Venses @ AcCompl-It: **Computing Complexity vs Acceptability with a Constituent Trigram Model and Semantics**

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 work carried out for the Ac-ComplIt task. ItVENSES is 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 statistically based approach and the semantic one used previously. The statistic approach is characterized by the use of trigrams of constituents computed by the system and checked against a trigram model derived from the constituency version of VIT – Venice Italian Treebank. Results measured in term of a correlation, are not particularly high, below 50% the Acceptability task and slightly over 30% the Complexity one.

1 Introduction

In this paper we will present work carried out by the Venses Team in Evalita 2020 (Basile et. 2020). We will describe in detail in the following work carried out on the Ac-ComplIt task. We present the modules for automatic classification that uses two different approaches: a fully BOW and statistic one, a fully semantically based one. The trigram model is built on the basis of the analysis performed by ItVenses at different levels of linguistic complexity.
The procedure we organized for the semantically-based analysis is as follows.

At first we massaged the text in order to obtained 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.
ItGetarun receives as input a string – the sentence(s) to be analysed - which is then tokenized into a list. The list is then sentence split, fully tagged, disambiguated and chunked. Sentence level chunks are then parsed together into a full sentence structure which is passed to the Island-Based predicate-argument structure (hence PAS) parser.
The output of the semantic parser is passed on to the module for classification called ItVenses. ItVenses inherits constituent labels from chunked sentences which have been first destructured, i.e. all embedded structures have been collapsed and linearized in order to construct a sequence of linear constituent labels.
In addition, ItVenses takes into account agreement, negation and non-factuality usually marked by unreal mood, information available at propositional level, used to modify previously assigned polarity from negative to positive, on the basis of PAS and their semantics. For this reason, we keep trace of hate and stereo words on a lexical basis, together with presence of negation. In particular, hate and stereo words and sentiment polarities (negative and positive), are checked together one by one, in order to verify whether polarity has to be attenuated, shifted or inverted (see Polanyi & Zaenen, 2006) as a result of the presence of intensifiers, maximizers, minimizers, diminishers, or simply negations at a higher than constituent level (see Ohana et al. 2016). All this information comes from the Deep

Island Parser (hence DIP) described in the section below.

2 The Deep Island Parser

Conceptually speaking, the deep island parser (hence DIP) is very simple to define, but hard to implement. A semantic island is made up by a set of A/As which are dependent on a verb complex (hence VCX). Arguments and Adjuncts may occur in any order and in any position: before or after the verb complex, or be simply empty or null. Their existence is determined by constituents surrounding the VCX. The VCX itself can be composed of all main and minor constituents occuring with the verb and contributing to characterize its semantics. We are here referring to: proclitcs, negation and other adverbials, modals, reconstruction verbs (lasciare/let, fare/make, etc.), and all auxiliaries. Tensed morphology can then appear on the main lexical verb or on the auxiliaries/modals/reconstruction verbs.
The DIP is preceded by an augmented context-free parser that works on top of a tagger and a chunker. Chunks are labeled with usual grammatical relations on the basis of syntactic subcategorization contained in our verb lexicon of Italian counting some 17,000 entries. There are some 270 different syntactic classes which differentiates also the most common preposition associated to oblique arguments. Position in the input string is assumed at first as a valid criterion for distinguishing SUBJects fro, OBJects. The semantic parser will then be responsible for a relabeling of the output.
The DIP receives a list of Referring Expressions and a list of VCX. Referring expressions are all nominal heads accompanied by semantic class information collected in a previous recursive run through the list of the now lemmatized and morphologically analyzed input sentence. It also receives the output of the context-free parser. The DIP searches for SUBJects at first and assumes it is positioned before the verb and close to it. In case there is none such chunk available the search is widened if intermediate chunks are detected: they can be Prepositional Phrases, Adverbials or simply Parentheticals. If this search fails, the DIP looks for OBJects close after the verb then and again possibly separated by some intermediate chunk. They will be relabeled as Subjects. Conditions on the A/As boundaries are formulated in these terms:
- between current VCX and prospective argument there cannot be any other VCX

Additional constraints regard presence of relative or complement clauses which are detected from the output chunked structure.
The prospective argument is deleted from the list of Referring Expressions and the same happens with the VCX. The same applies for the OBJect, OBJect1 and OBLique. When arguments are completed, the parser searches recursively for ADJuncts which are PPs, using the same boundary constraint formulation above.
Special provisions are given to copulative constructions which can often be reversed in Italian: the predicate coming first and then the subject NP. The choice is governed by looking at referring attributes, which include definiteness, quantification, distinction between proper/common noun. It assigns the most referring nominal to the SUBJect and the less referring nominal to the predicate. In this phase, whenever a SUBJect is not found from available referring expressions, it is created as little_pro and moprhological features are added from the ones belonging to the verb complex. The Predicate-Argument Structure (hence PAS) thus obtained, is then enriched by a second part of the algorithm which adds empty or null elements to untensed clauses.

3 The Classification Procedure

The classification and evaluation procedure is carried out on constituents and their corresponding semantics at propositional level in two steps.
The procedure is preceded by the creation of the model which is made up of the following three components:
- a dictionary of token trigrams, one for every occurrence in a sentence with associated frequency value and sentence id. We will use the following sentence no. AC-01-R0364 as example for the classification.

<sent>'AC-01-
R0364'<lik_scl>'1.666666667'</lik_scl><st_err>
'0.284267622'</st_err><text>Quando il dipartimento concedeva dei fondi lui spendevano tutti i soldi in trasferte.</text></sent>

The list below represents the sequence of constituents extracted from sentence reported above, with the final punctuation mark added.

The triple below is the first one extracted from the previous list.

tktr(1-[f,fs,f,sn,ibar,sq,sn,ibar,sn,sp,punto]-'AC-01-R0364_1').[2]
tktr(1- (f-fs-sn)-'AC-01-R0364_1').[3]

- a list of sentence constituent types corresponding to the training corpus made of an index, a list of trigrams with their local frequency of occurrence, an evaluation and classification value as derived from the training set: this is the list for the same sentence.

scst('AC-01-R0364'-[1-[f,fs,sn,ibar,sq,sn,ibar,sn,sp,punto],1- (f-fs-sn),1-(fs-sn-ibar),1- (sn-ibar-sq),1- (ibar-sq-sn),1- (sq-sn-ibar),1- (sn-ibar-sn),1- (ibar-sn-sp),1- (sn-sp-punto)]-['1.666666667','0.284267622']).

- a dictionary of type constituent trigrams or unique forms with frequency of occurrence in the whole corpus. For instance the following triple occurs 5 times in the training corpus:
tptr(5- (vcomp-savv-ibar)).

- a list of semantic parameters associated to each sentence, where since semantics is computed at propositional level, the list is constituted by a set of parameters preceded by a lemmatized predicate. Parameters considered are the following ones: agreement (may take on three values: false, true, null); negation (propositions – first slot - but also predicates may be lexically negatively marked! – second slot); speech act (8 different types); factivity (two values).

semp('AC-01-R0364'-[true-concedere-statement-factive-[pos,nil],false-spendere-statement-factive-[pos,neg]]-['1.666666667','0.284267622']).

Overall we collected from the training corpus 12309 token trigrams, 739 type trigrams, 2678 semantic feature sets. We then created the development corpus, by extracting 20% of sentences from the training corpus, which adds up to 414 sentences for the Complexity corpus and 252 sentences for the Acceptability corpus. The corresponding Development models were created by analysing the remaining sentences. We were then able to match the content of two models each for the two tasks: the new model of the reduced Training corpus that we obtained by extracting 20% of sentences which we matched against the corpus of the extracted sentences or DevSet. In order to evaluate the output we decided to consider as correct approximation a value whose difference from the target value was lower than 1. It is important to notice that results are to be referred to sentence level after splitting: this adds 3 more sentences to the Complexity DevSet which turns the total amount from 413 to 416. On the contrary, in the Acceptability DevSet the system didn't split any sentence. Here is the list of additional sentences processed: CO-01-R0317_2, CO-01-R0357_2, CO-01-R0637_2: they are caused by presence of dots which are interpreted by the parser as a possible sentence split.
We report here below Precision and Recall for the DevSet that we evaluated at first against the Training Corpus Model for coverage issues and then against the DevSet Corpus model. Results we obtained are as follows:

Coverage of the DevSet by the Training Corpus Model
- Acceptability
Total sentences processed 249 over 252 corresponding to 98.8%
207 over 249 Likert Scale (83.13%)
203 over 249 Standard Error (81.52%)
- Complexity
Total sentences processed 412 over 416 corresponding to 99.03%
398 over 416 Likert Scale (95.67%)
399 over 416 Standard Error (95.81%)

Results of the DevSet by the Development Corpus Model
- Acceptability
Total sentences processed 250 over 252 corresponding to 99.2%
151 over 252 Likert Scale (59.92%)
140 over 252 Standard Error (55.55%)
- Complexity
Total sentences processed 412 over 416 corresponding to 99.03%
263 over 416 Likert Scale (63.62%)
255 over 416 Standard Error (61.29%)

First step in the classification and evaluation procedure is the constituent trigram matching step. In this step trigrams are computed for the

[2] In more detail the sequence of constituents is as follows: [f-[fs-[fs-[Quando],f-[sn-[il dipartimento],ibar-[concedeva], sq-[dei fondi]]], sn-[lui],ibar-[spendevano],sn-[tutti i soldi],sp-[in trasferte]]]. As can be noted, we eliminate functional constituents like "fs" and "f" and keep only those containing a semantic head. We also keep the initial symbol.

[3] We use Italian constituent labels where F stands for S, SN for NP etc. and Phrase is turned into Sintagma.

input text and are matched against the token trigrams dictionary. The matching should produce a list of possible sentence types: we choose the sentence which has more than half of the trigrams matched. The sentence type trigram list is then used to check trigram sequences: here again more than half of the trigrams should be related in sequence. In case this process succeeds we take the associated classification and the evaluation stops. If the process fails, we search the trigram database derived from VIT, which is made of 273,000 (Delmonte et al., 2007) trigrams organized into four frequency related subclasses: rare trigrams with frequency of occurrence including all hapax, dis, trislegomena; frequent trigrams with frequency of occurrence from 4 to 20; very frequent trigrams with frequency of occurrence higher than 20. According to their placement, trigrams are regarded more or less easy to accept vs complex in case their frequency is rare.

VIT (Venice Italian Treebank) is a treebank consisting of 320.000 words created by the Laboratory of Computational Linguistics of the Department of Language Sciences of the University of Venice. The VIT Corpus consists of 57.000 words of spoken text and of 273.000 words of written text. Syntactic annotation was accomplished through a sequence of semi-automatic operations followed by manual validation. The first version of the Treebank was created in the years 1985-88 – manually parsing 40000 words of text with a constituent structure only representation. The resulting structure labels were collected and were used to build a context-free parser for a speech synthesizer (Delmonte R. and R. Dolci, 1991). The theoretical framework behind our syntactic representation was X-bar theory. One peculiarity of VIT is the intention to make it representative of the Italian linguistic syntactic and semantic variety: we thus introduced texts from five different genres – news, bureacratic genre, political genre, scientific genre, literary genre. This made the resulting structures a treebank with a high coverage but very sparse.

4 The Evaluation Module

We assigned rewards and penalties according to a scheme which was partly based on constituency and partly on semantics. In particular, we used agreement, negation, factivity from semantic processing and complex constituency structures from trigram model and a smal set of heuristically determined rules. To check agreement we took the main verb predicate and its morphology and matched this information with the one available on the lexically expressed subject. Here below some examples of semantic information used for agreement matching:

<sent>'AC-01-R0364'<lik_scl>'1.666666667'</lik_scl><st_err>'0.284267622'</st_err><text>
Quando il dipartimento concedeva dei fondi lui spendevano tutti i soldi in trasferte.</text></sent>

Sem = [concedere-statement-factive-[pos, nil], spendere-statement-factive-[pos, neg]]
Agrs = [false]
Negs = [neg]

In addition, we used lexical representations in order to verify the level of matching existing between two predicates. In particular we checked syntactic classes and conceptual classes[4] (Delmonte R., 1989; 1990; 1995).

Here are some verb lexical representation in our lexicon, where we list the root, the conjugation, the syntactic class, the aspectual class, the conceptual class, the list of arguments and their inherent semantic features preceded by constituent type and semantic role. Here below the example of “stonare”/clash

pv(ston,1,inerg,statv,exten,[np/subj1/theme_unaff/[-ani,+hum]]).

where “ston” = is the root, “1” = the conjugation (first implies the morpheme “are” to be adjoined), “intr” = the syntactic type, intransitive or unergative, “statv” = stative, the aspectual class, “exten” = extensional, the conceptual class. The list of possible arguments follows starting from

[4] Syntactic lexical classes include the following:
tr=transitive; tr_cop=transitive+predicative argument;
tr_perc=transitive_perceptive; ditr(+preps)=ditransitive;
psych1=psychic 1; psych2=psychic 2; psych3=psychic 3;
inac=unaccusative; inerg=unergative; rifl=reflexive;
rifl_rec=reflexive reciprocal; rifl_in=reflexive inherent;
erg_rifl=ergative reflexive; imp=impersonal;
imp_atm=impersonal atmospheric; cop=copulative;
mod=modal; C_mov=movement verb + another class;
C_prop=propositional verb + another class;
Conceptual lexical classes include the following:
ask_poss,at_posit,coerc,dir,dir_difclt,dir_tow,divid,eval,exten,exten_neg,factv, go_against,hold,hyper, inform, ingest, into_hole,let,manip,measu_maj,measu_min,ment_act, not_exten,not_let,not_react,over,percpt, perf,posit, possess,process,propr,react,rep_contr,subj,touch,unit

the "subj1" = subject, which is a "np" Noun-Phrase, and has "theme_unaff" = theme unaffected as semantic role. Semantic features are "-ani" = minus animate, "+hum" = plus human, i.e. only humans and not animate being are selected. In case a verb selects more argument types, the entry is repeated each one containing a different structural construction. This applies for instance to "scoppi"/burst,explode,break out.

pv(scoppi,1,inac,statv,exten,[np/subj1/theme_unaff/[-ani,+hum]]).
pv(scoppi,1,inac,statv,exten,[np/subj1/theme_unaff/[+hum],pp/obl/theme/di/[+abst]]).
pv(scoppi,1,inac,statv,exten,[np/subj1/theme_unaff/[+hum],vinf/vcomp/prop/a/[subj=subj1]]).

In the third entry, we have a quasi-idiomatic form "scoppiare a piangere"/burst into tears, where the infinitival has a subject bound to the higher governing verb's subject. This is done according to principles expressed in LFG theory (Bresnan, 1982; 2001).
Lack of agreement in lexical classes reduces the score associated to the similarity match between the two trigrams under evaluation for the current sentence. Other scoring functions are associated to speech act, grammatical agreement, presence/absence of negation at propositional/lexical level; factivity; complex constituency. Overall we have eight possible features.

Speech Act
Lexical classes:
syntactic
conceptual
Negation:
lexical
propositional
Agreement
Factivity
Complexity at constituent level

Table 1. Linguistic features used by ItVenses

Thus schematically we have:
Rewards:
0 no wrong agreements; 0 no negation; 0 no non-factive; same conceptual lexical features; similar syntactic lexical features; 0 no complex constituency structures
Else:
penalties (reducing acceptability vs increasing complexity)

Similarity in syntactic lexical classes tends to reduce the more detailed lexical classification into one single label, as for instance the label "transitive" will include: tr (transitive), tr_cop (transitive+predicative argument), tr_perc (transitive_perceptive), ditr(+preps) (ditransitive).
As to constituency complexity we count all constituent labels that are indicators of: sentential complement represented by FAC (Italian for SCOMP); subordinator for subordinate clause, CP; complementizer or interrogative pronoun represented by CP; relative clause, F2; coordinate clause, FC. According to the quantity of one or more of these constituent labels, we assign penalties or rewards. The decision is determined by heuristics but also by the length in number of constituents. For instance, 2 CP + 1 FAC will be computes as a penalty; 1 CP, 1 FAC, 1 F2 again penalty, however length in terms of constituents should be higher than 8. We also address specific constituent sequences which indicate complex or hard to understand structures as for instance the sequence:

[…, fc,sn,vcomp,sn,punto]

which classifies some 20 sentences in the Acceptability test set, one of which is sentence n. AC-OC-02-R0569:

"Ci dissero chi Maria aveva chiamato un uomo e Marco visitato l'anziano signore."

This sentence is ungrammatical due to presence of a lexical Object NP in the extraction place of the interrogative pronoun "chi". However this case of ungrammaticality is hard to detect solely on the base of constituent sequences because the NP containing "chi" is not lexically marked. On the contrary, the final participial clause is easily detectable.
The evaluation algorithm starts by searching trigrams collected in the current sentence analysis and by trying to match them with the ones memorized in the training set model. The search is successful if one or more matches have been obtained which have 3 or more trigrams. The following step is then collecting features as indicated in Table 1. from the syntactic and semantic output of the parser. These features are matched against the ones that are associated to each trigram sequence collected in the previous step. The matching algorithm receives a vector made of six slots:

match(Strct,Pred,Agrs,Negs,Fact,Spacs)

where, “Strct” stands for constituent structure; “Pred”, is the verbal predicate lemma; “Agrs”, is a binary value (true/false) for subject-verb agreement; “Negs” is a pair of binary values (neg/nil) for negation at lexical and propositional level; “Fact” is again a binary value (true/false) for factivity at propositional level; “Spacs” is one of the seven possible labels[5] used to classify speech act. For instance, in the case of sentence no. 'AC-01-R0364' above, the following counts are generated automatically:

Fact = ['AC-01-R0440_1'-factive, 'AC-01-R0440_1'-factive]
Spacs = [statement, statement]
N = N1 = Va = 0 [negation1, negation2]
N2 = N3 = 2 [agreement] *penalty
Sum = Val = 4 [final score] *penalty

5 Results and Discussion

As said above, results are not successful. In particular, results for the Complexity Task are well below the Baseline. Results for the Acceptability Task are higher and in one case they almost double the Baseline.

COMPL Task
RUN 1
Mean-Correlation: 0.312796825885, p value < 0.001
STD ERR-Correlation: 0.096751776, p value < 0.05
RUN 2
Mean-Correlation: 0.305504444563, p value < 0.001
STD ERR-Correlation: 0.0729839133, p value > 0.05

ACCEPT Task
RUN 1
Mean-Correlation: 0.441645891, p value < 0.001
STD ERR Correlation: 0.248478821, p value < 0.001
RUN 2
Mean-Correlation: 0.494713038815, p value < 0.001
STD ERR-Correlation: 0.405850132, p value < 0.001

As can be easily gathered, differences between Run-1 and Run-2 are not particularly high in the Complexity Task. Not so in the Acceptability task where Run-2 exceeds Run-1 by 0.053 points. Run-2 in both tasks is characterized by a different strategy determined by a policy of feature ablation. What we did, was trying to verify whether the presence of each of the eight features had an important impact on the final result and to what extent. Eventually, we found out that the use of lexical negation was not so relevant and so we deleted it from the final count. And that was the decision that determine the result for Run-2. The different behaviour of the system in the two tasks can be due to the length of the sentences which in the Complexity task is much longer. The system produces results for each proposition and not for the sentence as a whole – we don’t count relative and complement clauses as separate propositions. When generating the final document for the two runs we did not have a strategy in deciding in many cases, which proposition we had to choose as a representative of the whole sentence. We decided we could not make an average between the two or three propositions so we simply selected always the result obtained by the first proposition. This choice applied to 51 sentences, 41 with two propositions and 10 with three propositions. The Complexity text also suffered from failure of the parser in three sentences. We also have to consider the presence of 62 results determined heuristically, i.e. the system did not find the corresponding trigrams in the training set, so it used the VIT database and generated the final statistics by a set of heuristics. No such problems arose in the Acceptability Task, where all sentences where constituted by a single proposition. However, we had a higher number of heuristically determined statistics, 86. If we had the possibility to present more runs, then we could have achieved better results in the Complexity task.

6 Conclusion

We presented the results of our system for the two tasks Complexity and Acceptability. The system uses constituency-based trigrams associated to the semantics of each proposition. Evaluation is based on presence/absence of agreement/match between linguistic features, determined at a lexical, syntactic and semantic level. Worst results obtained for the Complexity Task may be due partly to the length of the sentences, which required a specific strategy in choosing the most relevant classification at propositional level. We concentrated our work on the use of constituent trigrams and did not consider the possibility to use ngrams based on words or lemmata which we had available from our deep analysis. In the future, we intend to use the same approach we produced for the other tasks of EVALITA which are all based on automatically

[5] We use the following: statement, question, exclamation, negated, unreal, opinionsubjective, conditional

generated fully supervised ngram models together with the one presented here.

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.

Bresnan Joan (ed.), 1982. The Mental Representation of Grammatical Relations, The MIT Press, Cambridge MA.

Bresnan Joan, 2001. Lexical function syntax. Oxford, Blackwell Publishers.

Delmonte, R., A. Bristot, S. Tonelli, 2007. VIT - Venice Italian Treebank: Syntactic and Quantitative Features, in K.De Smedt, Jan Hajic, Sandra Kübler(Eds.), Proc. Sixth International Workshop on TLT, Nealt Proc. Series Vol.1, 43-54.

Delmonte R., 1995. Lexical Representations: Syntax-Semantics interface and World Knowledge, in Notiziario AIIA (Associazione Italiana di Intelligenza Artificiale), Roma, pp.11-16.

Delmonte R., 1990. Semantic Parsing with an LFG-based Lexicon and Conceptual Representations, Computers & the Humanities, 5-6, 461-488.

Delmonte R., R.Dolci, 1991. Computing Linguistic Knowledge for a Text-To-Speech System With PROSO, in Proc. EUROSPEECH '91 – Second European Conference on Speech Communication and Technology, Genova, ISCA, Archive, pp. 1291-1294.downloadable at https://www.isca-speech.org/archive/ eurospeech_1991/e91_1291.html

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.

Ohana, B. and B. Tierney and S.J. Delany, 2016, Sentiment Classification Using Negation as a Proxy for Negative Sentiment, in Proceedings of 29th FLAIRS Conference, AAAI, 316-321.

Polanyi, Livia and Zaenen, Annie 2006. "Contextual valence shifters". In Janyce Wiebe, editor, Computing Attitude and Affect in Text: Theory and Applications. Springer, Dordrecht, 1–10.

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, 74-79.