Venses @ HaSpeeDe2 & SardiStance: **Multilevel Deep Linguistically Based Supervised Approach to Classification**

Rodolfo Delmonte
Dipartimento di Studi Linguistici e Culturali
Comparati
Ca' Bembo – Dorsoduro 1075 – Università Ca'
Foscari – 30131 Venezia
delmont@unive.it

Abstract

In this paper[1] we present the results obtained with ItVENSES a system for syntactic and semantic processing that is based on the parser for Italian called ItGetaruns to analyse each sentence. In previous EVALITA tasks we only used semantics to produce the results. In this year EVALITA, we used both a fully and mixed statistically based approach and the semantic one used previously. The statistic approaches are all characterized by the use of n-grams and the usual tf-idf indices. We added another parameter called the Kullback-Leibler Divergence to compute similarities. In addition we used emoticons and hashtags. Results for the two runs allowed have been fairly low – around 40% F1-score. We continued producing other runs on the basis of the statistical approach and after receiving the gold-test version and the evaluation script we discovered that in one of these additional runs - the fourth - we improved up to 54% macro F1 for HaSpeeDe2 task and up to 48% macro F1 for Sardines.

1 Introduction

In this paper we will present work carried out by the Venses Team in Evalita 2020 (Basile et. 2020). We will comment in the following both on the Sardines Task (Cignarella et al., 2020) and on the HaSpeeDe2 Task (Sanguinetti et al. 2020). The reason for this is discussed in the sections below, but it has been basically determined by the overlapping in the choice of the features to adopt for the classification tasks. To show how the two tasks share part of the features we created a table where we compare the output of the first step in the process, i.e. the creation of a frequency list dictionary. The frequency list that we show in Table 1. below is made of nominal entities that were extracted automatically from the total frequency list. We call this frequency list *InstanceList* and the position occupied by each entry as *InstanceListPosition* and the Rank as *InstanceRank*. In the first column we indicate rank; in the following two columns we report the word/s preceded by its frequency value. In the second couple of columns, column no. 4 and 5 we make a comparison between the two corpora based on frequency lists and the rank each entry has received.

We use three types of values: the frequency value from the general frequency list derived from the corpus; the rank position in the InstanceList in case the word appears in both InstanceLists; and the word "nil" in case the entry is not present in the general frequency list of the comparing corpus. In column 4 the comparison is made between the first list (HaSpeeDe2) and its instances and the second list (SardiStance). Every word is associated to the rank in the InstanceList and a second element which can be one of three: the position in the second list if available; the position in the general FrequencyList of the compared corpus; nil in case the word is not present.

Corpus/ FreqRank	HaSpeeDe2 (H)	SardiStance (S)	Comparison FreqOrder H vs S	Comparison FreqOrder S vs H
1	1567-rom	1807-sardine/ 142-sardina/ 97-'6000sardine'	Rom-1-nil	Sardine-1-nil
2	1308-migranti	325-salvini	Migranti-2-nil	Salvini-2-15
3	1046-italia	222-piazza/76-piazze	Italia-3-5	Piazza-3-479
4	985-immigrati	205-PD	Immigrati-4-1555	PD-4-23
5	624-roma	150-sinistra	Roma-5-33	Sinistra-5-117
6	609-italiani	133-politica	Italiani-6-168	Politica-6-256
7	334-campo/ 211-campi	129-movimento	Campo-7-733	Movimento-7-1980
8	454-immigrazione	113-italia	Immigrazione-8-2060	Italia-8-3
9	462-stranieri	108-bologna	Stranieri-9-1784	Bologna-9-1006
10	413-islam	95-lega	Islam-10-3099	Lega-10-258
11	369-casa	75-partito	Casa-11-219	Partito-11-600
12	316-clandestini	72-governo	Clandestini-12-729	Governo-12-115
13	310-profughi	61-emilia	Profughi-13-nil	Emilia-13-14822
14	310-terroristi	60-persone	Terroristi-14-nil	Persone-14-194
15	312-salvini	60-bibbiano	Salvini-15-2	Gente-15-200
16	290-nomadi	60-gente	Nomadi-16-nil	Bibbiano-16-nil
17	246-musulmani	58-paese	Musulmani-17-6528	Paese-17-20
18	254-islamici	54-popolo	Islamici-18-nil	Popolo-18-292
19	214-milano	54-bonaccini	Milano-19-323	Bonaccini-19-nil
20	207-paese	51-giovani	Paese-20-115	Giovani-20-348
21	201-europa	50-m5s	Europa-95-121	M5s-21-336
22	201-bene	48-destra	Bene-96-137	Destra-22-1530
23	173-pd	47-bene	Terrorismo-108-1765	Bene-23-22

Table 1. Instance list of features created automatically from the dictionary of unique wordform for two tasks

For instance, we can see that the words *rom, migranti, profughi, terroristi, nomadi, islamici/rom, migrants, refugees, terrorists, nomads ,islamists* are not present in the second list and so they characterize the first corpus (HaSpeeDe2) as being different from the Sardines one, specializing it in a particular list of topics or keywords. When we look at column 5, where the comparison is made in reverse order, we discover that *sardine, bibbiano, bonaccini/sardines, bibbiano, bonaccini* are not present in the first list. Most importantly we discovered that the most frequent words of the two lists are not shared, "rom", in list 1, and "sardine" in list two. In the sections below we present the module for supervised automatic classification and the experiments that we devised using basically two approaches: a semantic approach vs a statistic approach.

In Table 2 and 3 we report the subdivision into classes of the two training and test corpora for the two tasks, SardiStance and HaSpeeDe2 with percent values to allow for comparisons. As can be noticed in the SardiStance corpus the majority class is constituted by AGAINST, followed by FAVOR and then NONE. In the Test set, the distribution into the three classes favors AGAINST and for the other two classes is almost identical. The same happens in the other corpus, the HaSpeeDe2, where we notice a majority of occurrencies for the NULL class in the Training corpus. In the Test set, this is still valid but we see an important increase of the *BothHateandStereo* class and a strong reduction of the Stereo class. Of course, these differences in class distribution may have influenced the final outcome, in case as it is ours - there is a default choice at the end of the computation for each

tweet class. Here below some general quantitative information for the two corpora:

Corpus/ Class	HaSpeeDe2 Abs.Val.	HaSpeeDe2 Percent
NULL	3,049	44.5825%
OnlyHATE	748	10.9372%
OnlySTEREO	1,024	14.9729%
BothHATEAnd STEREO	2,018	29.5072%
Totals	6,839	100%

Table 2. Distribution of Classes for HaSpeeDe2 Tweets training and test corpora

Corpus/ Class	Sard Train	Sard. %	Sard. Test	Sard %
NONE	515	24.15	172	15.49
AGAINST	1,028	48.21	742	66.84
FAVOR	589	27.66	196	17.65
Totals	2,132	100%	1110	100%

Table 3. Distribution of Classes for SardiStance training and test corpora

2 The Module for Supervised Automatic Classification

We present the modules for automatic classification that uses ***three different approaches***: a fully BOW and statistic one, a fully semantically based one, and a mixed both bag-of-words and (partially) semantically-based one. With the exception of the fully semantic approach, the remaining approaches are however characterized by the use of n-grams and a fully supervised method to create the model. In all approaches the model is created on the basis of an automatically built dictionary of unique wordforms sorted by frequency where the first most frequent 25 nominal expressions are chosen as supplied instances for n-grams construction.

Eventually, we created six different classifiers that we will present in the sections below. They are a fully semantic classifier, a lexically-based semantic classifier, a mixed statistic and lexical semantic classifier using supervised n-grams, a fully statistic tf-idf classifier based on differences, a fully statistic Kullback-Leibler Divergence (hence KLD) classifier based on differences, a classifier based on emoticons and on hashtags.

First approach.

We will start by describing the lexically-based semantic classifier. This is used for both tasks but in a different manner. Whereas in the semantic classifier it is treated as an important component of the evaluation module, it becomes just a default classifier in the statistic classifiers, in case of failure of the previous ones. It is organized into a grid with seven slots:

[**Polarity, Appraisal, NegativeW, PositiveW, SwearW, HateW, StereoW**]

Polarity is computed at a propositional level by the deep parser and is described below. The remaining slots are all lexically processed. In particular Appraisal Classes are derived from previous work on political newspapers (Stingo and Delmonte, 2016); Swear Words, Negative and Positive Words are derived from previous work on opinion and sentiment analysis and were used in SenticPol (Delmonte, 2014); finally HateWords and StereoWords were collected from the HurtLex made available by the organizers, proceeding by a manual selection of Italian words and discarding all English words.

The second approach that we call semantically-based, uses a three levels of classification. Besides using an n-gram model, it uses a majority vote approach based on presence of emoticons previously classified on the basis of the training set. The most important module is fired in case of failure (no n-gram available to match) in the two previous steps and is totally based on semantics. It builds an interpretation from deep semantic analysis evaluating presence of appraisal theory labeled items, presence of hate/stereotype items from lexical lookup and their propositional level semantics. In the sections below we describe in details the three level classification module. This approach covers 93% of the whole training set – but see below. However its predicting power is not so great.

Third Approach.

The bag-of-words approach associates a numerical parameter to each word and the resulting sum for the each tweet. At first we uses TF-IDF as the mathematical formula for characterizing each word occurrence and each tweet. We applied TF-IDF to each word in each tweet and used the output to map the indices to n-grams and produce a model. Then we used this model to predict the similarity with n-grams obtained from the held out development set of tweets. The results were however very poor, 20% accuracy, which added to 12% obtained from the emoticons model made a 32% final accuracy.

We assumed the reason was that tweets are too short to be useful for term-frequency computation. In the majority of the cases wordforms ap-

peared only once in each document/tweet – apart from stop words. So we searched a formula which could be better suited for this task and could represent both frequency and dispersion at corpus level. We found it in a number of papers published by Gries (2008, 2020), but also in a paper online by Koos Wilt. The important part of the formula regards the role of frequency of occurrence in the total corpus which is used to produce TF so that it would resemble a probability of occurrence and the concept of entropy[2]. Gries defines this formula as a way of characterizing "keyness" by including dispersion information. To do that he augmented frequency information by using the Kullback-Leibler Divergence. Wordforms can become key not only for their frequency of occurrence, their dispersion or both. The formula is able to "tease apart distributional differences".

p = frequency of w in document A of the corpus / divided by total frequency of w in the corpus
q = total number of tokens in document A of the corpus / divided by total number of tokens in the corpus

KLD = p X log(p/q) ➔ ∑ p X log(p/q)

In the same paper Gries suggests to compute keyness also to n-grams besides multiword expressions and this is what we did. The summation applies to the document/tweet and is used to differentiate each tweet from one another and produce a similarity or distance evaluation. We proceeded as before to verify the predictive ability of this new formula and came out with 44/45% accuracy, a 12% gain.

3 The Semantically-Based Module And The N-gram Models

The general procedure we organized for the three approaches is as follows.
At first we massaged the text in order to obtain a normalized version – wrong word accents like "nè" instead of "né" etc. The text is then turned into an xml file to suit the Prolog input requirements imposed by the system. It is then precompiled by a set of regular expressions: we separate the hash symbol # from its tag; we separate the @ symbol from the following username; we cancel the word URL; we separate all punctuation marks from a preceding or following word; then we lowercase all words and produce a sorted list which is then used to count frequencies associated to each wordform and produce the dictionary of unique wordforms or types.
Then we choose the first 25 nominal entities from the list erasing generic or general nouns like "person", "people" etc. The final list of features is treated as supplied instances to search for the construction of n-grams from 4-gram up to 8-grams: we take all sequences of four/eight tokens where the ending or beginning word must be taken from the list of instances. If eight is not available we accept down to 4-grams. Instances are collapsed under three unique general topic which are the following ones: racism, politics, sardines/Salvini.
Since we process each tweet using lemmata in every approach, we do sentence splitting and tagging. Every tagged token is then lemmatized and in the semantically-based approach it is subsequently associated to a lexically validated three-valued sentiment label.
In the semantically-based approach, we then compute syntactic constituency and dependencies for every sentence. This information is passed to the semantic processor which produces predicate argument structures for every sentence present in each tweet. In case no punctuation is available and the sentence is longer than 40 tokens we activate an empirical set of rules to insert punctuation and divide the tweet into sentences by checking the presence of words starting with uppercase letter and not being a Named Entity. If the sentence splitter fails we activate a search for sentence level coordinating or subordinating conjunctions. Many tweets are just fragments and contain a list of nouns and adjectives: we add a dummy verb ESSERE/to_be in order to allow the semantics to work.
Propositional level semantics is made by the computation of factivity, negation, subjectivity, modality, speech_act, diathesis, which then produce a fixed set of semantic labels to allow for a correct interpretation.
In the mixed approach and in the statistics-only approach we procede as follows. Before producing n-grams, we erase punctuation with the exception of the hash symbol that informs the system of the presence of an hastag or a slogan. Similarity is computed by matching every lemma

[2] According to Wilt Koos, ibid. pag.2: "Classification according to the KLD takes place on the assumption the training set reflects order and the test set, a document to be categorized, reflects a deviation from this order and is therefore chaotic or entropic. The lower the entropy regarding the training set, the more likely it is a given test set belongs to that training set. "

from two n-grams labeled with the same main topic. We established a ratio of 0.3 as the threshold for acceptance, but then we check the semantics be identical or very similar. We assume with Emily Bender that "a system trained on form alone cannot in principle learn meaning"[3] . So we use an approach with is based partially on bag-of-words n-grams – using frequency lists and n-grams - but we associate semantic interpretation to every n-gram of the model. Semantics is used to verify and confirm the first approximation of a similarity measure based on wordforms[4] and lemmata. We assume that n-grams belonging to a statement cannot possibly be regarded to have the same meaning in case the comparison is made with an n-gram extracted from a proposition which has negation at propositional level.

4 The Experiment and the Evaluation Module of *ItVenses*

We organized our classifiers to produce two runs as required by the two tasks, SardiStance and HaSpeeDe. However, we then realized that we needed to produce more runs in order to take into account all variables involved in the statistically-based module. Eventually we had to choose one modality for the single run with the statistical module trusting the results obtained from the Development set as described here below.

To produce a development set we held out 20% of all training corpus - 427 tweets for SardiStance and 1000 tweets for HaspeeDe2 - that we called devtset and remodulated the n-gram model accordingly by subtracting the n-grams related to the same sequence of tweets.

For HaSpeeDe2 the system produced 23,000 n-grams for the training corpus and 19,738 for the development. The development set is made of 1,000 tweets held out from the total 6839 which adds up to 136,536 tokens.

For SardiStance, we have 4,993 n-grams from the training corpus and 4,003 for the development: the development set is made of 427 tweets held out from the total 2,132 tweets, adding up to 57,774 tokens.

The system takes as input the analysis of one tweet at a time. In the mixed semantic-statistic module, the multilevel evaluation process consists of four steps which take advantage of the following previously compiled analyses. We have a full-fledged semantic analysis at propositional level; a trivalued labeling of each word-lemma by lexically-driven sentiment dictionaries; a six slot analysis of ironic/sarcastic contents at tweet level; a model for emoticons; a list of special hashtags inducing a direct evaluation. This is what we use in the semantic-only approach. The evaluation process is performed recursively for each tweet, and starts by searching for presence of Emoticons extracted in the previous analysis and organized in a model: in this case, the decision is taken by majority vote based on the type of emoticons present in the tweet. As for the semantic-only module, the problem was how to select best candidate from the pool of model n-grams with different value labels. We solved this problem by a scoring procedure. We produced two levels of scoring: a first one based on the number of sentiment labels with positive/negative value producing as a score a ratio of the total number divided by total number of words in the n-gram. Negative words are valued the double. The second scoring analysis is based on the contents of the propositional level semantics: here we associate 0.25 for each proposition marked differently from statement; another 0.25 is added for presence of predicates different from "dummy" verb ESSERE; eventually another 0.25 is added in case one of the arguments or attributes is shared with the input n-grams.

Eventually, we imposed coincidence at the level of Discourse Class associated to the utterance. We use seven different labels: statement, question, exclamation, negated, unreal, opinionsubjective, conditional.

4.1 Creating and Accessing N-grams models

If the semantics-only method needs just words from the two tweets to be evaluated by means of linguistic parameters, the two other methods or approaches we used are based on n-gram models which introduce a great number of variables. First of all, our n-gram model are organized in a different manner from the way in which they are usually conceived, so that their usage is also peculiar and needs detailed explanation. N-grams are not collected randomly by recursively creating bigrams and trigrams.

We can define three phases in the processing of our n-gram models: phase 1, building; phase 2,

[3] Emily Bender at a meeting in Uppsala University organized by Joakim Nivre.

[4] Rather than using actual wordforms we could use the rank number associated to each type in the dictionary as would be done in current machine learning approaches. But given the size of the training corpus we did not think it would be necessary: the model for the SardiStance task takes just 5Mb of memory and the one for Absita 10Mb.

choosing; phase 3, evaluating. We will clarify each phase in details below.

Phase 1. Building fully supervised n-gram models

As explained above, we collect topic words from unique dictionary derived from the training set. Topic words are the key entry in the n-gram, in that n-grams are built from each tweet around topic words. There two constraints at the basis of each n-gram: one is content related and the other is quantity related. The quantity constraint requires each n-gram to be longer than 3 words in sequence, in addition to the topic word. The content constraint requires that each n-gram must have at least a topic word at the beginning or end of the sequence of words. That is, each n-gram has a topic word as head or as tail. N-grams are strictly conditioned by the length of the tweet from which they are extracted. Short tweets may have only one n-gram at most or none. Long tweets may have two or more n-grams depending on their content: they would be all contained in the same list headed by the sum KLD index for that tweet. N-grams can be expressed in actual words or in lemmata. In the latter case, words are no longer available to subsequent analysis. We organized models with both words and lemmata. Every n-gram comes with the class attributed to the tweet in which they were contained.

Phase 2. Choice constraints on n-grams

Thus n-grams are each associated to two KLD indices, one for each word, and another one from the lump sum - which is unique - of all the words indices contained in the tweet. In this way, n-grams coming from the same tweet can be easily identified and this information can be used to select sequences of n-grams. Sequences of n-grams when matched with the input tweet are used to reinforce the similarity hypothesis. Choosing n-grams from the model is basically done on the basis of the ratio of intersecting words/lemmata. We established different ratios: one fifth or 20% of intersection, one fourth or 25%, one third or 30% and finally half or 50% intersecting words/lemmata. The ratio may vary according to another important parameter which is tied to the way in which the n-gram is used. We can decide to use words, lemmata, but also to erase grammatical or function words. In case we erase function words in the intersection only content words will be computed, which is a much smaller number and requires a smaller ratio to compare. We tried all three choosing manners.

Phase 3. Evaluating n-gram candidates

Once the methods have been selected and candidate n-grams are extracted from the model according to choice constraints, the outcome may be just one candidate and the evaluation stops or more than one candidate which is the rule. Now we have a list of candidate n-grams with the best ones at the top. The list may be created in a number of different manners. It has the KLD index inherited from the tweet and three other indices: one is the ratio of intersection words/lemmata, the higher this ratio the more relevant is the n-gram. Another index is the sum of the KLD indices associated to each of its word/lemma, the lower this sum the more relevant is the ngram (rare content words have a lower KLD index). Finally the third index is the one associated to the tweet in which the n-grams are contained. Choosing the best candidate in fact usually means selecting the best candidates from the list, because it almost never happens that there is only one candidate at the top with the best ratio or best index. The choice requires collecting candidates at the top with the same ratio/index. However this may require another step since the best candidates may be associated to different classes. So that after the first sieve has reduced the number of best candidates, another sieve requires selecting the most frequent class and this is done by reordering the best candidates on the basis of their class. In fact, this might also be one possible general method: rather that selecting only best candidates, one might reorder all candidates chosen on the basis of the intersection ratio, and count and choose the most frequent class. Eventually, another evaluation modality can be derived from the KLD indices. We compute differences on the basis of the KLD sum index for each model n-gram compared to the input n-gram and use this difference as the relevant index. When candidates are sorted in a list, the top will be populated by the lowest indices which can be used to characterize similarity. We chose the class of the top n-gram, but also tried a best way by selecting the first n-gram carrying a non negative index. Negative sums may still indicate higher differences between two n-grams.

Thus overall we come up with 6 different methods multiplied by two (function words erased/all words/lemmata), which amounts to 12 different methods. We experimented them all but at the end we concentrated only on a few.Since it is reasonable to assume that not all tweets of the

training set will be classified in the model due to the lack of an instance defined by the list of automatically derived keywords in the training corpus, we ascertained at first what was the coverage of the training text for the development set, using in this case the model for the training set.
We report here below both training set coverage of the development set and development set results for both tasks. As can be easily noticed, coverage for the semantic-statistic module is poor, and the same applies for the so-called lexical-semantic module, which is even worse and as said above we only used as default.

	Cover Sardines	Devel Sardines	Cover HaSpDe	Devel HaSpDe
Sem-Stat	57.98%	35.31%	58.34%	38.67%
Stat-Only	93.91%	39%	92.6%	44.8%
Lex-Sem		39.34%		37.8%

Table 4. Coverage and Results for the Development Set for both Tasks

5 Results and Discussion

We present at first results of the SardiStance task where .

Task SardiStance

Run1 (Semantic module)
Macro-F1 0.3881500114277561

Run2 (Statistic module)
Macro-F1 0.3637025029179095

We then performed additional runs with the statistical module always with the test set. However we were unable to know the results until the evaluation script and the Test Gold set were distributed to all participants. We then realized that we had one run with the worst result and another with the best result. The former run was by choosing the first candidate from the list proposed by the KDL indices with a positive value in the list of candidated produced by a difference computed between the index of testset n-grams and the index of trainmodel n-grams. The latter run was instead obtained by choosing the best candidate – the one with higher value in terms of number of shared words from the intersection at word level between testset n-gram and trainmodel n-gram, and got the following results:

Task SardiStance

Run3 (Statistic module – first candidate with positive value)
Macro-F1 0.299607934

Run4 (Statistic module – higher word intersection)
Macro-F1 0.427668958

Even considering the last fourth run our ranking would not change. We assume that basing the evaluation on one n-gram alone is not the best solution. So we modified our evaluation procedure by requiring a sequence of at least two n-grams for each tweet/news to be chosen at the same time, using the same tweet-related KLD to select them. In this case we were able to cover more text and get a better similarity measure that we report in the subsection below.
We present here below the official results obtained at first for the HaSpeeDe2 task A-B both for News and for Tweets, and then the results obtained for SardiStance. Consider that we could report results for two runs only, and we choose the Semantic-Statistic and the Statiscs-Only.

Task HaspeeDe2 - News

Task A	
RUN-1	RUN-2
Macro-F1: 0.5024333	Macro-F1: 0.3805618
Task B	
RUN-1	RUN-2
Macro-F1: 0.5386702	Macro-F1: 0.3671441

Task HaspeeDe2 - Tweets

Task A	
RUN-1	RUN-2
Macro-F1: 0.5054034	Macro-F1: 0.4726022
Task B	
RUN-1	RUN-2
Macro-F1: 0.5078902	Macro-F1: 0.4671661

As for the SardiStance tasks, results for the HaSpeeDe2 task, obtained and delivered in due time are not particularly satisfactory, even though they are in line to results obtained for the development set. As for the SardiStance tasks, in fact there is a remarkable difference from the result obtained for the Development set in the Semantic-statistic.

5.1 The Improvements in the Statistical Module

After receiving the Test Gold version and the evaluation script, we continued producing other runs on the basis of the statistical approach and the choice of the algorithm we had available, for instance restricting choice of candidates only to those in which two or more n-grams had been selected. We discovered that in one these additional runs - the fifth for SardiStance and the sixth for HaSpeeDe2- we improved up to 54% macro-F1 for HaSpeeDe2 task and up to 48% macro-F1 for SardiStance. Here below the results for SardiStance and further the ones for HaSpeeDe2.

Run-5 SardiStance Task

Macro F1	**0.484871151**

Run-6 HaSpeeDe2 Task News

Task A	***Task B***
Macro-F1: 0.53828428	Macro-F1: 0.54071432

Run-6 HaSpeeDe2 Task Tweets

Task A	***Task B***
Macro-F1: 0.52836397	Macro-F1: 0.53965935

6 Conclusion

In this paper we presented the system we used for the two tasks HaSpeeDe2 and SardiStance. We used different approaches one of which was based on previous participation in similar Evalita tasks. Two methods are however innovative in their use of fully supervised n-grams, automatically derived. We use statistical measure to classify n-grams and a variety of different possible solutions which we explain in detail. The high number of possible results are however only evaluated against the development set. We are convinced that participants to these tasks which are mainly directed to the use of commonly available machine-learning software - should be allowed to propose a higher number of runs due to the variability of behaviour of the algorithm when relevant parameters in statistical tools are modified.

References

Basile, Valerio and Croce, Danilo and Di Maro, Maria, and Passaro, Lucia C., 2020. EVALITA 2020: Overview of the 7th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, in Proceedings of Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2020), CEUR.org.

Cignarella, Alessandra Teresa and Lai, Mirko and Bosco, Cristina and Patti, Viviana and Rosso, Paolo, 2020. Overview of the EVALITA 2020 Task on Stance Detection in Italian Tweets (SardiStance), in Proceedings of the 7th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian (EVALITA 2020), CEUR.org.

Delmonte R., 2014. ITGETARUNS A Linguistic Rule-Based System for Pragmatic Text Processing, Proceedings of Fourth International Workshop EVALITA 2014, Pisa, Edizioni PLUS, Pisa University Press, vol. 2, pp. 64-69.

Gries, Stefan Th. 2008. Dispersions and adjusted frequencies in corpora. International Journal of Corpus Linguistics 13/4: 403-437.

Gries, Stefan Th. 2010. Dispersions and adjusted frequencies in corpora: further explorations. In Stefan Th. Gries, Stefanie Wulff, and Mark Davies eds. Corpus linguistic applications: current studies, new directions. Amsterdam: Rodopi, 197-212.

Koos van der Wilt, Linguistics improves statistical classification: the positive effects of reducing feature dimensionality or grammatical feature selection. Downloadable at https://www.academia.edu/27207951/Linguistics_improves_statistical_classification_with_KLD_NB_TF_IDF_K_NN_the_positive_effects_of_reducing_feature_dimensionality_or_grammatical_feature_ selection_ Koos_van_der_Wilt.

Sanguinetti, Manuela and Comandini, Gloria, and Di Nuovo, Elisa and Frenda, Simona and Stranisci, Marco and Bosco, Cristina and Caselli, Tommaso and Patti, Viviana and Russo, Irene, 2020. Overview of the EVALITA 2020 Second Hate Speech Detection Task (HaSpeeDe 2), in Proceedings of the 7th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA 2020), CEUR.org.

Stingo M., & R. Delmonte, 2016. Annotating Satire in Italian Political Commentaries with Appraisal Theory, IN Larry Birnbaum, Octavian Popescu and Carlo Strapparava (eds.), Natural Language Processing meets Journalism - Proceedings of the Workshop, NLPMJ-2016, PP. 74-79.