# ALB at SemEval-2018 Task 10: A System for Capturing Discriminative Attributes

**Bogdan Dumitru, Alina Maria Ciobanu, Liviu P. Dinu**
Faculty of Mathematics and Computer Science, University of Bucharest
Human Language Technologies Research Center, University of Bucharest
bogdan27182@gmail.com,
alina.ciobanu@my.fmi.unibuc.ro, ldinu@fmi.unibuc.ro

## Abstract

Semantic difference detection attempts to capture whether a word is a discriminative attribute between two other words. For example, the discriminative feature *red* characterizes the first word from the *(apple, banana)* pair, but not the second. Modeling semantic difference is essential for language understanding systems, as it provides useful information for identifying particular aspects of word senses. This paper describes our system implementation (the ALB system of the NLP@Unibuc team) for the 10th task of the Semaval 2018 workshop, "Capturing Discriminative Attributes". We propose a method for semantic difference detection that uses an SVM classifier with features based on co-occurrence counts and shallow semantic parsing, achieving 0.63 F1 score in the competition.

## 1 Introduction and Related Work

Semantic similarity detection is a well-studied research problem with numerous applications. State of the art models are extremely capable, determining the degree of semantic similarity between words with high accuracy.

However, looking the other way around, the semantic difference between words has received significantly less attention. As a result, one can argue that a semantic similarity model without the capability of spotting the semantic difference as well is not a complete system, and may not prove very useful in practice.

Semantic difference is a ternary relation $(w_1, w_2, w_3)$, where $w_1$ and $w_2$ are called *concepts* and $w_3$ is called *discriminative feature*. The discriminative feature characterizes the first concept, $w_1$, but not the second one, $w_2$. If the discriminative feature characterizes both $w_1$ and $w_2$ or none of them, then we do not have semantic difference.

Semantic difference detection is a binary classification task where given a triplet of words, a model needs to determine if a semantic difference is present or not. As emphasized by Krebs and Paperno (2016), this non-trivial task has numerous applications, such as automatized lexicography, conversational agents or machine translation.

Most research on discriminative features is related to computer vision (Farhadi et al., 2009; Russakovsky and Fei-Fei, 2012), as these attributes proved to be very useful in interpreting visual data (Huang et al., 2016), being able to link visual features and semantic labels (Guo et al., 2015). A recent study on this topic belongs to Lazaridou et al. (2016), who proposed a method for identifying discriminative attributes when given word pairs and their visual representations.

In this paper, we describe a system for semantic difference detection that outputs a set of features for every triplet in the input data, based on preprocessed external resources (the English Wikipedia database). Further, these features are used to train an SVM for binary classification. The current feature selection allows even a direct approach such as evaluating the following inequation:

$$\sum_{i=1}^{|F_1|} f_i - \sum_{i=1}^{|F_2|} f_i > 0 \qquad (1)$$

to obtain similar results as the SVM. Here, $F_1$ and $F_2$ are values of the same features, extracted for $(w_1, w_3)$ and $(w_2, w_3)$, respectively.

Our model uses two different classes of features. The first class is generated using simple co-occurrence counts and the second class is generated by an arc-factored approach (McDonald et al., 2005) for semantic dependency parsing.

Semantic dependency parsing aims to provide a shallow semantic analysis of the text. As distinct



*Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 963–967
New Orleans, Louisiana, June 5–6, 2018. 

from deeper semantic analysis, shallow semantic parsing captures relationships between pairs of words or concepts in a sentence (Thomson et al., 2014).

## 2 Dataset and Preprocessing

The input data (training, validation and testing) is translated into an intermediary configuration as described below.

Each word triplet from the input data is split into two word pairs: $(w_1, w_3)$ and $(w_2, w_3)$. The initial part of our model extracts features for each pair, and in the last steps, where the performance is computed, a cross-reference is done with the original input database.

We use the English Wikipedia as an external data source for feature extraction. We convert the raw Wikipedia database to plain text and concatenate the sentences of all the articles in a large text corpus.

## 3 System Framework

In this section we present our approach and methodology for capturing discriminative attributes.

### 3.1 Problem Reduction

First, we transform the problem of semantic difference detection into a simpler one: detecting if a feature characterizes a concept. Every ternary relation in the input data is split into two subproblems of detecting if a feature (i.e. $w_3$) characterizes $w_1$ and $w_2$, respectively. Solving subproblems independently gives us more flexibility in feature extraction.

Determining the validity of the ternary relation from the outputs of the newly created binary relations is achieved with the following equation:

$$o = \neg(p \Longrightarrow q) \tag{2}$$

which for convenience can be rewritten as:

$$o = p \wedge \neg q \tag{3}$$

where $p = C(w_1, w_3)$, $q = C(w_2, w_3)$, $o$ is the triplet label and $C(w_a, w_b)$ is the model or function that decides if $w_b$ characterizes $w_a$.

### 3.2 Features

We use two categories of features in training our system: *co-occurrence* and *POS-tag* features. Features from both categories are extracted from English Wikipedia sentences.

#### 3.2.1 Co-occurrence

This is a measure of occurrence of two words in a text alongside each other and in a specific order. For our system, we consider the two words as an unordered pair and count accordingly. For every pair of words $(w_1, w_2)$ we extract the following features:

- *Co-occurrence$_1$*: counts the number of adjacent occurrences of $w_1$ and $w_2$ disregarding the order.
- *Co-occurrence$_2$*: counts the number of occurrences of both $w_1$ and $w_2$ in a text window of size 2.
- *Co-occurrence$_3$*: counts the number of occurrences of both $w_1$ and $w_2$ in a text window of size 3.

If two words occur in the same sentence, it provides the intuition that there should be a relation between the distance $d = |w_1 - w_2|$ and their semantic relation. This is what we attempt to capture with the features described above. We drop *co-occurrence$_2$* and *co-occurrence$_3$* in our final system configuration, since they do not add any contribution to the final score, as shown in Table 2.

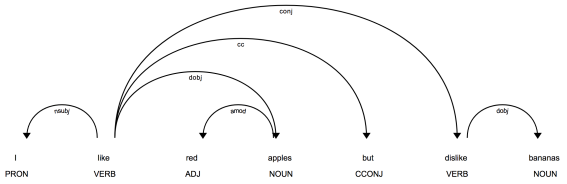


Figure 1: Syntactic dependency tree.

#### 3.2.2 POS-tag Features

Every sentence that contains a pair of words $(w_1, w_2)$ is parsed and tagged. Based on a statistical model, a prediction is made of which tag applies to each word. Next, a syntactic dependency tree is built as shown in Figure 1 and used to derive various rules that are used to extract POS-related features.

For every sentence containing both words of the pair, if a rule $R_i$ hits, then we increment the total hit count $H_c$ of that specific pair. $|R|$ represents the total number of rules to be applied and $H_c$ is defined by the following formula:

$$H_c = \sum_{i=1}^{|R|} R_i(w_1, w_2) \quad (4)$$

As an example, if we take the pair of words *($w_1$ = moth, $w_2$ = flies)*, we obtain the following values: 1,088 *co-occurrence$_1$* and a count of 763 hits of the rules on sentences containing both words. Hence the pair *(moth, flies)* has feature values (1,088, 763). Several more examples are presented in Table 1.

| $w_1$, $w_2$ | *co-occurrence*$_1$ | $H_c$ |
|---|---|---|
| desk, drawers | 169 | 120 |
| cod, honks | 0 | 0 |
| shirt, sleeves | 450 | 1,109 |
| tie, sleeves | 16 | 13 |
| lime, holes | 0 | 3 |
| cheese, holes | 29 | 20 |

Table 1: Examples of word pairs and feature values.

### 3.3 Rules

We perform rules implementation in a purely heuristic manner, using methods from previous research (Kübler et al., 2009). While we keep rules composition simple, they turn out to be very powerful and we use only two of them in the final system implementation. Rules are prone to both false positives and false negatives, but provided enough input sentences, both errors tend to be minimized.

- *Rule$_1$*: if $w_2$ is the root of an arc in the parsing tree and $w_1$ is one of its children and no negation is present in the children list, then the rule will return a hit.

- *Rule$_2$*: if the child of a root noun is a verb, then recursively the children of the verb will be considered related to the noun and the pairs (*root_noun, verb_child*) will be compared with ($w_1$, $w_2$), increasing the hit count if the pairs match.

### 3.4 Linear SVM

For classification we use a linear SVM (Vapnik, 1995). The output of the SVM is given by the equation:

$$u = wx - b \quad (5)$$

where $w$ is the normal vector to the hyperplane and $x$ represents the input data.

In the linear case, the margin is defined as the distance between the closest positive and negative example, and the hyperplane defined by the above equation (see Figure 2). Maximizing the margin can be approached as an optimization problem:

$$\text{Minimize } \tfrac{1}{2}\|w\|^2 \text{ subject to } y_i(wx_i - b) \geq 1, \forall i;$$

where $x_i$ is the $i$th training sample and $y_i$ is the correct classification of the sample.

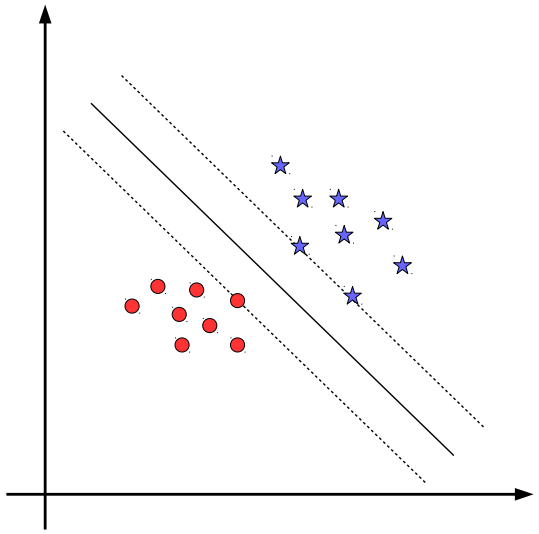

Figure 2: Linear SVM.

### 3.5 System Workflow

Until now we have described all the building blocks of our system. Now we chain them together.

The first step is to transform the original datasets into datasets of pairs. For all training, validation and testing data we run Algorithm 1. After this step, we end up with a dataset containing the features of all pairs, of all three datasets.

The next step is to train and validate the SVM on the datasets of triplets. The last step is to use the trained SVM to predict labels for the triplets in

**for** *every dataset* **do**
  transform data into pairs of words;
  **for** *every pair of words* **do**
    extract all sentences from Wikipedia containing $w_1$ and $w_2$;
    extract co-occurrence features;
    **for** *every sentence* **do**
      run active rules;
    **end**
    compute $H_c$ using active rules;
  **end**
**end**

**Algorithm 1:** Dataset preparation.

the test dataset. A slightly different approach that we try is to train the SVM on pairs extracted from triplets with label 1, and then apply Equation 2 to obtain the labels for the initial test dataset triplets.

In another system configuration, we eliminate the SVM and compute the final triplet score from the existing features using Equation 1. By doing so, we eliminate the training and validation steps, thus transforming our system from a learning one to a purely deterministic one. However, if the number of features is increased, such an approach may prove unfeasible and inefficient.

## 4 Results

We have implemented several system configurations by selecting different rules, features and learning methods. We have chosen three configurations: two of them produced the top results in our experiments on the development dataset, and the other had the peculiarity of not having a learning mechanism. The performance of these systems on the development dataset is reported in Table 2. The systems are evaluated using the F1 score.

The first system, *ALB*, uses only the first co-occurrence score, along with $H_c$. Even if only two features per pair of words are used, this system configuration produced the best F1 score of 0.69. This is the only system that we submitted for evaluation, obtaining 0.63 F1 score on the test dataset.

The second system, *ALB+*, uses all three co-occurrence scores as features and treats rules output as separate features. Both *ALB* and *ALB+* use the SVM trained on triplets.

The third system, *EQ1*, uses the same two features as *ALB* and replaces the SVM component with Equation 1. This system obtained the lowest F1 score, but not too distant from the others.

It is interesting to mention that if we use only *co-occurrence*$_1$ as a discriminant, the score is $> 0.6$. Analyzing the output of our best system, we observe that the errors it produces are not biased towards one of the labels (417 errors for label 0 and 430 for label 1).

| System | F1 Score |
|---|---|
| **ALB** | **0.69** |
| ALB+ | 0.67 |
| EQ1 | 0.62 |

Table 2: Results for capturing discriminative attributes on the validation dataset.

## 5 Conclusions

In this paper we have presented our results and system description for Task 10 of SemEval 2018, "Capturing Discriminative Attributes".

Our approach shows promising results in using the relation between words in context for semantic differences. The obtained results are competitive, although being outperformed by other approaches in the official ranking. There is enough room for improvements and at least two possible approaches are already being analyzed.

The first one is straightforward: extending the feature set with at least one order of magnitude compared with *ALB+*, and if necessary replacing the SVM with a fully connected neural network. The heavily used sequence-to-sequence model can also be applied on sentences to automatically capture relations between word pairs.

The second possible approach is to use a neural network to automatically infer rules. Next, we can apply generated rules to compute $H_c$ and assign a semantic difference probability to every pair in our dataset. We can use a pruned version of the training dataset from this task, extract word pairs in the same manner as we did in our system implementation and feed word pairs along with sentences and labels to a convolutional neural network.

## 6 Acknowledgments

Research supported by UEFISCDI, project number 53BG/2016.

## References


Ali Farhadi, Ian Endres, Derek Hoiem, and David A. Forsyth. 2009. Describing Objects by Their Attributes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1778–1785.

Yuchen Guo, Guiguang Ding, Xiaoming Jin, and Jianmin Wang. 2015. Learning Predictable and Discriminative Attributes for Visual Recognition. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3783–3789.

Chen Huang, Chen Change Loy, and Xiaoou Tang. 2016. Unsupervised Learning of Discriminative Attributes and Visual Representations. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 5175–5184.

Alicia Krebs and Denis Paperno. 2016. Capturing Discriminative Attributes in a Distributional Space: Task Proposal. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 51–54.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan and Claypool Publishers.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2016. "The red one!": On Learning to Refer to Things Based on Discriminative Properties. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 213–218.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-margin Training of Dependency Parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98.

Olga Russakovsky and Li Fei-Fei. 2012. Attribute Learning in Large-Scale Datasets. In *Trends and Topics in Computer Vision*, pages 1–14. Springer Berlin Heidelberg.

Sam Thomson, Brendan O'Connor, Jeffrey Flanigan, David Bamman, Jesse Dodge, Swabha Swayamdipta, Nathan Schneider, Chris Dyer, and Noah A Smith. 2014. CMU: Arc-Factored, Discriminative Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 176–180.

Vladimir Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag New York.