Code explanations
Elad Kapuza, Shir Shtinits, Hillel Merran

SVM

Function	Description	Parameter	Return
RemoveHashTag	Remove '#' from an Hash tag. This function is used to decompose it	word - a word (string)	the word without '#'
SplitCamelWordIntoList	Split a camel case word into words list e.g HelloWorld into [Hello, World]	camelWord - a camel case word (string)	List of words from the camel case word
CreateCorpusAndPostListsAndTagList	Create a corpus of termes, a list of posts, of tags, from the data posts	**corpus** (return) - list of all the words in the data **postsList** (return) - list of posts, each post is a list of words **tagList** (return) - list of tags, each tag is in {0: non-racist, 1: racist} **stopWordsList** - a list of stopwords to remove from the posts	-
PostToVector	Convert a post to a vector for the learning algorithm	**corpus** - list of all words in the data **post** - the post to convert. the post is a list of words	the vector which represents the post
StringToVector	Convert a post to a vector for the learning algorithm	**corpus** - list of all words in the data **stopWordsList** - list of stopwords to remove from the post **str** - a post, as a string	the vector which represents the post
CreateVectors	Convert a list of posts to a list of vectors	**corpus** - list of all words in the data	-

		postsList - list of the posts to convert. each post is a list of words **vectorsList** (return) - list of vectors. each vector represents a post	
main	• create a stemmed corpus with words from the data • create a list of all posts and a list of all tags • create the vectors from the posts • run SVM with Cross-Validation (5-folds)	-	-

Optimization

Function	Description	Parameters	Return
sum_of_features	Calculate the sum of the different clique features vectors of a sentence	**y** - tag of a post. y = (post_tag, [s1_tag, ..., sK_tag]) **s** - a post (list of sentences). s = [s1_string, ..., sK_string] **n** - length of the features vectors	the sum of the vectors
Hamming_loss	Calculate the Hamming loss of a tag. Namely how many coordinates of 2 binary vectors are	**w** - a first tag. w = (post_tag, [sentence1_tag, ... , sentenceK_tag]) **z** - a second tag. z = (post_tag, [sentence1_tag, ... , sentenceK_tag])	the Hamming loss
argmin	Resolve the 4th step in MIRA algorithm. That is calculate the weight vector for iteration i+1	**w** - original weights vector **y** - real tag of a post. y = (post_tag, [sentence1_tag, ..., sentenceK_tag]) **s** - a post (list of sentences). s = [s1_string, ..., sK_string]	weight vector for next iteration

Naïve Bayes

Function	Description	Parameters	Return
simplify_sentence	Convert a sentence (string) to a list of stemmed words, without stopwords	**row** - a sentence (string) **stopWordsList** - a list of stopwords	the list of the words of the simplified sentence
main	create a list of all posts and tags run Naive-Bayes Classifier with Cross-Validation (5-folds)	-	-

MEMM

Function	Description	Parameters	Return
simplify_sentence	Convert a sentence (string) to a list of stemmed words, without stopwords	**row** - a sentence (string) **stopWordsList** - a list of stopwords	the list of the words of the simplified sentence
clique_score	Calculate the score of a tag for a clique	**s1, s2** - sentences of the clique (string) **post_tag** - tag of the post {0: non-racist, 1: racist} **tag1, tag2** - tags of s1, s2 {-1: anti-racist, 0:neutral, 1:racist} **stopWordsList** - a list of stopwords	the score of the clique
score	Calculate the score of a tag for a post (and sentences)	**weights** - vector of weights, one for each feature **y** = (post_tag, [sentence1_tag, ..., sentenceK_tag]) - tag of the post **s** = [sentence1, ..., sentenceK] - list of sentences (string) of the post	the score of the clique
max_clique_score	Determine the tag of the first sentence in a clique which have the highest clique score	**weights** - vector of weights, one for each feature **clique_no** - index of the clique **current_2nd_sentence_tag** - tag of the 2nd sentence of the clique **sentence1** - first sentence of the clique (string) **sentence2** - second sentence of the clique (string) **stopWordsList** - list of the stopwords **viterbi** - dictionary for Viterbi algorithm. {key=state: value=score}	**backpointer** - tag of the first sentence which maximize the score of the clique **score** - maximum score of the clique

argmax	Determine the tag of the sentences given the tag of the post using Viterbi algorithm	**weights** - vector of weights, one for each feature **d_tag** - tag of the post {0: non racist, 1: racist} **s** - list of the sentences (string) of the post	list of tags. each tag is in {-1: anti-racist, 0: neutral, 1:racist} and correspond to a sentence
classifier	Determine the tag of the post and sentences given the tag of the post using Viterbi algorithm	**weights** - vector of weights, one for each feature **s** - list of the sentences (string) of the post	tag of the post and sentences - (post_tag, [s0_tag, ... , sM-1_tag])
main	create a list of all posts and a list of their tag run the MIRA algorithm to learn the vector of weights test the model with Cross-Validation (5-folds)		

Features

Function	Description	Parameters	Return
StemWord	Convert a word to its stemmed form	word - a word (string)	The stemmed form of the word
CreateRacistStemmedCorpus	Create a corpus of racist stemmed word from csv file	-	Set of racist words
CreateAntiRacistStemmedCorpus	Create a corpus of anti-racist stemmed word from csv file	-	Set of anti-racist words
RemoveHashTag	Remove '#' from an Hash tag. This function is used to decompose it	word - a word (string)	the word without '#'
SplitCamelWordIntoList	Split a camel case word into words list e.g HelloWorld into [Hello, World]	camelWord - a camel case word (string)	List of words from the camel case word
simplify_sentence	Convert a sentence (string) to a list of stemmed words, without stopwords	**row** - a sentence (string) **stopWordsList** - a list of stopwords	the list of the words of the simplified sentence
CreateStemmer	Create stemmer object	-	Stemmer object
Similarity	Calculate sumilarity value between two sentences by counting common words	**sentence1** - first sentence (list of words) **sentence2** - second sentence (list of words)	Similarity value
IsSimilarityLow	Check if the similarity level between two sentences is low	**sentence1** - first sentence (list of words) **sentence2** - second sentence (list of words)	True if the similarity level is low and false otherwise
IsSimilarityHigh	Check if the similarity level between two sentences is high	**sentence1** - first sentence (list of words)	True if the similarity level is high and false otherwise

		sentence2 - second sentence (list of words)	
ContainsExclamationMark	Check if the sentence contains exclamation mark	sentence - sentence (string)	True if the sentence contains exclamation mark and false othrewise
BothContainExclamationMark	Check if the both two sentences contain exclamation mark	**sentence1** - first sentence (string) **sentence2** - second sentence (string)	True if the both two sentences contain exclamation mark and false othrewise
ContainsRacistWord	Check if the sentence contains a racist word	**sentence** - sentence (list of words) **racistWord** - racist word (string)	True if the sentence contains the racist word
ContainsAnyRacistWord	Check if the sentence contains any racist word from the racist words corpus	sentence - sentence (list of words)	True if the sentence contains any racist word
ContainsAntiRacistWord	Check if the sentence contains an anti-racist word	**sentence** - sentence (list of words) **antiRacistWord** - anti-racist word (string)	True if the sentence contains the anti-racist word
ContainsAnyAntiRacistWord	Check if the sentence contains any anti-racist word from the anti-racist words corpus	sentence - sentence (list of words)	True if the sentence contains any anti-racist word
Polarity	Calculates the polarity of the sentence (sentiment level between -1 to 1)	sentence - sentence (string)	Polarity value
IsPositiveSentimental	Check if the sentence has a positive sentiment	sentence - sentence (string)	True if the sentence has a positive sentiment and false otherwise
IsNegativeSentimental	Check if the sentence has a negative sentiment	sentence - sentence (string)	True if the sentence has a negative sentiment and false otherwise

Subjectivity	Calculates the subjectivity level of the sentence (between 0 to 1)	sentence - sentence (string)	Polarity value
IsSubjective	Check if the sentence is subjective	sentence - sentence (string)	True if the sentence is subjective and false otherwise
IsObjective	Check if the sentence is objective	sentence - sentence (string)	True if the sentence is objective and false otherwise
HasAdjective	Check if the sentence contains adjectives	sentence - sentence (string)	True if the sentence contains adjectives and false otherwise
clique_to_features	Creates list of features with values of 0 or 1	**post_tag** - The true tag value of the post (int) **s1_tag** - The true tag value of a sentence in the post (int) **s2_tag** - The true tag value of the following sentence in the post (int) **s1** - A sentence in the post **s2** - The following sentence in the post **stopWordsList** - Corpus of stop words	List of features with values of 0 or 1