TEXT MINING
PROGETTO D'ESAME: ANALISI DI RECENSIONI DI FILM

Lorenzo Bandini

1. INTRODUZIONE

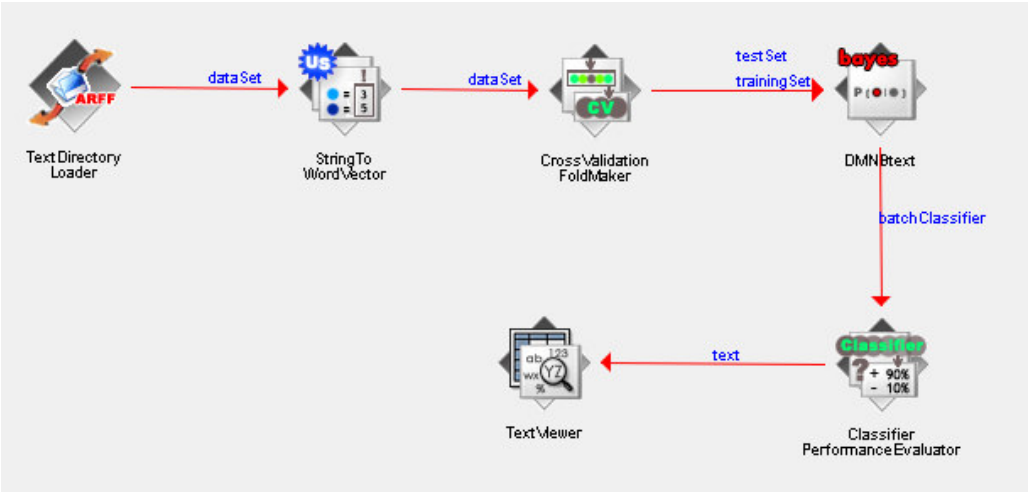
Questo elaborato si pone l'obiettivo di sintetizzare un processo di text mining su di uno specifico dominio applicativo tramite un processo di analisi su dati testuali non strutturati, al fine di descriverne i pattern e comprenderne i contenuti.

2. DESCRIZIONE DEL PROBLEMA

Il dominio applicativo preso in esame è costituito da una collezione di recensioni di film, ognuna di esse etichettata con un giudizio, positivo o negativo, a seconda del contenuto del testo della recensione. L'obiettivo che ci si pone è quindi quello di prendere questo dataset e di svolgere su di esso un'attività di text mining, al fine di estrarre pattern significativi e precedentemente sconosciuti. I passi di questa analisi quindi saranno principalmente due:

1. Classificazione predittiva di recensioni positive e negative in base al testo di queste ultime, utilizzando diverse tecniche di mining supervisionate e non supervisionate, i quali risultati verranno poi confrontati e discussi;
2. Estrazione di un particolare pattern, che caratterizza i testi delle opinioni negative, tramite Latent Semantic Analysis.

Gli strumenti utilizzati saranno Weka e R Studio. Per facilitare la prima fase dell'analisi, è stata utilizzata la modalità knowledgeFlow del tool Weka: questa modalità permette di esprimere il flusso di controllo dell'analisi, utilizzando un'interfaccia grafica intuitiva, e permette di automatizzare il processo di mining.



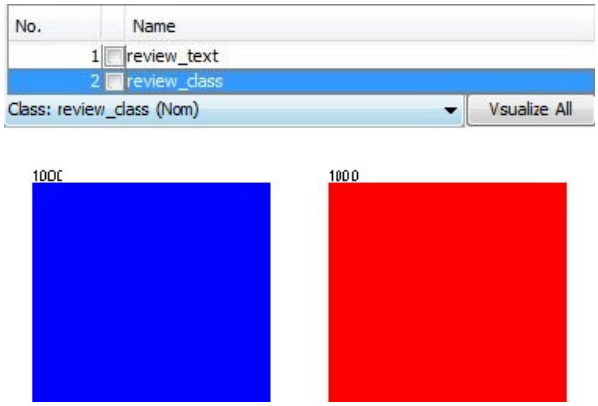
Knowledge Flow utilizzato nel progetto

3. DATA UNDERSTANDING

Il dataset preso in analisi è stato scaricato da questo sito: [http://www.cs.cornell.edu/People/pabo/movie-review-data/], e contiene al suo interno 2000 recensioni di film, delle quali 1000 classificate come positive e 1000 come negative. Queste recensioni sono salvate in file di testo separati, e raggruppati in due sottocartelle, una per le recensioni positive e una per quelle negative. Per poter rendere disponibili i dati

quindi è stato utilizzato il Text Directory Loader messo a disposizione da Weka, che permette di trasformare dati testuali suddivisi in cartelle in file con estensione .arff, comprensibile dal tool Weka. Il file generato quindi ha 2 attributi:

- review_text , di tipo string, contenente il testo delle recensioni;
- review_class , di tipo Nominale che può assumere valori appartenenti all'insieme {neg, pos}, derivati dai nomi delle cartelle che contenevano rispettivamente recensioni negative e positive.



L'attributo review_text non può però essere processato in questo formato. C'è quindi bisogno di un pre-processing sui dati.

4. DATA PREPARATION

Per poter essere utilizzate, le recensioni devono essere strutturate(modello bag of word) al fine di ottenre una matrice termini documenti. Per fare ciò viene utilizzato il filtro di Weka StringToWordVector. I parametri di questo filtro sono:

- IDFTransform -> True;
- TFTransform -> True;
- attributeIndices -> 1;
- MinTermFreq -> 2;
- DoNotCheckCapabilities -> False;
- LowerCaseTokens -> True;
- NormalizeDocLength -> Normalize All Data;
- StopWordsHandler -> stopwords_eng.txt.

Altri parametri non elencati qui verranno settati in differenti configurazioni nella fase successiva, in modo tale da poter osservare le differenze nei risultati dell'analisi.

5. MODELING

5.1 CLASSIFICAZIONE SUPERVISIONATA IN WEKA

Gli algoritmi utilizzati sono i seguenti:

- Bayes: DMNBText;
- Tree: J48;
- Support Vecot Machine: SMO.

I parametri degli algoritmi saranno lasciati ai valori di default, per far risaltare le differenze che si verranno a creare modificando i dati, e non gli algoritmi. Per il training, è stato scelto il CrossValidation, che esegue 10 volte l'algoritmo utilizzando ogni volta 9 fold come training e 1 come test, ogni volta combiando il fold utilizzato.
Come accennato prima, oltre a diversi classificatori, verranno combinati in diverse configurazioni anche i seguenti parametri del filtro StringToWordVector:

- Stemmer -> modificando questo parametro, è possibile impostare un algoritmo di stemming per portare le parole selezionate alla loro radice;
- WordsToKeep -> modificando questo parametro, è possibile impostare il numero di parole che il filtro deve generare;
- DoNotOperateOnPerClassBasis -> settando questo parametro a FALSE, è possibile estrarre almeno WordToKeep parole per ciascuna classe.

Come ulteriore variazione degli algoritmi, verrà anche introdotto il filtro AttributeSelector, tramite il quale si esegue la selezione di un sottoinsieme di attributi più informativi.

ALGORITHM	ACCURACY	PRECISION		RECALL		F-MEASURE	
		Pos	Neg	Pos	Neg	Pos	Neg
j-48 stemmer -> NO WordsToKeep -> 1650 DoNotOperate.. -> True AttributeSelector -> NO	67.65	0,682	0,672	0,662	0,691	0,672	0,681
j-48 stemmer -> SI WordsToKeep -> 1650 DoNotOperate.. -> True AttributeSelector -> NO	67.65	0,682	0,672	0,662	0,691	0,672	0,681
j-48 stemmer -> NO WordsToKeep -> 250 DoNotOperate.. ->True AttributeSelector -> NO	59	0,586	0,594	0,611	0,569	0,598	0,581
j-48 stemmer -> SI WordsToKeep -> 1650 DoNotOperate.. -> False AttributeSelector -> NO	66.45	0,669	0,660	0,650	0,679	0,660	0,669
j-48 stemmer -> SI WordsToKeep -> 1650 DoNotOperate.. -> True AttributeSelector -> SI	70.8	0,734	0,687	0,653	0,763	0,691	0,723
DMNBText stemmer -> NO WordsToKeep -> 1650 DoNotOperate.. -> True AttributeSelector -> NO	83.35	0,831	0,837	0,838	0,829	0,834	0,833
DMNBText stemmer -> SI WordsToKeep -> 1650 DoNotOperate.. -> True AttributeSelector -> NO	83.35	0,831	0,837	0,838	0,829	0,834	0,833
DMNBText stemmer -> NO WordsToKeep -> 250 DoNotOperate.. ->True AttributeSelector -> NO	73.45	0,728	0,742	0,749	0,720	0,738	0,731
DMNBText stemmer -> SI WordsToKeep -> 1650 DoNotOperate.. -> False AttributeSelector -> NO	84.9	0,845	0,853	0,855	0,843	0,850	0,848
DMNBText stemmer -> SI WordsToKeep -> 1650	81.6	0,816	0,816	0,816	0,816	0,816	0,816

DoNotOperate.. -> True AttributeSelector -> SI							
SMO stemmer -> NO WordsToKeep -> 1650 DoNotOperate.. -> True AttributeSelector -> NO	79.05	0,789	0,792	0,793	0,788	0,791	0,790
SMO stemmer -> SI WordsToKeep -> 1650 DoNotOperate.. -> True AttributeSelector -> NO	79.05	0,789	0,792	0,793	0,788	0,791	0,790
SMO stemmer -> NO WordsToKeep -> 250 DoNotOperate.. ->True AttributeSelector -> NO	71.2	0,711	0,713	0,714	0,710	0,713	0,711
SMO stemmer -> SI WordsToKeep -> 1650 DoNotOperate.. -> False AttributeSelector -> NO	81.75	0,811	0,824	0,828	0,807	0,819	0,816
SMO stemmer -> SI WordsToKeep -> 1650 DoNotOperate.. -> True AttributeSelector -> SI	81.65	0,805	0,829	0,835	0,798	0,820	0,813

Osservazioni:

- L'algoritmo J48 ha tempi di calcolo molto superiori rispetto agli altri due algoritmi, ed inoltre classifica il dataset in maniera peggiore; SMO ha tempi di esecuzioni più brevi di J48 e risultati migliori, ma è comunque peggiori di DMNBText; L'algoritmo migliore è infatti DMNBText, in quanto restituisce i risultati migliori e lo fa in tempi molto brevi;
- La presenza o meno dello stemmer non è influente; questo può essere dovuto dalle tante parole con etimologie diverse contenute in questo dataset;
- Il parametro DoNotOperateOnPerClassBasis è decisamente uno dei più interessanti. Settando questo parametro a False infatti si nota un netto miglioramento nella creazione del modello in tutti gli algoritmi. Ciò è dovuto dal fatto che vengono selezionati due insiemi di parole, una per ciascuna classe. Questi insiemi di parole saranno quindi i più caratterizzanti per le due classi, e aiuteranno il classificatore ad ottenere dei risultati migliori;
- Diminuire il valore di WordsToKeep fa peggiorare i classificatori. Il minor numero di attributi creati infatti sfavorisce i classificatori, che dovendo utilizzare meno attributi per la costruzione del modello, non riescono a definire bene le due classi;
- L'introduzione dell'AttributeSelector invece, pur diminuendo il numero di parole selezionate, riesce a far ottenere a tutti i classificatori dei risultati migliori rispetto ai risultati ottenuti non utilizzandolo. Selezionando gli attributi più significativi infatti, aiuta il classificatore a distinguere meglio le due classi, caratterizzate ovviamente da parole diverse.

Negli esempi sopracitati, il filtro StringToWordVector è stato applicato all'intero dataset, che in seguito è stato diviso in training e test set prima della classificazione. In questo modo ho selezionato feature basandosi

anche sui dati per testare il modello. In un caso reale, documenti e dati da classificare sono di solito ignoti nella fase di training, infatti ogni trasformazione applicata al training set è inferita solo sulla base dei dati presenti nel training set stesso. Per classificare nuovi documenti quindi, le stesse trasformazioni si applicano ad ogni documento da classificare per renderli compatibili con il modello. Per fare ciò, in Weka si può utilizzare il FilteredClassifier, che applica un filtro ai dati di training, genera un modello di classificazione dal training set trasformato, applica il risultato del filtro di trasformazione al test set, ed infine classifica il test set.
Sono quindi stati scelti i migliori settaggi per ogni classificatore, ed è stato applicato ad ognuno di essi l'algoritmo FilteredClassifier.

ALGORITHM	ACCURACY	PRECISION		RECALL		F-MEASURE	
		Pos	Neg	Pos	Neg	Pos	Neg
Filtered j-48 stemmer -> SI WordsToKeep -> 1650 DoNotOperate.. -> False AttributeSelector -> NO	67.85	0,686	0,672	0,659	0,698	0,672	0,685
Filtered DMNBText stemmer -> SI WordsToKeep -> 1650 DoNotOperate.. -> False AttributeSelector -> NO	83.9	0,836	0,842	0,843	0,835	0,840	0,838
Filtered SMO stemmer -> SI WordsToKeep -> 1650 DoNotOperate.. -> False AttributeSelector -> NO	81.2	0,806	0,818	0,822	0,802	0,814	0,810

Si può notare come I risultati siano leggermente peggiori rispetto a quelli mostrati in precedenza; d'altro canto saranno sicuramente più attendibili, in quanto rispecchiano meglio gli scenari d'uso reali.

5.2 CLASSIFICAZIONE NON SUPERVISIONATA IN R

Oltre ai classificatori citati sopra, è stato implementanto in R una funzione di scoring per valutare le recensioni solo sulla base di ciò che contenevano, senza addestrare prima un classificatore. A questo proposito è stata creata una funzione di scoring:

```
score.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
        require(plyr)
        require(stringr)
        scores = laply(sentences, function(sentence, pos.words, neg.words) {
        sentence = gsub('[^A-z ]','', sentence)
        sentence = tolower(sentence)
        word.list = str_split(sentence, '\\s+')
        words = unlist(word.list)
        pos.matches = match(words, pos.words)
        neg.matches = match(words, neg.words)
        pos.matches = !is.na(pos.matches)
        neg.matches = !is.na(neg.matches)
        score = sum(pos.matches) - sum(neg.matches)
         return(score)
```

```
  }, pos.words, neg.words, .progress=.progress )
        scores.df = data.frame(score=scores, text=sentences)
        return(scores.df)
}
```

Questa funzione di scoring estrae I caratteri alfabetici dale recensioni, li converte in lowercase i li splitta in parole. A questo punto confronta le parole con Il dizionario di opinion distribuito da Hu e Liu (http://www.cs.uic.edu/~liub/FBS/sentimentanalysis.html), contenente circa 7000 parole tra positive e negative. A questo punto calcola lo score come differenze tra il numero di parole positive ed il numero di parole negative trovate nella recensione. Questo punteggio verrà quindi utilizzato per valutare la classe di appartenenza del testo della recensione: se il punteggio è maggiore o uguale a 0, verrà valutata come positiva, in caso contrario come negativa.
I risultati ottenuti sono i seguenti:

```
FALSE : 602
TRUE : 1398
ACCURACY: 0.699
POSITIVE PRECISION: 0.674
POSITIVE RECALL: 0.7094737
NEGATIVE PRECISION: 0.724
NEGATIVE RECALL: 0.6895238
```

I risultati ottenuti sono discreti; questa funzione ovviamente dipende molto dalla bontà del dizionario di termini utilizzato, ed inoltre non riesce a cogliere le il senso delle frasi, dovendosi basare solo sulle parole non su come vengono utilizzate all'interno della frase(es. sarcasmo). Inoltre nella realtà la presenza di più parole positive rispetto a quelle negative non significa necessariamente che si tratti di una recensione positiva.

5.3 CONCLUSIONI

In conclusione si può veder come i classificatori riscontrino qualche difficoltà nel valutare un record solo attenendosi ai campi testuali contenuti in esso, senza utilizzare altri attributi, che in questo dataset non erano disponibili. La classificazione supervisionata, pur non avendo raggiunto risultati ottimi, si è dimostrato uno strumento potente e veloce per la classificazione. I risultati ottenuti sono comunque accettabili, dato che nel caso migliore si raggiunge un'accuratezza del 84.9%, supportata da una precision ed una recall anch'esse di buon livello.

6. LATENT SEMANTIC ANALYSIS

L'ultimo passo del processo di analisi sul dataset consiste nell'estrarre ed interpretare un pattern che caratterizzi particolarmente le recensioni negative presenti nel dataset. L'approccio seguito prevede un'analisi che sfrutta la Latent Semantic Analysis per ridurre la dimensionalità e fare emergere i termini più correlati alle recensioni con giudizio negativo. Per fare ciò è stato utilizzato l'ambiente R Studio.
Il primo passo dei questa fase è la creazione della matrice termini-documenti. Per fare ciò le recensioni vengono prima trasformate sotto forma di file csv, ed in seguito caricate nell'ambiente di lavoro.

	review_text	review_class
1	plot : two teen couples go to a church party , dr>	neg
2	the happy bastard's quick movie review \ndamn tha>	neg
3	it is movies like these that make a jaded movie v>	neg
4	" quest for camelot " is warner bros . ' first f>	neg
5	synopsis : a mentally unstable man undergoing psy>	neg
6	capsule : in 2176 on the planet mars police takin>	neg

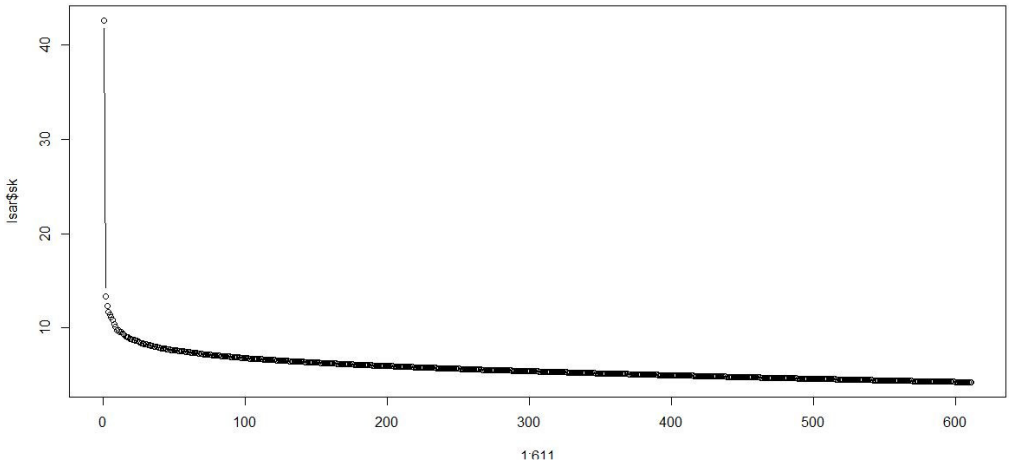
A questo punto, dopo aver caricato il file contenente il dizionario di stop words, vengono eliminati i simboli di punteggiatura, tutte le lettere vengono portate in minuscolo e le stop words vengono eliminate.Si ottiene così questa matrice:

```
<<TermDocumentMatrix (terms: 4105, documents: 2000)>>
Non-/sparse entries: 313134/7896866
Sparsity           : 96%
Maximal term length: 17
Weighting          : term frequency (tf)
```

A questo punto la matrice contiene le occorrenze dei termini, e per migliorarne il contributo informativo si applicano metodi di term weighting come prodotto di due fattori: la term frequency log e l'entropia del termine, per avere un fattore locale(basato solo sui termini presenti nello stesso doc) ed uno globale(fattore che tiene conto anche degli altri doc nel corpus).
Decomponendo ora la matrice termini-documenti tramite LSA, è possibile portare i termini e le recensioni in uno spazio comune, le cui dimensioni sono variabili latenti. Applicando LSA in R alla matrice quindi, si ottengono le 3 matrici:

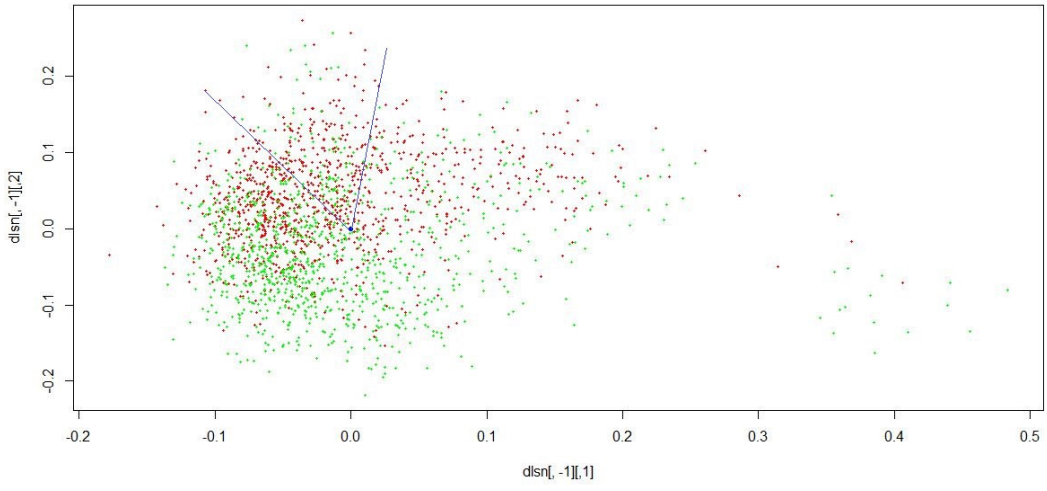
- Termini-variabili latenti(U);
- Documenti-variabili-latenti(V);
- Valori singolari(Σ);

La matrice dei valori singolari presenta 611 dimensioni.

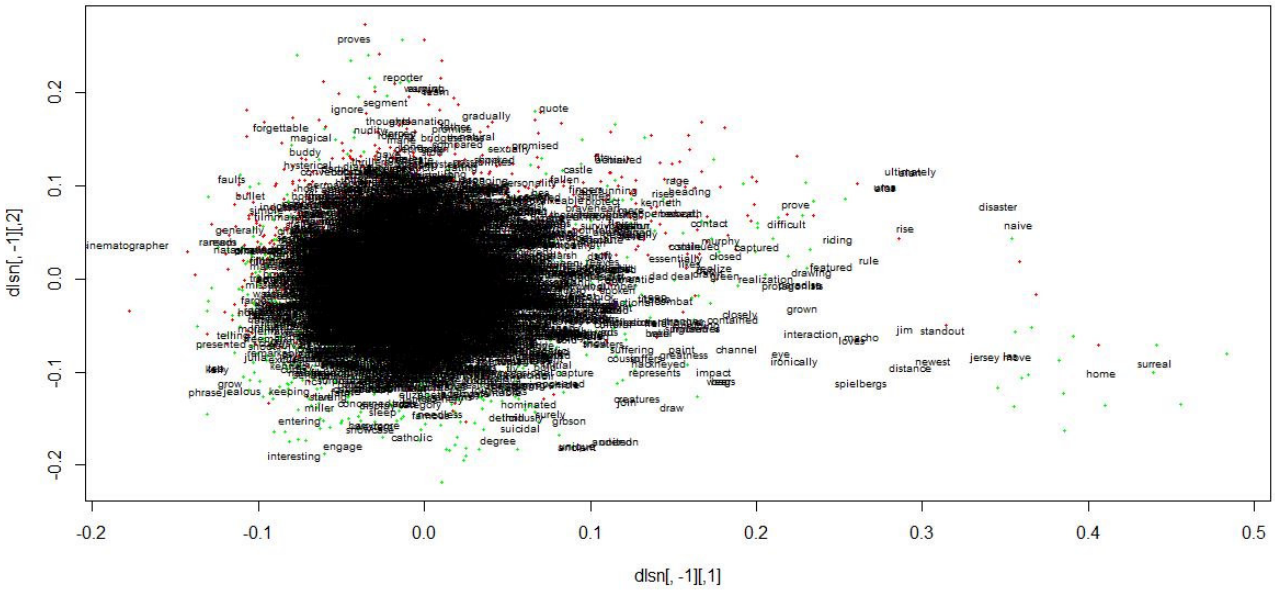


A questo punto è possibile proiettare le recensioni nello spazio latente e colorare i punti in base alla seguente classificazione:

- Rosso -> recensioni negative;
- Verde -> recensioni positive;



Analizzando il grafico sembra esserci una zona dello spazio LSA(delimitata dalle rette blu) in cui si concentrano tendenzialmente recensioni negative. Analizzando i termini presenti in questa zona sarà possibile identificare e interpretare un pattern che caratterizza maggiormente tali recensioni.



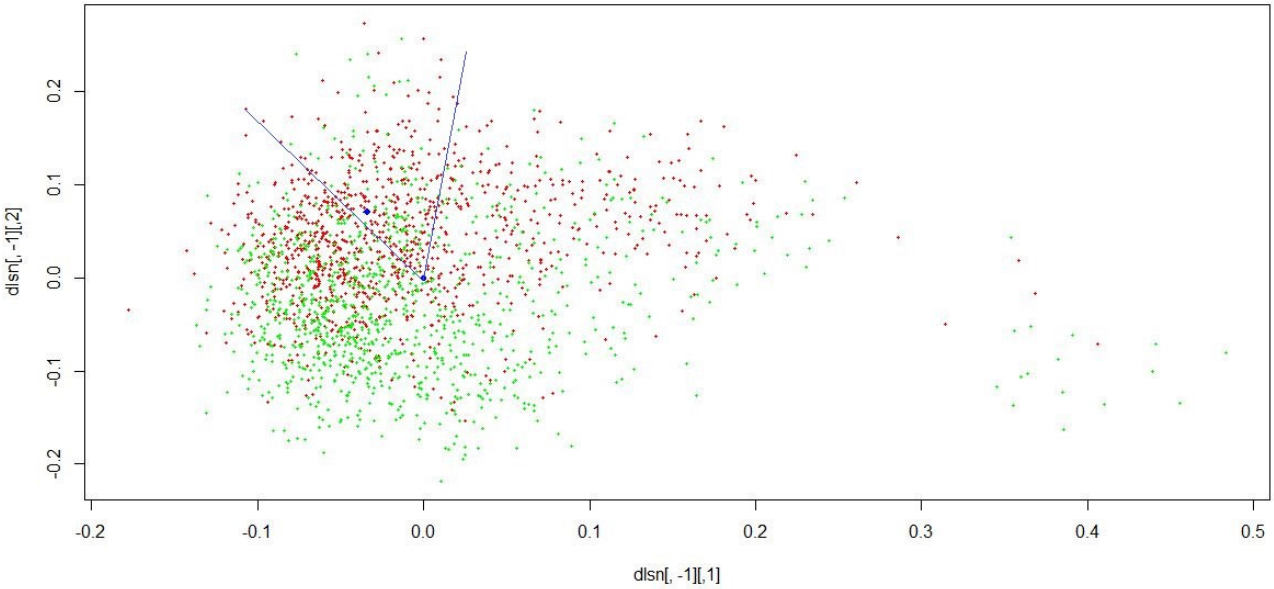
Tra i termini più rappresentativi è stato individuato il termine “script”.
Facendo una prima verifica di correlazione, tra la presenza esatta del termine “script” nel testo e le recensioni di classe “neg” tramite test Chi Quadro, si ottiene conferma della nostra ipotesi:

```
data:  script.vs.neg
X-squared = 40.964, df = 1, p-value = 1.551e-10
```

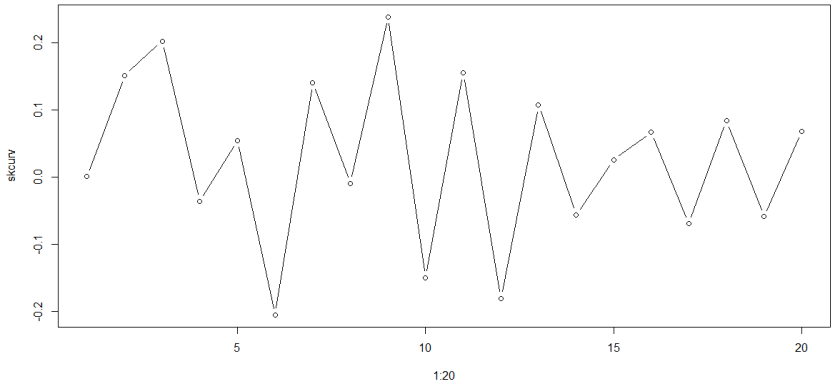
Ciò significa che la probabilità che il termine “script” non sia correlato alla classe “neg” è di 1.551e-10.
Ora cercando i termini che nello spazio hanno maggiore similarità coseno(>0.99) rispetto menagement otteniamo che il migliore è “played”.
A questo punto si verifica sempre tramite test Chi Quadro che la coppia “script played” sia oggettivamente correlata alla classe “neg”:

```
data:  played_script.vs.neg
X-squared = 21.409, df = 1, p-value = 3.71e-06
```

Ossia la prob. che la coppia di termine adverse weather NON sia lessicalmente correlata agli incidenti destroyed è 3.71e-06.
A questo punto ripetiamo l'analisi, ma questa volta nello spazio LSA, per stabilire se esiste oggettivamente correlazione semantica tra la coppia di termini "script played" e gli incidenti distruttivi.
In questo caso i documenti considerati potrebbero anche non contenere i due termni, ma termini semanticamente correlati ad essi. Per fare questo trasformiamo la query "script played" in un documento e facciamo il fold_in nello spazio.



La query risulta dentro al cono definito in precendenza. A questo punto possiamo far restituire i 10 documenti che risultano più semanticamente simili alla nostra query. Tra questi però possiamo trovare ancora documenti dove la negatività della recensione non dipende dal copione: es. “this film is extraordinarily horrendous and i'm not going to waste any more words on it”.
Scegliamo quindi uno spazio LSA con piu' dimensioni e quindi con minore perdita d'informazione. Per fare ciò individuamo il punto di knee nella sequenza degli autovalori tra i minimi locali:



Il primo minimo locale e' 4, perciò selezioniamo le prime 4 dimensioni dello spazio LSA ed estraiamo da questo spazio i primi 10 doc semanticamente piu' simili alla query. Ora in ogni doc visualizzato le recensioni indicano cattivi copioni e interpretazioni non buone degli attori.

Per determinare se esiste una correlazione tra “script played” e recensioni negative con un modello di ricerca semantico basato su ranking, occorre, a differenza del modello di ricerca booleano basato su match lessicale analizzato in precedenza, fissare il numero massimo di risultati restiuiti dal modello di ricerca da considerare (R-precision). Questo numero e' dato dalla numerosita' delle recensioni negative nell'intero dataset delle recensioni. Calcoliamo percio' la distribuzione della classe delle recensioni nei primi 1000 risultati(numero di recensioni negative) della ricerca. Vengono quindi restituite 675 recensioni negative sulle 500 attese. Per verificare se ciò indica o meno correlazione tra “script played” e recensioni negative, si svolge una verifica oggettiva mediante test Chi-Quadro. Il risultato è il seguente:

```
data:  psq.vs.neg
X-squared = 245, df = 1, p-value < 2.2e-16
```

In conclusione si può vedeer come il pessimo copione o la cattiva interpretazione da parte degli attori contribuisca a far ricevere circa il 67% delle recensioni negative.