ASAPPpy: a Python Framework for Portuguese STS*

José Santos[1,2] [0000−0001−9207−9761], Ana Alves[1,3] [0000−0002−3692−338X], and Hugo Gonçalo Oliveira[1,2] [0000−0002−5779−8645]

[1] CISUC, University of Coimbra, Portugal
[2] DEI, University of Coimbra, Portugal
[3] ISEC, Polytechnic Institute of Coimbra, Portugal
santos@student.dei.uc.pt,ana@dei.uc.pt,hroliv@dei.uc.pt

Abstract. This paper describes ASAPPpy – a framework fully-developed in Python for computing Semantic Textual Similarity (STS) between Portuguese texts – and its participation in the ASSIN 2 shared task on this topic. ASAPPpy follows other versions of ASAPP. It uses a regression method for learning a STS function from annotated sentence pairs, considering a variety of lexical, syntactic, semantic and distributional features. Yet, unlike what was done in the past, ASAPPpy is a standalone framework with no need to use other projects in the feature extraction or learning phase. It may thus be extended and reused by the team. Despite being outperformed by deep learning approaches in ASSIN 2, ASAPPpy can explain the model learned by the relevant features that have been selected as well as inspect which type of features plays a key role in the STS learning.

Keywords: Semantic Textual Similarity · Natural Language Processing · Semantic Relations · Word Embeddings · Supervised Machine Learning.

1 Introduction

Semantic Textual Similarity (STS) aims at computing the proximity of meaning of two fragments of text. Shared tasks on this topic have been organised in the scope of SemEval 2012 [2] to 2017 [10], targeting English, Arabic and Spanish. In 2016, the ASSIN shared task [14] focused on STS for Portuguese, and its collection was made available. ASSIN 2 was the second edition of this task, with minor differences on the STS annotation guidelines and covering more simple text.

ASAP(P) is the name of a collection of systems developed in CISUC for computing STS based on a regression method and a set of lexical, syntactic, semantic and distributional features extracted from text. It has participated in several STS evaluations, for English and Portuguese, but was only recently integrated in two single independent frameworks: ASAPPpy, in Python, and ASAPPj, in Java. Both of the previous versions of ASAPP participated in ASSIN 2,

* This work was funded by FCT's INCoDe 2030 initiative, in the scope of the demonstration project AIA, "Apoio Inteligente a empreendedores (chatbots)"

but this paper is focused on the former, ASAPPpy. Also, although both ASSIN and ASSIN 2 cover STS and Textual Entailment (TE), this paper is mainly focused on the approach followed for STS, including feature engineering, feature selection and learning methods. The performance of ASAPPpy in STS was satisfactory for an approach that follows traditional supervised machine learning, also enabling an analysis of the most relevant features, but it was clearly outperformed by approaches based on deep learning or its products, including recent transformer-based language models, like BERT [11].

In the remainder of the paper, we overview previous work that led to the development of ASAPPpy, focused on previous versions of this system. We then describe the features exploited by ASAPPpy and report on the selection of the regression method and features used, also covering the official results in ASSIN 2, which we briefly discuss.

2 An overview of ASAP(P) for STS

The first version of ASAP [3] dates from 2014, with the participation in the SemEval task *Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment* [18], in English, though only on the subtask of semantic relatedness. Here, a set of 65 features was extracted from sentence pairs ranging from overlapping token counts, phrase chunks, or topic distributions.

From the first participation, we proposed to learn a model based on regression analysis that considered different textual features, covering distinct aspects of natural language processing. Lexical, syntactic, semantic and distributional features were thus extracted from sentence pairs. The main difference in successive versions is the increasing adoption of more and more distributional features, initially based on topic modeling, and recently on different word embedding models. Its main contribution was in the use of complementary features for learning a STS function, a part of the challenge of building Compositional Distributional Semantic Models.

One year later, ASAP-II [5] participated in a task that was closer to our current goal: *Semantic Textual Similarity* (STS) at SemEval 2015 [1]. Even though the task covered three languages – English, Spanish and Arabic – , we only targeted English. At first, the goal of STS may look similar to the one of SemEval 2014's task, but the available datasets were very different from each other. One such difference was the occurrence of named entities in the SemEval 2015 dataset. To address this, ASAP-II retrieved named entities and compound nouns from DBPedia [7], an effort to extract structured information from Wikipedia. Due to DBPedia's central role in the Linked Data initiative, it is also connected to WordNet [13], which enables the connection between some DBPedia entities and their abstract category.

Finally, one year later, motivated by the organisation of the first ASSIN shared task [14], ASAP focused on Portuguese, becoming ASAPP – Automatic Semantic Alignment for Phrases applied to Portuguese [4]. The first ASAPP ex-

ploited several heuristics over Portuguese semantic networks [16] for extracting semantic features, beyond lexical and syntactic. As the same nature of its predecessors, several tools have been used for the extraction of morpho-syntactic features, including tokenization, part-of-speech tagging, lemmatization, phrase chunking, and named entity recognition. For the first ASSIN, this was achieved with NLPPort [21], built on top of the OpenNLP framework, though with some modifications targeting Portuguese processing.

The original participation in ASSIN did not exploit distributional features. Only later, word embeddings (word2vec CBOW [17]) and character n-grams were adopted by ASAPP (version 2.0) [6]. When trained in the ASSIN training collections, adding distributional features to the others led to improvements in the performance of STS. We also concluded that, although the ASSIN collections were divided between European and Brazilian Portuguese, better results were achieved when a single model was trained in both.

Up until this point, all versions of ASAP(P) could not be seen as a single well-integrated solution. Different features were extracted with different tools, not always applying the same pre-processing or even using the same programming languages, and sometimes by different people. After extraction, all features were integrated in a single file, then used in the learning process. Towards better cohesion and easier usability, in 2018, we started to work on the integration of all feature extraction procedures in a single framework. Yet, due to specific circumstances, we ended up developing two versions of ASAPP: ASAPPpy, fully in Python, and ASAPPj, fully in Java. Each was developed by a different person, respectively José Santos and Eduardo Pais, both supervised by Ana Alves. This paper is focused on ASAPPpy.

Besides training and testing both versions of ASAPP in the collection of the first ASSIN, their development coincided with ASSIN 2, where they both participated. Curiously, the data of ASSIN 2 is closer to that of SemEval 2014's task [18], where the first ASAP participated.

3 Feature Engineering for Portuguese STS

The main difference between ASAPPpy and previous versions of ASAPP is that it is fully implemented in Python. This includes all pre-processing, feature extraction, learning, optimization and testing steps.

ASAPPpy follows a supervised learning approach. Towards the participation in ASSIN 2, different models were trained in the training collection of ASSIN 2, and some also in the collections of the first ASSIN (hereafter ASSIN 1). Both collections have the same XML-like format, where a similarity score (between 1 and 5) and an entailment label are assigned to each pair of sentences, based on the opinion of several human judges. The first sentence of the pair is identified by t and the second by h, which stands for *text* and *hypothesis*, respectively.

The ASSIN 1 collection comprises 10,000 pairs, divided in two training datasets, each with 3,000 pairs, and two testing datasets, each with 2,000, covering the European-Portuguese (PTPT) and Brazilian-Portuguese (PTBR) vari-

ants. The ASSIN 2 collection is divided into training and validation datasets, with 6,500 and 500 pairs, respectively, and a testing dataset, with 3,000 pairs whose similarity our model was developed to predict. In contrast to ASSIN 1, the ASSIN 2 collection only covers the Brazilian-Portuguese (PTBR) variant.

To compute the semantic similarity between the ASSIN sentence pairs, a broad range of features was initially extracted, including lexical, syntactic, semantic and distributional. All features were obtained using standard Python as well as a set of external libraries, namely: NLTK [8], for getting the token and character n-grams; NLPyPort[4], a recent Python port of the NLPPort toolkit [21] based on NLTK, for Part-of-Speech (PoS) tagging, Named Entity Recognition (NER) and lemmatisation; Gensim [20], for removing non-alphanumeric characters and multiple white spaces, and, in combination with scikit-learn [19], to extract the distributional features. Semantic features were based on a set of Portuguese relational triples (see section 3.3) and distributional features relied on a set of pre-trained Portuguese word embedding models (see section 3.4).

Table 1 summarises all the features extracted, to be described in more detail in the remainder of this section.

Features	**Count**
Common token 1/2/3-grams (Dice, Jaccard, Overlap coefficients)	9
Common character 2/3/4-grams (Dice, Jaccard, Overlap coefficients)	9
Difference between number of each PoS-tag (25 distinct tags)	25
Semantic relations between tokens in both sentences (4 types)	4
Difference between number of NEs of each category (10 categories)	10
Difference between number of NEs	1
TF-IDF vectors cosine	1
Average word-embeddings cosine (5 models)	5
Average TF-IDF weighted word-embeddings cosine (5 models)	5
Token n-grams binary vectors cosine	1
Character n-grams binary vectors cosine	1
Total	71

Table 1. Features extracted to train the STS models.

3.1 Lexical Features

Lexical features compute the similarity between the sets and sequences of tokens and characters used in both sentences of the pair. This is achieved with the Jaccard, Overlap and Dice coefficients, each computed between the sets of token n-grams, with $n = 1$, $n = 2$ and $n = 3$, and character n-grams, with $n = 2$, $n = 3$ and $n = 4$, individually. In total, 18 lexical features were extracted given that, for each n-gram, both token and character, we computed the three different coefficients. Figure 1 illustrates how sentences were split into n-grams, in this

[4] `https://github.com/jdportugal/NLPyPort`

particular case, character 2-grams, and provides the value of the coefficients computed over them, used as features.

t: Uma pessoa tem cabelo loiro e esvoaçante e está tocando violão

↓

Character n-grams of size 2 in t: {Um, ma, pe, es, ss, so, oa, te, em, ca, ab, be, el, lo, lo, oi, ir, ro, es, sv, vo, oa, aç, ça, an, nt, te, es, st, tá, to, oc, ca, an, nd, do, vi, io, ol, lã, ão}

h: Um guitarrista tem cabelo loiro e esvoaçante

↓

Character n-grams of size 2 in h: {Um, gu, ui, it, ta, ar, rr, ri, is, st, ta, te, em, ca, ab, be, el, lo, lo, oi, ir, ro, es, sv, vo, oa, aç, ça, an, nt, te}

$$\textbf{Jaccard}(T,H) = \frac{|T \cap H|}{|T \cup H|} = \frac{20}{42} = 0.4762 \quad (1)$$

$$\textbf{Overlap}(T,H) = \frac{|T \cap H|}{|\min(T,H)|} = \frac{20}{28} = 0.7143 \quad (2)$$

$$\textbf{Dice}(T,H) = \frac{|T \cap H|}{|T| + |H|} = \frac{20}{62} = 0.3226 \quad (3)$$

Fig. 1. Example of computing the 2-grams overlap.

Two variants of the previous lexical features were considered only for ASSIN 2. Their value is the cosine similarity between binary vectors obtained from each sentence as follows: (i) extract the list of n-grams in sentence t, h or both, considering different values of n; (ii) represent each sentence as a vector where each dimension corresponds to one of the extracted n-grams and is 1, if the n-gram is in both t and h, or 0, if it is in only one. This was made for token 1/2/3-grams (first feature) and character 2/3/4-grams (second feature). Figure 2 illustrates the computation of this alternative character n-gram based feature to the sentences used in the previous examples.

3.2 Syntactic Features

The only syntactic features exploited were based on the PoS tags assigned to the tokens in each sentence of the pair, namely the absolute difference between the number of occurrences of each PoS tag (25 distinct) in sentence t with those in sentence h. Considering the sentences used in the previous example, figure 3 shows the PoS tags for each word and the array of features obtained after applying the aforementioned method. In these two sentences, only five distinct tags were identified, which meant that for the remaining 20 the feature has value zero.

t: Uma pessoa tem cabelo loiro e esvoaçante e está tocando violão

↓

Character 2/3/4-grams in t: {um, ma, pe, es, ss, so, oa, te, em, ca, ab, be, el, lo, lo, oi, ir, ro, es, sv, vo, oa, aç, ça, an, nt, te, es, st, tá, to, oc, ca, an, nd, do, vi, io, ol, lã, ão, uma, pes, ess, sso, soa, tem, cab, abe, bel, elo, loi, oir, iro, esv, svo, voa, oaç, aça, çan, ant, nte, est, stá, toc, oca, can, and, ndo, vio, iol, olã, lão, pess, esso, ssoa, cabe, abel, belo, loir, oiro, esvo, svoa, voaç, oaça, açan, çant, ante, está, toca, ocan, cand, ando, viol, iolã, olão}

h: Um guitarrista tem cabelo loiro e esvoaçante

↓

Character 2/3/4-grams in h: {um, gu, ui, it, ta, ar, rr, ri, is, st, ta, te, em, ca, ab, be, el, lo, lo, oi, ir, ro, es, sv, vo, oa, aç, ça, an, nt, te, gui, uit, ita, tar, arr, rri, ris, ist, sta, tem, cab, abe, bel, elo, loi, oir, iro, esv, svo, voa, oaç, aça, çan, ant, nte, guit, uita, itar, tarr, arri, rris, rist, ista, cabe, abel, belo, loir, oiro, esvo, svoa, voaç, oaça, açan, çant, ante}

Binary vector $\overrightarrow{t}$: [0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0]

Binary vector $\overrightarrow{h}$: [1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1]

$Cosine(\overrightarrow{t}, \overrightarrow{h}) = 0.5955$

Fig. 2. Alternative lexical features.

t: Uma pessoa tem cabelo loiro e esvoaçante e está tocando violão

↓

PoS-tags in t: {art n v-fin n N conj-c N conj-c v-fin N n}

h: Um guitarrista tem cabelo loiro e esvoaçante

↓

PoS-tags in h: {art n v-fin n N conj-c N}

N	art	conj-c	n	v-fin
1	0	1	1	1

Fig. 3. Syntactic features based on PoS-tagging.

3.3 Semantic Features

Language is flexible in a way that the same idea can be expressed through different words, generally related by well-known semantic relations, such as synonymy or hypernymy. Such relations are implicitly mentioned in dictionaries and explicitly encoded in wordnets and other lexical knowledge bases (LKBs). In order to extract the semantic relations between words in each pair of sentences, a set of triples, in the form $word_1$ SEMANTIC-RELATION $word_2$, was used. They were acquired from ten lexical knowledge bases (LKBs) for Portuguese [16] and, for this work, only those that occurred in at least three LKBs were considered. Based on this, four features of this kind were computed, by counting the number of semantic relations that existed between words in sentence t and words in sentence h and then normalising the result. The following semantic relations were considered: (i) synonymy; (ii) hypernymy/hyponymy; (iii) antonymy; (iv) any other relation covered by the set of triples. Before searching for relations, words were lemmatized with NLPyPort. Considering the sentences used in the two previous examples, table 2 shows the array of features obtained with the aforementioned method. In this case, there was a single relation, *pessoa* (person) hypernym-of *guitarrista* (guitar player).

antonyms	synonyms	hyperonyms	other
0.0	0.0	0.0625	0.0

Table 2. Semantic features based on semantic relations.

Besides semantic relations, Named Entities (NE) were also exploited, due to their importance for understanding the meaning of text. Although the collection of ASSIN 2 would not include NEs, these features were still exploited, considering the application of the model to other tasks. Computed features included the absolute difference between the number of entities of each type identified in sentence t and those in sentence h. As ten different NE types were recognized (i.e, Abstraction, Event, Thing, Place, Work, Organization, Person, Time, Value, Other), this resulted in ten features, plus one for the absolute difference of the total number of NEs between the sentences.

3.4 Distributional Features

Distributional features were based on the TF-IDF matrix of the corpus, which allowed the representation of each sentence as a vector. The first feature of this kind was the cosine of the TF-IDF vector of each sentence.

In addition to the TF-IDF matrix, and given the importance of distributional similarity models for computing semantic relatedness, four pre-trained word embeddings for Portuguese, based on different models and data, were also exploited, namely: (i) NILC embeddings [17], which offer a wide variety of pre-trained embeddings, learned with different models in a large Portuguese corpus.

From those, CBOW Word2vec and GloVe, both with 300-dimensioned vectors, were selected; (ii) fastText.cc embeddings [9], which provide word vectors for 157 languages, trained on Common Crawl and Wikipedia using fastText. For the present system, only the Portuguese word vectors were used; (iii) ConceptNet Numberbatch [22], obtained by applying a generalisation of the retrofitting technique [12], which improves the representation of words in the form of vectors by utilising the ConceptNet knowledge base. Given that the pre-trained vectors used are multilingual, only the vectors of Portuguese words were used; (iv) PT-LKB embeddings [15], a different distributional model, not learned from corpora, but built by applying the node2vec method to the same ten LKBs used for the semantic features [16]. The vectors used had 64 dimensions, the value that achieved best results in word similarity tests [15].

For each model, two different features were considered, both after the conversion of each sentence into a vector computed from the vectors of its tokens. The difference is in how this sentence vector was created. For the first feature, it was obtained from the average of the token vectors. For the second, it was computed from the weighted average of the token vectors, weighted with the TF-IDF value of each token. In all cases, the similarity of each pair of sentences was computed with the cosine of their vectors.

4 Training a Portuguese STS model

Based on the extracted features, various regression methods, with implementation available in scikit-learn [19], were explored for learning a STS model. Since the work on ASAPPpy started before the ASSIN 2 training collection was released, initial experiments towards the selection of the regression method were performed in the collection of ASSIN 1. It was also our goal to analyse whether the number of features could be reduced and its impact on the results. Experiments for this are reported in this section.

The results submitted to ASSIN 2 were obtained with the selected method, but also trained in the ASSIN 2 training collection, with features selected from the results in the validation collection. In all experiments, performance was assessed with the same metrics adopted in ASSIN and other STS tasks, namely Pearson correlation (ρ, between -1 and 1) and Mean Squared Error (MSE) between the values computed and those in the collection.

4.1 Selection of the Regression Method

Towards the development of the STS models in ASAPPpy, models were trained with different regression methods, in both the PTPT and PTBR training collections of ASSIN 1, and then tested individually on the testing collections of each variant. After initial experiments, three methods were tested, namely: a Support Vector Regressor (SVR), a Gradient Boosting Regressor (GBR) and a Random Forest Regressor (RFR), all using scikit-learn's default setup parameters.

Having in mind the efficiency of the model, we further tried to reduce the dimensionality of the feature set. For this purpose, we explored three types of feature selection methods, also available in scikit-learn: Univariate, Model-based and Iterative Feature Selection. In order to assess which method improved the performance of the model the most, in comparison to each other and to using all features, the model's coefficient of determination R^2 of the prediction was used for each method. Although we did not perform any measurement of the computational costs of these experiments, empirically we were able to assess that both Univariate and Model-based methods were significantly faster than Iterative Feature Selection when executed on the same machine.

We should add that, for these experiments, only 67 of the 71 features described in section 3 were considered. Four distributional features were only added later, namely those using Numberbatch embeddings and the binary vectors based on the presence of n-grams. With the aforementioned feature selection methods, the initial set of 67 features was reduced to 12, with marginal improvements in some cases, as the results in Tables 3 and 4 show. Although all selection methods were tested, the applied selection is the result of an Iterative Feature Selection, because it was the method leading to the highest performance. In the end, the selected features were: the Jaccard, Dice and Overlap coefficients for token 1-grams and character 3-grams; the Jaccard coefficient for character 2-grams; the cosine similarity between the sentence vectors computed using the TF-IDF matrix; the fastText.cc word embeddings; and the word2vec, fastText.cc and PTLKB word embeddings weighted with the TF-IDF value of each token. This means that the reduced model only uses lexical and distributional features. It does not use syntactic nor semantic features, though semantic relations should be captured by the distributional features, namely the word embeddings.

Tables 3 and 4 report the performance of each model on both ASSIN 1 PTPT and PTBR testing datasets, respectively before and after feature selection. The best performing model is based on SVR and achieved a Pearson ρ of 0.72 and MSE of 0.63, when tested on the PTPT dataset, using feature selection. For PTBR, ρ was 0.71 and MSE 0.37, for the same model.

Method	**PTPT**		**PTBR**	
	ρ	**MSE**	ρ	**MSE**
SVR	0.66	0.71	0.67	0.42
GBR	0.71	0.67	0.70	0.39
RFR	0.71	0.65	0.71	0.38

Table 3. Performance of different regression methods in ASSIN 1, before feature selection.

Method	**PTPT**		**PTBR**	
	ρ	**MSE**	ρ	**MSE**
SVR	**0.72**	**0.63**	**0.71**	**0.37**
GBR	0.71	0.66	0.70	0.39
RFR	0.72	0.64	0.71	0.38

Table 4. Performance of different regression methods in ASSIN 1, with feature selection.

4.2 ASSIN 2 STS Model

Although performed on the ASSIN 1 collection, experiments described in the previous section support the selection of the Support Vector Regressor (SVR) as the learning algorithm for the three runs submitted to ASSIN 2. All such runs were trained considering the same features and algorithm parameterisation, and were only different in the composition of the training data, which was the following:

- **Run #1** used all available data for Portuguese STS: ASSIN 1 PTPT/PTBR train and test datasets + ASSIN 2 train and validation datasets, comprising a total of ≈17,000 sentence pairs.
- **Run #2** considered that the ASSIN 2 data would be exclusively in Brazilian Portuguese, so did not use the ASSIN 1 PTPT data, comprising a total of ≈12,000 sentence pairs.
- **Run #3** had in mind that ASSIN 1 data could be different enough from ASSIN 2, thus not useful in this case, so used only the ASSIN 2 training and validation data, comprising a total of ≈7,000 sentence pairs.

Despite originally exploiting the full set of 71 features, all submitted runs were based on a reduced featured set. Features were selected based on the Pearson ρ of a model trained in all available data except the ASSIN 2 validation pairs, and validated in the latter pairs. In the end, models considered only 27 features, which were the 40% most relevant according to Univariate Statistics for different percentiles. In this case, this was the feature selection method that lead to the best performance. These are the 27 features effectively considered:

- Jaccard, Overlap and Dice coefficients, each computed between the sets of token 1/2/3-grams and character 2/3/4-grams.
- Averaged token vectors, computed with the following word embeddings: word2vec-cbow, GloVe (300-sized, from NILC [17]), fastText.cc [9], Numberbatch [22] and PT-LKB [15].
- TF-IDF-weighted averaged token vectors, computed with the following word embeddings: word2vec-cbow, GloVe (300-sized, from NILC [17]), fastText.cc [9] and Numberbatch [22].

Table 5 shows the official results of each run in the ASSIN 2 test collection. The best performance was achieved by run #3, with $\rho = 0.74$ and $MSE = 0.60$, despite the fact that this was the model that used the least amount of training data. Having no improvements with ASSIN 1 data is an indication of the (known) differences between the ASSIN 1 and ASSIN 2 collections. Such differences may explain the performance obtained in run #3, in which the data used for training, being exclusively from ASSIN 2, resulted in a model that could fit better the testing data. In opposition to the differences in the Pearson ρ, MSE was similar for every run, but slightly higher precisely for run #3.

After the evaluation, we repeated this experiment using the full set of 71 features, to conclude that using all features is not a good option. Pearson ρ achieved this way are equally poor and were 0.65, 0.66 and 0.66, respectively for

Runs	**#1**	**#2**	**#3**
ρ	0.726	0.730	0.740
MSE	0.58	0.58	0.60

Table 5. Official results of ASAPPpy in ASSIN 2 STS.

the configuration of the runs #1, #2 and #3. A curious result is that the MSE was significantly higher for run #3 configuration (0.85), the one trained only on the ASSIN 2 training data, when compared to the others (0.65 and 0.71). In the official results, run #3 had also the highest MSE, but only by a small margin.

4.3 Textual Entailment

Although it was not the primary focus of ASAPPpy, we tried to learn a classifier for textual entailment using the same features extracted for STS. Three models were trained, respectively with the features used in each run, with the configurations shown in the table 6. Yet, unlike the STS training phase, we chose to use the entire ASSIN 1 and training part of ASSIN 2 collection for the first two runs (≈17,000), selecting the best one (according to 10-fold cross-validation) to train a third model only on the training part of the ASSIN 2 dataset (≈7,000). Regarding the ASSIN 1 dataset, where there were three classes: Entailment, None and Paraphrase, this third class was considered as Entailment, in order to standardize the two datasets, since ASSIN 2 contains only the first 2 classes.

The performance of ASAPPpy in this task, below both baselines, is clearly poor. However, this was not the main goal of our participation. If more effort was dedicated to this task, we would probably analyse the most relevant features specifically for entailment, and possibly train new models from this knowledge.

Runs	**#1**	**#2**	**#3**
Training	ASSIN 1+2	ASSIN 1+2	ASSIN 2 (train)
Model	SVC	RFC	RFC
F1	0.401	0.656	0.649
Accuracy	53.10%	66.67%	65.52%

Table 6. Performance of the Textual Entailment models in ASSIN 2.

5 Conclusion

We described the participation of ASAPPpy in ASSIN 2 and explained some decisions that lead to using SVR-based models trained with a reduced set of lexical and distributional features. The main difference between the three submitted runs is the training data and the best performance ($\rho = 0.74$ and $MSE = 0.60$) was achieved by the model trained only in ASSIN 2 data. Using ASSIN 1 data

lead to no improvements, which supports the differences between the two collections. For instance, ASSIN 2 does not include complex linguistic phenomena nor named entities, which is not the case of ASSIN 1. But this does not necessarily mean that ASSIN 2 is easier, which is also suggested by the performance of our models, only slightly better in ASSIN 2.

We see the results achieved as satisfactory, at least for an approach based on traditional machine learning. Yet, they are clearly outperformed by approaches of other teams relying in deep learning or its products. On the other hand, our results can be interpreted, not only during the extraction of each feature, but also by applying feature selection during the training phase. For instance, features exploiting word embeddings and distance metrics between sentences shown to be the most relevant when computing the STS between phrases in Portuguese.

The curent version of ASAPPpy and its source code is available from `https://github.com/ZPedroP/ASAPPpy`. Still, in the future we would like to experiment with contextual word embeddings [11] given their recent positive performance in a set of different Natural Language Processing tasks. Pre-trained embeddings of that kind may be further fine-tuned on the ASSIN data, and be used alone as the representation of each sentence, or as additional features.

References

1. Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., Guo, W., Lopez-Gazpio, I., Maritxalar, M., Mihalcea, R., Rigau, G., Uria, L., Wiebe, J.: SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015). pp. 252–263. Association for Computational Linguistics, Denver, Colorado (Jun 2015), `https://www.aclweb.org/anthology/S15-2045`
2. Agirre, E., Diab, M., Cer, D., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: Proc. 1st Joint Conf. on Lexical and Computational Semantics-Vol. 1: Proc. of main conference and shared task, and Vol. 2: Proc. of 6th Intl. Workshop on Semantic Evaluation. pp. 385–393. Association for Computational Linguistics (2012)
3. Alves, A., Ferrugento, A., , M., Rodrigues, F.: ASAP: Automatic semantic alignment for phrases. In: SemEval Workshop, COLING 2014, Ireland. n/a (2014)
4. Alves, A., Rodrigues, R., Gonçalo Oliveira, H.: ASAPP: Alinhamento semântico automático de palavras aplicado ao português. Linguamática **8**(2), 43–58 (2016)
5. Alves, A., Simões, D., Gonçalo Oliveira, H., Ferrugento, A.: ASAP-II: From the alignment of phrases to textual similarity. In: 9th International Workshop on Semantic Evaluation (SemEval 2015). n/a (2015)
6. Alves, A., Gonçalo Oliveira, H., Rodrigues, R., Encarnação, R.: ASAPP 2.0: Advancing the state-of-the-art of semantic textual similarity for Portuguese. In: Proceedings of 7th Symposium on Languages, Applications and Technologies (SLATE 2018). OASIcs, vol. 62, pp. 12:1–12:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (June 2018)
7. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber,

G., Cudré-Mauroux, P. (eds.) The Semantic Web. pp. 722–735. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
8. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python. O'Reilly Media (2009)
9. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017)
10. Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., Specia, L.: SemEval-2017 task 1: Semantic Textual Similarity multilingual and crosslingual focused evaluation. In: Procs. of 11th Intl. Workshop on Semantic Evaluation (SemEval-2017). pp. 1–14. Association for Computational Linguistics (2017)
11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019)
12. Faruqui, M., Dodge, J., Jauhar, S.K., Dyer, C., Hovy, E., Smith, N.A.: Retrofitting word vectors to semantic lexicons. In: Proceedings of NAACL (2015)
13. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database (Language, Speech, and Communication). The MIT Press (1998)
14. Fonseca, E., Santos, L., Criscuolo, M., Aluísio, S.: Visão geral da avaliação de similaridade semântica e inferência textual. Linguamática **8**(2), 3–13 (2016)
15. Gonçalo Oliveira, H.: Learning word embeddings from portuguese lexical-semantic knowledge bases. In: Computational Processing of the Portuguese Language - 13th International Conference, PROPOR 2018, Canela, Brazil, September 24-26, 2018, Proceedings. LNCS, vol. 11122, pp. 265–271. Springer (September 2018)
16. Gonçalo Oliveira, H.: A survey on Portuguese lexical knowledge bases: Contents, comparison and combination. Information **9**(2) (2018)
17. Hartmann, N.S., Fonseca, E.R., Shulby, C.D., Treviso, M.V., Rodrigues, J.S., Aluísio, S.M.: Portuguese word embeddings: Evaluating on word analogies and natural language tasks. In: Proc 11th Brazilian Symposium in Information and Human Language Technology. STIL 2017 (2017)
18. Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., Zamparelli, R.: Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In: Proceedings of 8th International Workshop on Semantic Evaluation (SemEval 2014). pp. 1–8. Association for Computational Linguistics, Dublin, Ireland (2014)
19. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
20. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proc LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010)
21. Rodrigues, R., Gonçalo Oliveira, H., Gomes, P.: NLPPort: A Pipeline for Portuguese NLP. In: Proceedings of 7th Symposium on Languages, Applications and Technologies (SLATE 2018). OASIcs, vol. 62, pp. 18:1–18:9. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (June 2018)
22. Speer, R., Chin, J., Havasi, C.: Conceptnet 5.5: An open multilingual graph of general knowledge. In: Pro 31st AAAI Conference on Artificial Intelligence. pp. 4444–4451. San Francisco, California, USA (2017)