capisco perchè te la prendi con me, ti ho già regalato tutte le mi capre, non posso darti anche il bue'
zs(sequences=sentenza,candidate_labels=moods, multiclass=True)
```

Out[11]:
```
{'sequence': 'non capisco perchè te la prendi con me, ti ho già regalato tutte le mi capre, non posso darti anche il bue',
 'labels': ['sconforto',
  'dubbio',
  'indifferenza',
  'calma',
  'relax',
  'tristezza',
  'interesse',
  'pace',
  'eccitazione',
  'gioia',
  'noia',
  'fastidio',
  'rabbia',
  'felicità'],
 'scores': [0.30105099081993103,
  0.1431449055671692,
  0.12648096680641174,
  0.09243334829807281,
  0.08244765549898148,
  0.07852306216955185,
  0.05074809119105339,
  0.032442253082990646,
  0.02758951485157013,
  0.023189708590507507,
  0.015240885317325592,
  0.011493487283587456,
  0.008943037129938602,
  0.006272169295698404]}
```

In [12]:
```python
zs(sequences='le tue orecchie sono bruttissime, non uscirò mai con te',candidate_labels=['estetica','logica','attualità','cotillons'], multiclass=True)
```

Out[12]:
```
{'sequence': 'le tue orecchie sono bruttissime, non uscirò mai con te',
 'labels': ['estetica', 'logica', 'attualità', 'cotillons'],
 'scores': [0.3290303647518158,
  0.27184581756591797,
  0.26614731550216675,
  0.1329764872789383]}
```

3 : POS-TAG task example

Positional tagging, task già noto alla maggior parte delle persone aventi la terza elementare come **analisi grammaticale** Nell'analisi grammaticale italiana ogni parola può appartenere ad una delle nove categorie lessicali dell'italiano, cinque variabili: articolo, aggettivo, sostantivo o nome, pronome, verbo, e quattro invariabili: avverbio, preposizione, congiunzione, interiezione o esclamazione.

In [18]:
```python
nlp.path
```

Out[18]:
```
PosixPath('/home/agropunx/anaconda3/envs/geografia/lib/python3.8/site-packages/it_core_news_lg/it_core_news_lg-3.3.0')
```

In [19]:
```python
import spacy
from spacy.lang.it.examples import sentences

try:
    nlp = spacy.load("it_core_news_lg")
except:
    !python -m spacy download it_core_news_lg
    nlp = spacy.load("it_core_news_lg")
doc = nlp(sentences[2])
print('testo originale ---> ',doc.text)
print('\n*** Pos-Tagging')
print('WORD', ' '*(10-len('word')), 'POS-TAG',' '*(10-len('POS-TAG')), 'SYNTACTIC DEP')
print('-------------------------------------')
for token in doc:
    print(token.text,' '*(10-len(token.text)), token.pos_,' '*(10-len(token.pos_)), token.dep_ )

print('\n*** Analisi Entità specifiche nel testo:')
print('ENTITY NAME',' '*(20-len('ENTITY NAME')), 'LABEL')
print('-------------------------------------')
for ent in doc.ents:
    print(ent.text ,' '*(20-len(ent.text)), ent.label_)
```

```
testo originale --->  San Francisco prevede di bandire i robot di consegna porta a porta

*** Pos-Tagging
WORD        POS-TAG     SYNTACTIC DEP
-------------------------------------
San         PROPN       nsubj
Francisco   PROPN       flat:name
prevede     VERB        ROOT
di          ADP         mark
bandire     VERB        xcomp
i           DET         det
robot       NOUN        obj
di          ADP         case
consegna    NOUN        nmod
porta       VERB        advcl
a           ADP         case
porta       NOUN        obl

*** Analisi Entità specifiche nel testo:
ENTITY NAME           LABEL
-------------------------------------
San Francisco         LOC
```

In [20]:
```python
doc = nlp(sentences[3])
print('testo originale ---> ',doc.text)
print('\n*** Pos-Tagging')
print('WORD', ' '*(10-len('word')), 'POS-TAG',' '*(10-len('POS-TAG')), 'SYNTACTIC DEP')
print('-------------------------------------')
for token in doc:
    print(token.text,' '*(10-len(token.text)), token.pos_,' '*(10-len(token.pos_)), token.dep_ )

print('\n*** Analisi Entità specifiche nel testo:')
print('ENTITY NAME',' '*(20-len('ENTITY NAME')), 'LABEL')
print('-------------------------------------')
for ent in doc.ents:
    print(ent.text ,' '*(20-len(ent.text)), ent.label_)
```

```
testo originale --->  Londra è una grande città del Regno Unito.

*** Pos-Tagging
WORD        POS-TAG     SYNTACTIC DEP
-------------------------------------
Londra      PROPN       nsubj
è           AUX         cop
una         DET         det
grande      ADJ         amod
città       NOUN        ROOT
del         ADP         case
Regno       PROPN       nmod
Unito       PROPN       flat:name
.           PUNCT       punct

*** Analisi Entità specifiche nel testo:
ENTITY NAME           LABEL
-------------------------------------
Londra                LOC
Regno Unito           LOC
```

In [21]:
```python
doc = nlp(sentenza.replace('me','Giacomo').replace('mi','sue').replace('ho','ha').replace('posso','può'))
print('testo originale ---> ',doc.text)
print('\n*** Pos-Tagging')
print('WORD', ' '*(10-len('word')), 'POS-TAG',' '*(10-len('POS-TAG')), 'SYNTACTIC DEP')
print('-------------------------------------')
for token in doc:
    print(token.text,' '*(10-len(token.text)), token.pos_,' '*(10-len(token.pos_)), token.dep_ )

print('\n*** Analisi Entità specifiche nel testo:')
print('ENTITY NAME',' '*(20-len('ENTITY NAME')), 'LABEL')
print('-------------------------------------')
for ent in doc.ents:
    print(ent.text ,' '*(20-len(ent.text)), ent.label_)
```

```
testo originale --->  non capisco perchè te la prendi con Giacomo, ti ha già regalato tutte le sue capre, non può darti anche il bue

*** Pos-Tagging
WORD        POS-TAG     SYNTACTIC DEP
-------------------------------------
non         ADV         advmod
capisco     VERB        ROOT
perchè      SCONJ       mark
te          PRON        expl
la          PRON        obj
prendi      VERB        ccomp
con         ADP         case
Giacomo     PROPN       obl
,           PUNCT       punct
ti          PRON        iobj
ha          AUX         aux
già         ADV         advmod
regalato    VERB        conj
tutte       DET         det:predet
le          DET         det
sue         DET         det:poss
capre       NOUN        obj
,           PUNCT       punct
non         ADV         advmod
può         AUX         aux
darti       VERB        conj
anche       ADV         advmod
il          DET         det
bue         NOUN        obj

*** Analisi Entità specifiche nel testo:
ENTITY NAME           LABEL
-------------------------------------
Giacomo               PER
```

In [ ]: