Contradictory, My Dear Watson

Il tipo di task che si analizza in questo progetto è il Textual Entailment, quindi, il rapporto di implicazione, contraddizione o neutralità tra due frasi. A questo fine è stato scelto il dataset fornito da Kaggle nella competizione *Contradictory, My Dear Watson*. Il dataset in questione è multilingue e multi classe è il task di classificazione, inoltre, il task richiede due features testuali, cioè una premessa e un'ipotesi. Il progetto e il seguente report, dunque, sono stati strutturati soppesando le eventuali difficoltà dovute alle considerazioni appena menzionate.

1. Data Understanding

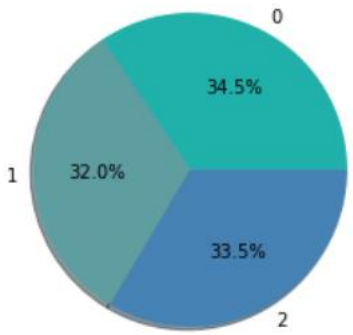
Il dataset utilizzato è composto da 12120 record di cui abbiamo 6 attributi : l'ID del record, la lingua e la sua abbreviazione, una premessa, un' ipotesi e l'etichetta che va ad indicare il rapporto tra ipotesi e tesi.

	id	premise	hypothesis	lang_abv	language	label
0	5130fd2cb5	and these comments were considered in formulat...	The rules developed in the interim were put to...	en	English	0
1	5b72532a0b	These are issues that we wrestle with in pract...	Practice groups are not permitted to work on t...	en	English	2
2	3931fbe82a	Des petites choses comme celles-là font une di...	J'essayais d'accomplir quelque chose.	fr	French	0
3	5622f0c60b	you know they can't really defend themselves l...	They can't defend themselves because of their ...	en	English	0
4	86aaa48b45	ในการเล่นบทบาทสมมุติก็เช่นกัน โอกาสที่จะได้แสด...	เด็กสามารถเห็นได้ว่าชาติพันธุ์แตกต่างกันอย่างไร	th	Thai	1
...	...	...	...	...	...	...
12115	2b78e2a914	The results of even the most well designed epi...	All studies have the same amount of uncertaint...	en	English	2
12116	7e9943d152	But there are two kinds of the pleasure of do...	But there are two kinds of the pleasure of doi...	en	English	0
12117	5085923e6c	The important thing is to realize that it's wa...	It cannot be moved, now or ever.	en	English	2
12118	fc8e2fd1fe	At the west end is a detailed model of the who...	The model temple complex is at the east end.	en	English	2
12119	44301dfb14	For himself he chose Atat??rk, or Father of th...	Ataturk was the father of the Turkish nation.	en	English	0

12120 rows × 6 columns

Le figure sottostanti mostrano che le etichette sono distribuite uniformemente sia tra tutti i record sia tra le lingue, ma con immediatezza si può individuare la prevalenza della lingua inglese nel dataset.

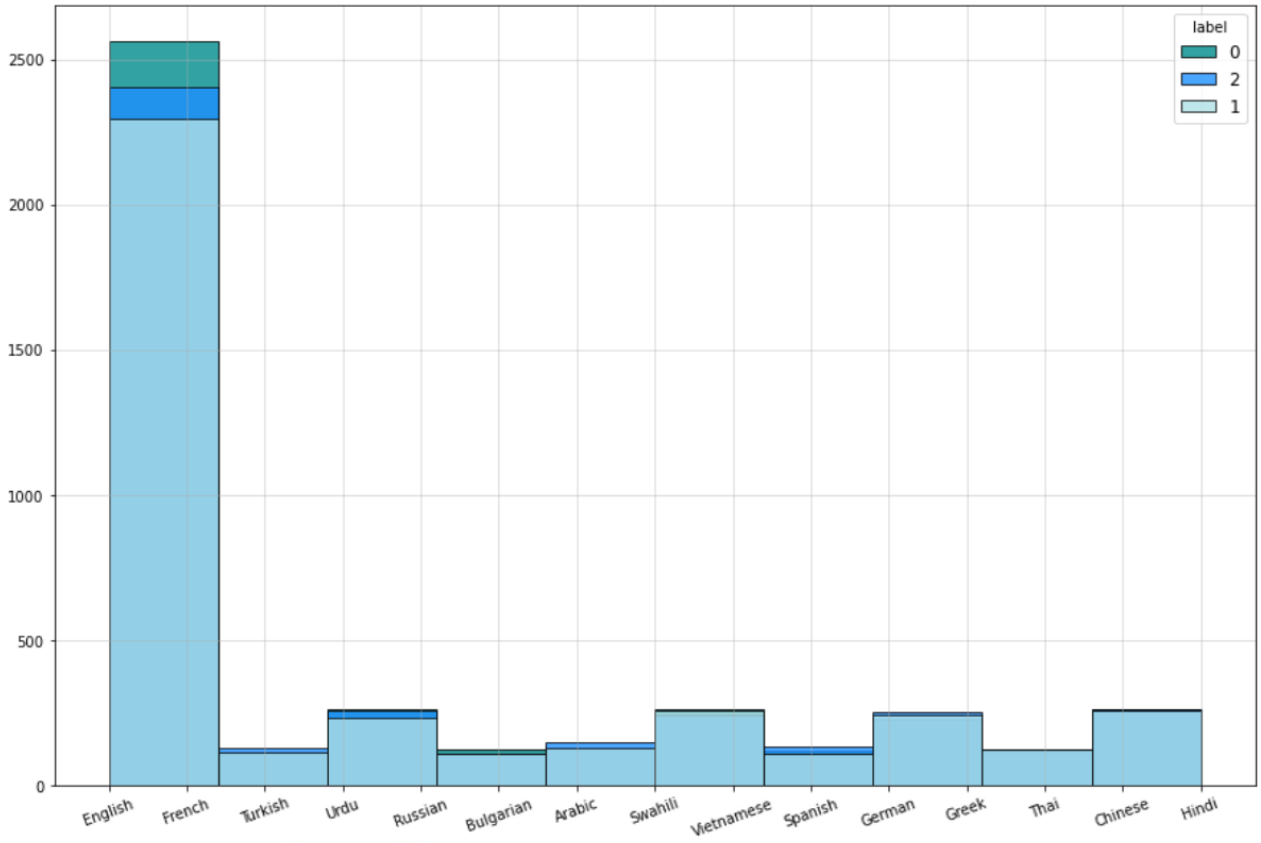
1 Percentuale Classi

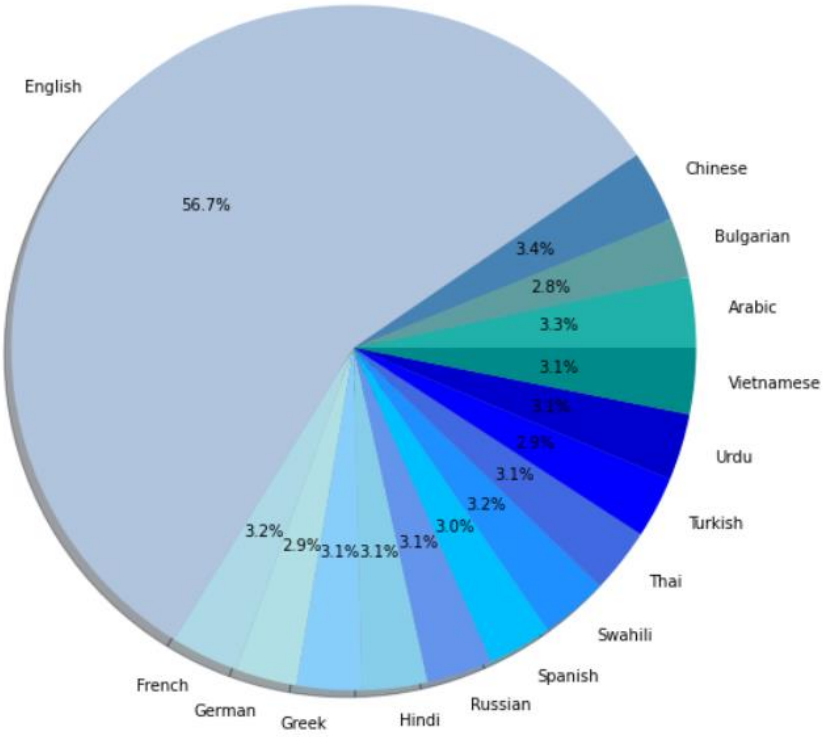


Legend:

0 == entailment
1 == neutral
2 == contradiction

2 Lingue e classi





3 Distribuzione lingue

Considerando la distribuzione delle lingue nel dataset e non potendo trascurare la netta prevalenza di modelli strutturati appositamente per l'inglese, si è deciso di valutare diversi metodi di machine learning sia sull'intero dataset che solamente sulla porzione in lingua inglese.

2. Classificazione con Naive Bayes Classifier, SVM, Random Forest

2.1. Frequencies embeddings

Come noto, i primi processi da eseguire sui dati testuali sono la normalizzazione e la tokenizzazione. Subito, dunque, si deve tenere in considerazione il tipo di lingua da trattare.

Al fine di valutare il comportamento dei vari classificatori con maggiore accuratezza si è deciso utilizzare sia la tokenizzazione di default presente nel metodo CountVectorizer() della libreria Sklearnt, si una implementata con la libreria Spacy. Questo è stato fatto per la lingua inglese e per una porzione di dataset contenente record in russo, in spagnolo e in greco (lingue presenti nel modello *xx_sent_ud_sm* di spacy). Inoltre i classificatori sono stati valutati sull'intero dataset.

Il processo di embedding e di classificazione è avvenuto, per ogni classificatore, attraverso il classico schema di tokenizzazione e vettorizzazione, selezione delle features e weighting, con la seguente forma:

```
Pipeline([
  ('vect', CountVectorizer()), #tokenization
  ('sel', SelectKBest(chi2)),  # feature selection
  ('tfidf', TfidfTransformer()),  # weighting
  ('learner', LinearSVC())  # learning algorithm
])
```

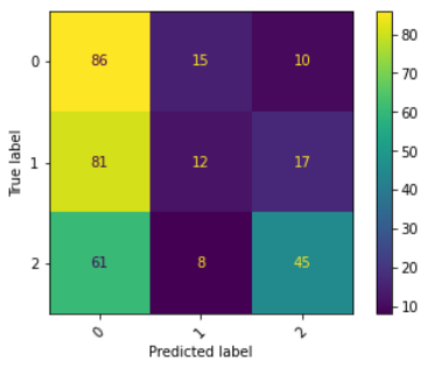
Per ogni classificatore si è effettuata una GridSearch sul numero di features da mantenere e sui parametri salienti del classificatore in questione.

Per quanto riguarda la tokenizzazione con Spacy, per l'inglese si è implementato un metodo che tenesse conto dei trigrammi, risultato però, meno efficacie della tokenizzazione utilizzata da sklearnt. Anche per quanto riguarda la tokenizzazione su più lingue, si può notare che il metodo implementato con spacy non aiuta la classificazione. Nella tabella sottostante sono riportati i valori di accuracy, si rimanda ai notebook per altre metriche.

	Inglese		Russo_spagnolo_greco	
	Default	Spacy	Default	Spacy
Naive Bayes Classifier	0.39	0.36	0.33	0.34
SVM	0.41	0.36	0.43	0.34
Random Forest	0.41	0.36	0.40	0.34

Si riporta di seguito il classification report e la confusion matrix del modello migliore, cioè la Support Vector Machine con tokenizzazione di default sul dataset multilingua.

```
Classification report:
              precision    recall  f1-score   support

           0       0.38      0.77      0.51       111
           1       0.34      0.11      0.17       110
           2       0.62      0.39      0.48       114

    accuracy                           0.43       335
   macro avg       0.45      0.43      0.39       335
weighted avg       0.45      0.43      0.39       335
```



Si può constatare come la classe 1, cioè il rapporto di neutralità, sia quella più difficile da individuare per il classificatore. Oltre ad avere un basso valore di precision è anche la classe alla quale il classificatore assegna meno valori. Si può notare questa tendenza anche nella confusion matrix degli altri classifiatori, con alcune eccezioni.

Per quanto riguarda invece l'intero dataset, si è valutato solo il tokenizattore di sklearnt, ottenendo i seguenti valori di accuracy, che non raggiungono i valori dell'equivalente modello utilizzato su porzioni di dataset monolingue o comunque con meno lingue.

Naive Bayes Classifier	SVM	Random Forest
0,38	0,39	0,39

Una questione importante da affrontare è come mantenere premesse e ipotesi due features diverse. Per quanto riguarda i modelli appena descritti, non si è potuto mantenere questa separazione utilizzando il tokenizzatore di default e per questo anche i modelli spacy sono stati allenati in questo modo. Possiamo dunque pensare che i modelli abbiano imparato a riconosce implicazione, contraddizione o neutralità all'interno di una frase, non il rapporto tra due.

2.2. Word2Vec embeddings

I modelli Word2Vec attraverso due livelli in una rete neurale creano embedding di parole in contesto a bassa dimensionalità. Si è scelto di utilizzare il modello word2vec Skip-gram fornito dalla libreria Genism[1]. Una volta ottenuti gli embedding delle parole attraverso il modello, sono stati generati dei vettori aggregati per le frasi basandosi sui singoli vettori di ogni parola nella frase. I "vettori frase" finali sono stati poi creati attraverso la media dei vettori parola per le parole contenute nella frase.

Questo procedimento è stato svolto sia sull'intero dataset che sulla porzione inglese, utilizzando il *word_tokenizer* di NLTK. Inoltre si è deciso di valutare la classificazione sia unendo premessa e ipotesi in un'unica frase come per i frequencies embedding, sia mantenendo le due features separate nella fase di embedding e concatenando i "vettori-premessa" con i "vettori-ipotesi" prima della normalizzazione con MinMaxScaler di sklearnt. In ogni caso gli embedding per le singole parole sono stati generati fornendo al modello l'intero training set, ma i "vettori frase" sono stati generati separatamente. Nella tabella sottostante i valori di accuracy.

	Inglese		Dataset intero	
	1 feature	2 features	1 feature	2 features
Naive Bayes Classifier	0.34	0.34	0.34	0.35
SVM	0.36	0.41	0.33	0.37
Random Forest	0.32	0.36	0.32	0.35

Si può notare come la Support Vector Machine e il Random Forest giovino della separazione delle due frasi. Confrontando, però, questi risultati con quelli ottenuti dalla classica Pipeline di sklearnt notiamo che solo SVM con separazione premessa-ipotesi raggiunge gli stessi risultati per l'inglese. Si è voluto anche provare a testare i classificatori senza la parte di feature selection e di weighting ottenendo risultati peggiori. Possiamo dunque pensare che il task non benefici di questo tipo di embedding.

2.3. Doc2Vec embedding.

Il tipo di embedding creato da Doc2vec estende quello della sezione precedente inserendo un identificatore dei documenti all'interno dello spazio degli embedding delle parole.

[1] https://radimrehurek.com/gensim/index.html

Questo modello è stato utilizzato sia provando ad usare l'id dei vari record come id dei documenti, sia utilizzando le etichette delle classi come id. Nel primo caso, dunque, non è stata mantenuta la separazione premessa-ipotesi, ma sono stati creati vettori unici per l'intero record già in fase di embedding. Nel secondo caso, invece, i classificatori sono stati valutati sia mantenendo la separazione che no, ottenendo risultati migliori con la separazione.

Il procedimento utilizzato per mantenere la separazione è simile a quello usato nella sezione precedente, con la differenza che il vocabolario del modello è stato creato utilizzando solamente le premesse. Si pensa, dunque, che si possano raggiungere ulteriori miglioramenti utilizzando l'intero training set.

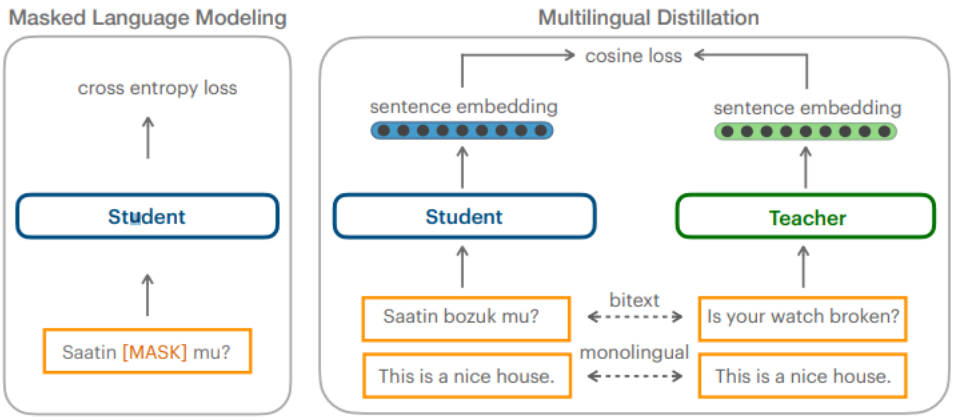
L'intero procedimento è stato svolto solo per la lingua inglese. Nella tabella i valori di accuracy.

	ID = id record	ID = classe /1 features	ID = classe /2 features
Naive Bayes Classifier	0,34	0,31	0,33
SVM	0,39	0,31	0,41
Random Forest	0,35	0,31	0,42

Si osserva come utilizzare come ID la classe di appartenenza dei documenti senza mantenere la separazione premessa-ipotesi risulti una scelta poco felice, essendo questi i risultati peggiori. La separazione, però, livella nuovamente le performance a quelle dei frequencies embedding, senza miglioramenti. Si pensa, però, che il document embedding sia comunque sufficientemente adatto al task in questione. Inoltre, valutando i classificatori senza feature selection e weighting si sono ottenuti risultati uguali o talvolta migliori (data la casualità di alcune condizioni determinanti, a partire dalla suddivisione in train e test, non tutte le run dello stesso modello ottengono lo stesso risultato).

2.4. LASER embedding.

LASER (Language-Agnostic SEntence Representations) è una raccolta di script e modelli creati da Facebook Research per calcolare sentence embedding multilingua ed è in grado di trasformare le frasi in vettori indipendenti dalla lingua. Si basa LSTM encoder/decoder a cui aggiunge un approccio di tipo teacher-student, la cui architettura è sintetizzata nell'immagine sottostante.



I classificatori con questo tipo di embedding sono stati valutati sia unendo premessa ed ipotesi, sia mantenendole separate. Per mantenere le due frasi come features separate sono stati valutati due metodi : uno in cui i vettori delle premesse e quelli delle ipotesi sono stati concatenati dopo l'embedding e uno in cui gli embedding sono stati creati da un unico input per record contenente le due frasi separate.

In tabella i valori di accuracy.

	1feature	Embedding separato	Embedding unico
SVM	0,50	0,51	0,51
Random Forest	0,41	0,49	0,44

Con questo embedding si raggiungono i valori massimi di accuracy per SVM e Random Forest, notiamo soprattutto come il Random Forest migliori quando viene mantenuta la separazione premessa-ipotesi.

3. Classificazione con ELMo embedding

ELMo (Embedding from Language Model) sfrutta gli stati nascosti di una deep LSTM , bidirezionale e character-level per generare rappresentazioni contestuali di parole in una frase. Al fine di valutare questo embedding è stato utilizzato il codice *TextClassification.py* fornito dalla libreria simple-elmo. Questo script può essere utilizzato per un task di document pair classification (come nel rilevamento dell'entailment o paraphrase). Viene utilizzata la media degli embedding ELMo per tutte le parole in un documento, quindi, la cosine-similarity tra due documenti viene calcolata e utilizzata come singola caratteristica del classificatore (Logistic regressor). Il classificatore viene valutato con macro F1 score e cross validation con k=10, ottenendo i seguenti risultati.

```
Distribution of classes in the whole sample: {0: 2427, 2: 2277, 1: 2166}
Train shape: 6870
Real scores:
=====
Precision: 0.413
Recall: 0.432
F1: 0.400
=====
Random choice scores:
=====
Precision: 0.331
Recall: 0.331
F1: 0.330
```

4. Classificazione con CNN e LSTM

Per valutare l'utilizzo di reti neurali di tipo convoluzionale e di tipo ricorrente si sono utilizzate la piattaforma open-source TensorFlow e la libreria Keras.

Come reti convuzionali sono state create reti con filtro 1D, essendo il testo un dito di dato sequenziate.

Come rete neurale ricorrente è stata scelta la Long Short Term Memory, implementata anche con strati bidirezionali.

Per ogni sezione seguente sono state valutate quattro differenti reti neurali. La struttura di tali reti è indicata nelle figure sottostanti, a fianco il nome con cui mi riferirò ai modelli.

```
input = Input(shape=(200,), dtype="int64")
x   =  Embedding(max_features, embedding_dim)(input)
#x   =  Dropout(0.2)(x)
x   =  Conv1D(100, 3, padding='valid',activation='relu', strides=1)(x)
x   =  GlobalMaxPooling1D()(x)
x   =  Dense(64, activation="relu")(x)
x   =  Dropout(0.2)(x)
x   =  Dense(32, activation="relu")(x)
x   =  Dropout(0.2)(x)
x   =  Dense(3, activation="softmax")(x)
```

1_Conv1D

1 strato conv. con filtro di dimensione 3

```
# A integer input for vocab indices.
inputs = Input(shape=(200,), dtype="int64")

# Next, we add a layer to map those vocab indices into a space of dimensionality
# 'embedding_dim'.
x = Embedding(max_features, embedding_dim)(inputs)
x = Dropout(0.1)(x)

# Conv1D + global max pooling
x = Conv1D(128, 7, padding="valid", activation="relu", strides=3)(x)
x = Conv1D(128, 7, padding="valid", activation="relu", strides=3)(x)
x = GlobalMaxPooling1D()(x)

# We add a vanilla hidden layer:
x = Dense(128, activation="relu")(x)
x = Dropout(0.1)(x)

# We project onto a single unit output layer, and squash it with a sigmoid:
x = layers.Dense(3, activation="softmax", name="predictions")(x)
```

2_Conv1D

2 strati conv. Con filtro di dimensione 7

```
input = Input(shape=(200, ))
x   =  Embedding(max_features, embedding_dim)(input)
x   =  LSTM(200, return_sequences=True,name='lstm_layer')(x)
x   =  GlobalMaxPool1D()(x)
x   =  Dense(64, activation="relu")(x)
x   =  Dropout(0.2)(x)
x   =  Dense(3, activation="softmax")(x)
```

1_lstm

1 strato lstm con pooling

```
inputs = Input(shape=(200,))
# Embed each integer in a 128-dimensional vector
x = Embedding(max_features, embedding_dim)(inputs)
# Add 2 bidirectional LSTMs
x = Bidirectional(LSTM(200, return_sequences=True))(x)
x = Bidirectional(LSTM(64))(x)
#x = Dropout(0.2)(x)
x = Dense(64, activation='relu')(x)
x = Dropout(0.2)(x)
# Add a classifier
x = layers.Dense(3, activation="softmax")(x)
```

2_BiDir

2 strati bidirezionali lstm

Alcune modifiche sono state apportate a seconda del tipo di embedding usato, ad esempio lo shape in input o la scelta delle dimensioni degli strati. Si rimanda ai notebook per i dettagli.

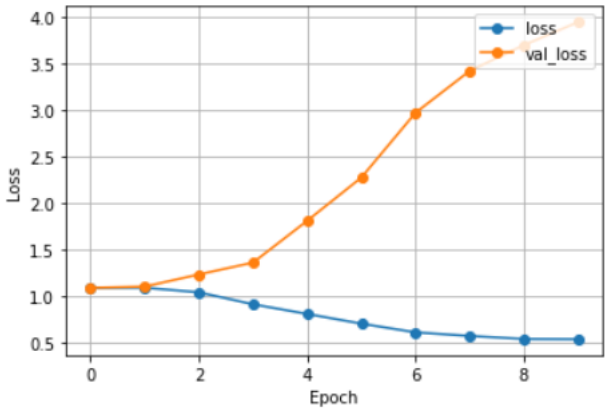
Ogni rete ha usato come loss function la categorical crossentropy, come optimizer ADAM e come metrica di valutazione l'accuracy. Le epoche del training variano da 3 a 5.

4.1. Keras embedding → word embedding

Il testo è stato tokenizzato e reso in forma vettoriale mediante il TextVectorization() di TensorFlow, per alcuni problemi di encoding, però, si è dovuto utilizzare un vocabolario del testo creato tramite NLTK. Infatti, tramite il primo metodo s'incontrava un problema di codifica volendo utilizzare solo le singole parole e non gli ngrammi, questo anche perché fin dalle prime valutazioni si è riscontrata una pessima performance dei classificatori con questo tipo di tokenizzazione. Nuovamente si è proceduto a valutare i classificatori sia unendo premessa ed ipotesi, sia mantenendole separate, sia sull'intero dataset che sulla porzione inglese.

	Inglese		Intero dataset	
	1 features	2 features	1 features	2 features
2_Conv1D	0.139	0.267	0.180	0.248
1_Conv1D	0.143	0.163	0.177	0.175
1_lstm	0.131	0.235	0.204	0.201
2_BiDir	0.229	0.135	0.215	0.125

Si nota con immediatezza che le reti neurali lavorano meglio su un dataset più grande, nonostante sia multilingue. Inoltre vediamo che la separazione premessa-ipotesi giova solamente all'inglese e non alla rete bidirezionale. Questo andamento però potrebbe essere anche dovuto alla necessità di più epoche. Infatti, nonostante tutti i modelli vadano in overfitting in poche epoche (notare la figura sottostante), la rete bidirezionale ottiene risultati accettabili solo allenata con un numero di epoche maggiore (almeno 10).



4.2. Pre-trained emdebbing → Glove e FastText

Le reti neurali presentate nella sezione precedente tranne la 2Conv1D (sostituita da una più semplice) sono state valutate anche tramite l'utilizzo di pre-trained embdeddings per la lingua inglese.

Il Glove embedding genera embedding di parole aggregando le metrici di co-occorenza delle parole dato un corpus. Il modello scelto è *Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors)* utilizzato con i vettori 100d e 200d in base alla forma vettoriale utilizzata con il TextVectorization().

FastText è un altro metodo di word embedding. Si tratta di un'estensione del modello word2vec. Invece di apprendere direttamente i vettori per le parole, FastText rappresenta ogni parola come un n-grammo di caratteri. Il modello scelto è *crawl-300d-2M.vec.zip: 2 million word vectors trained on Common Crawl (600B tokens)*, i vettori delle parole dunque sono stati creati con 300 dimensioni.

Essendo gli embedding addestrarti in precedenza sulla lingua inglese, questo tipo di processo è stato svolto solamente per tale lingua.

	1 feature		2 features	
	Glove	**FastText**	**Glove**	**FastText**
Conv1D	0.249	0.174	0.324	0.166
1_Conv1D	0.215	0.174	0.237	0.186
1_lstm	0.221	0.070	0.254	0.210
2_BiDir	0.282	0.175	0.134	0.074

Notiamo immediatamente come i pre-trained embedding di Glove migliorino le prestazioni delle reti neurali e come giovino anche della separazione tra premessa e ipotesi. Notiamo invece solo un lieve miglioramento dato dagli embedding di FastText con cui soprattutto la LSTM tocca performance sorprendentemente basse.

Risulta dunque corretta la percezione già prima enunciata secondo cui gli embedding word2vec o una loro estensione non risultino adatti a questo tipo di task.

5. Dense neural network con LASER embedding

Date le alte performance dei classificatori non neurali con questo embedding si è deciso di valutare anche due reti neurali dense. Si è deciso di valutare i classificatori mantenendo la separazione tra premessa-ipotesi, in aggiunta si è provato ad aggiungere all'embedding il valore di cosine similarity tra le due frasi, su ispirazione della classificazione svolta con l'embedding ELMo.

Le reti neurali testate sono strutturate come segue, a fianco i nomi con cui mi riferirò ai modelli

```
input = Input(shape=(embeddings.shape[1], ))
x    =  Dense(64, activation="relu")(input)
x    =  Dropout(0.2)(x)
x    =  Dense(3, activation="softmax")(x)
model = Model(inputs=input, outputs=x)
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Dense_64

Un solo strato denso tra l'input e il livello di output.

```
input = Input(shape=(embeddings.shape[1], ))
x = Dense(512, activation='relu')(input)

x = Dropout(0.2)(x)
x = Dense(128, activation='relu')(x)

x = Dropout(0.2)(x)
x = Dense(64, activation='relu')(x)

x = Dropout(0.2)(x)
x = Dense(32, activation='relu')(x)

x = Dropout(0.2)(x)
x = Dense(8, activation='relu')(x)

x = Dropout(0.2)(x)
x = Dense(3, activation="softmax")(x)
mod = Model(inputs=input, outputs=x)
mod.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Dense_512

Più strati densi

Entrambe le reti sono state valutate cambiando il valore di numero di epoche, quelli riportati sono i valori migliori.

	2 feature	2 feature + cos_similarity
Dense_64	0.41	0.43
Dense_512	0.49	0.54

Si può notare un leggero miglioramento con l'addizione della similarità coseno al input della rete neurale e con un numero maggiore di strati. I valori raggiunti dalla rete Dense_512 con aggiunta della similarità superano quelli della SVM con 2 feature. Si è dunque deciso di provare a valutare la SVM con l'aggiunta della similarità coseno raggiungendo un'accuracy del 0.57. Si dimostra così come l'addizione di questo valore apporti un'informazione utile alla classificazione.

6. BERT

BERT (Bidirectional Encoder Representations from Transformers) è una rete neurale che utilizza i transformers (strutture encoder/decoder) come blocchi nella sua struttura al fine di costruire ad ogni livello embedding contestualizzati. Il livello di contestualizzazione si affina con la profondità dei livelli di BERT.

Questo modello è pre-addestrato sulla predizione di masked-word e sulla classificazione della “frase successiva”. I modelli usati per questo task sono bert_base_uncased per l’inglese e bert-base-multilingual-cased per l’intero dataset. Grazie alla possibilità di aggiungere token speciali per segnalare al modello l’inizio e la fine di una frase, tramite un semplice preprocessing si è potuto mantenere la separazione premessa-ipotesi.

```
Test set:
Accuracy: [1372/2061] 0.6657
Classification report:
              precision    recall  f1-score   support

           0       0.67      0.72      0.69       720
           1       0.63      0.66      0.65       637
           2       0.69      0.62      0.65       704

    accuracy                           0.67      2061
   macro avg       0.67      0.67      0.66      2061
weighted avg       0.67      0.67      0.67      2061

Confusion matrix:
[[516 118  86]
 [110 421 106]
 [142 127 435]]
```

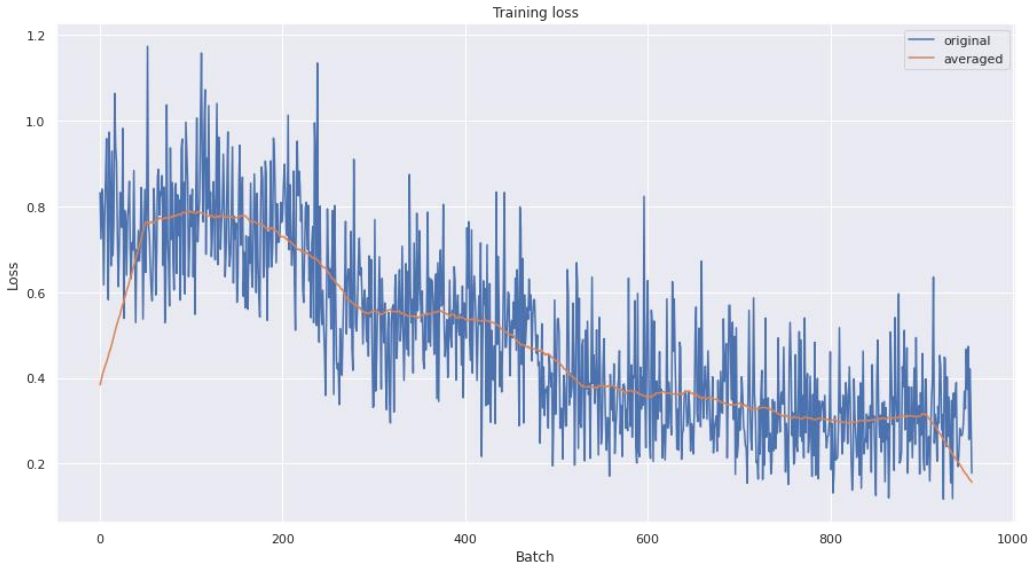
BERT_inglese

```
Test set:
Accuracy: [2322/3636] 0.6386
Classification report:
              precision    recall  f1-score   support

           0       0.70      0.63      0.66      1237
           1       0.61      0.66      0.63      1147
           2       0.62      0.63      0.63      1252

    accuracy                           0.64      3636
   macro avg       0.64      0.64      0.64      3636
weighted avg       0.64      0.64      0.64      3636

Confusion matrix:
[[774 207 256]
 [157 756 234]
 [177 283 792]]
```

BERT_intero dataset

I risultati mostrati sono stati ottenuti utilizzando il modello pre-addestrato con un singolo layer di classificazione aggiuntivo.

Di seguito si mostra l’andamento della training loss all’aumentare del batch per il dataset completo di tutte le lingue



7. Conclusioni

Si conclude sottolineando che solo attraverso la creazione di embedding più complessi, come quelli creati da BERT o da LASER, si possono raggiungere performance accettabili da parte dei classificatori analizzati. La necessità di embedding sofisticati è dovuta non solo a fatto di trattare lingue diverse nello stesso momento (come abbiamo visto le classificazioni solo per l'inglese non raggiungono risultati elevati con gli embedding classici) ma anche alla necessità di individuare una relazione tra due frasi, come richiede il nostro task. Si individua in questo aspetto una caratteristica cruciale per una buona classificazione e altri metodi di separazione premessa-ipotesi dovrebbero essere testati per valutarne l'effettiva importanza. Inoltre si individua nella possibilità di aggiungere la cosine similarity tra il vettore premessa e il vettore ipotesi ai vettori in input una buona addizione di informazione che porta a risultati migliori sui classificatori su cui si è testata.