ISSN 2007-9737

Automatic Detection of Semantic Classes of Verb-Noun Collocations

Olga Kolesnikova[1], Alexander Gelbukh[2], Liliana Chanona-Hernández[3]

[1] Instituto Politécnico Nacional, Escuela Superior de Cómputo, Mexico

[2] Instituto Politécnico Nacional, Centro de Investigación en Computación, Mexico

[3] Instituto Politécnico Nacional, Escuela Superior de Ingeniería Mecánica y Eléctrica, Mexico

gelbukh@gelbukh.com, {kolesolga, lchanona} @gmail.com

Abstract. It does not surprise us that a bank can be a financial institution as well as a piece of land. Quite often one word is used with different meanings. But sometimes the opposite happens: we choose different words to express the same idea. For example, to give a smile means ‘to smile’, and to lend support means ‘to support’ (Longman Dictionary of Contemporary English, 1995). These two collocations convey the same idea: to smile is to ‘perform’, or ‘do’ a smile, and to support is to ‘do’ support, so that both verb-noun collocations share the same semantics: to do what is denoted by the noun. Likewise, we find that to acquire popularity and to sink into despair both mean ‘to begin to experience the <noun>’, and to establish a relation and to find a solution mean ‘to create the <noun>’. Such semantic patterns or classes are called lexical functions. In this article, we explain the concept of lexical functions, give a summary of state-of-the-art research on automatic detection of lexical functions, and present the framework and results of our experiments on supervised learning of lexical functions fulfilled on the material of Spanish verb-noun collocations.

Keywords. Verb-noun collocations, lexical functions, semantic classification, supervised machine learning.

1 Introduction

Collocations, or recurrent word combinations often used together in a language, present a challenge to natural language analysis and natural language generation due to their limited compositionality and idiomatic nature [10]. Knowledge of collocation should be integrated into natural language systems to avoid incorrect interpretation and production of texts. First, collocations are to be distinguished from free word combinations because these two groups of language phenomena are handled differently. A number of collocation extraction tools and methods have been developed so far [5, 15, 19]. Secondly, it is useful to annotate collocations with grammatical and semantic information and store them in databases and machine readable dictionaries which then can be incorporated in various applications [3, 18].

It is typical for collocation inventories to include such grammatical information as part-of-speech of collocation constituents. A good example is *Oxford Collocations Dictionary for Students of English* [12], the most widely used English collocation dictionary. However, collocational databases are still in need of semantic information markup to represent the opaque meaning of collocations explicitly. One possible solution is to define collocations in the way a common monolingual dictionary does it, i.e. providing definitions and usage examples. However, such approach requires a lot of manual work, time, and financial resources. Another way is to retrieve semantic information automatically.

This is followed by a question: What type of semantic information can be retrieved and how? In this work, we suggest that the semantics of collocations (in particular, we study verb-noun collocations) can be generalized and presented as abstract meanings found in collocational groups.

Each group is then characterized by a single and distinct abstract meaning, thus forming a semantic class. We use the formalism of lexical functions explained in Section 2 to encode the generalized semantics of collocations and apply supervised machine learning methods to detect lexical functions automatically.

As it has just been mentioned, Section 2 presents the concept of lexical function by giving its definition, explaining its notation, and illustrating these with examples. By way of introduction, we will only note here that lexical functions capture similar semantic features of groups of collocations. In addition, here we give some anticipative insight into the concept via examples in the paragraph that follows.

The fact that one word can have many senses is well known in linguistics. However, sometimes the opposite happens: we choose different words to express the same idea, but the choice is purely lexical and not semantically motivated. For example, *to give a smile* means ‘to smile’ or ‘perform a smile’. When we want to say ‘to perform’ *support*, we can also use the verb *to give*: *to give support.* The verb *to lend* is also used with *support* in the same meaning: *to lend support* is *to give support*.

These collocations share the same semantics: to do what is denoted by the noun. Likewise, we find that *to acquire popularity* and *to sink into despair* both mean ‘to begin to experience the <noun>’, and *to establish a relation* and *to find a solution* mean ‘to create the <noun>’.

Such semantic patterns or classes are called lexical functions. The latter can serve as an instrument to generalize the meaning of collocations and to construct a semantic classification of collocations. A database where collocations are annotated with lexical functions will be a valuable resource for natural language applications.

The rest of the paper is organized as follows. Section 2 explains the concept of lexical function, Section 3 reviews state-of-the-art results in automatic detection of lexical functions. Section 4 describes the setting of our experiments including data, data representation, and methods. It also gives results we achieved and discusses them. Section 5 presents conclusions.

2 Lexical Functions

2.1 Definition, Notation, and Examples

Lexical function (LF) is a concept developed by Mel’čuk [11] as a part of the Meaning-Text Theory to capture lexical relations among words. (A brief description of the Meaning-Text Theory can be found in [7]). Lexical function is a mapping from one word called **the keyword** to another it collocates with in corpora called **the lexical function value**. For example, in *to make an announcement*, the keyword is *announcement***,** and the lexical function value is *to make*. This mapping is further characterized by its meaning encoded by an abbreviated Latin word with the semantics similar to the lexical function meaning.

The collocation given above represents the lexical function named Oper$_1$. ‘Oper’ is from Latin *operari*, to ‘do’, ‘perform’, ‘carry out’ (what is denoted by the noun). That is, to express the meaning ‘to perform an announcement’, one says in English *to make an announcement*. Other examples of Oper$_1$ are *to break the news*, *to narrate a story*, *to deliver a message*, *to perform a drama*, *to give a smile*, *to lend support*, *to mount resistance*, *to have authority*, *to exercise power*. All these collocations share the common semantic pattern ‘to do the <noun>’.

The LF name is followed by a subscript representing syntactic functions of words or phrases which lexicalize semantic roles: 1 stands for the agent, 2, for recipient/patient, etc. The subscript 1 in Oper$_1$ means that the agent of *announcement* in the above collocation is the grammatical subject: *He* (agent) *has made an announcement*. Using the notation of lexical function, *to make announcement* is re-written as Oper$_1$(*announcement*) = *make*, *to execute a program* as Oper$_1$(*program*) = *to execute*, *to run a risk* as Oper$_1$(*risk*) = *to run*.

However, *to receive treatment* is represented as Oper$_2$(*treatment*) = *to receive*, the subscript 2 signifies that the recipient/patient is the subject:
He has received an effective treatment and feels well now. Another example is Oper$_2$(*attention*) = *to gain*: *The author gained the attention of the audience*. The subscript 0 means that the keyword itself funcitons as the subject: Func$_0$(*snow*) = *to fall*,

Snow was falling all day long, Func is from Lat. *functionare*, to function.

More details on LF notation is given by Mel'čuk [11]. He describes 64 LFs identified in collocations of various syntactic structures (adjective-noun, noun-noun, verb-noun, verb-adverb, etc.) and illustrates them with abundant English material. Since our work is done on verb-noun collocations, we concentrate on verb-noun lexical functions and use examples of such functions and collocations.

Besides Oper$_1$, Oper$_2$, and Func$_0$ mentioned above, we give another two examples of verb-noun LFs. The first is Func$_{01}$(*joy*) = *to fill* [somebody]; here 0 says that the keyword is the subject and 1 means that the agent of the keyword (*joy*) is the verb's object: *When she saw her friend, a sudden joy filled her*.

The second is Real$_1$, from Lat. *realis*, real. Real$_1$ means 'to fulfill the requirement of the <noun>', or 'to do with the <noun> what one is supposed to do with the <noun>, and the examples are *to prove an accusation*, *to drive a bus*, *to succumb to an illness* [11].

2.2 Simple and Complex Lexical Functions

Lexical functions can represent single as well as multiple semantic features or 'units of meaning'. Simple LFs represent a single semantic element: 'do' (Oper), 'function' (Func), 'fulfill the requirement' (Real).

Other examples of simple LFs are: Incep, from Lat. *incipere*, to begin; Cont, from Lat. *continuare*, to continue; Fin, from Lat. *finire*, to cease; Caus, from Lat. *causare*, to cause. Complex LFs are combinations of simple LFs.

IncepOper$_1$ means 'to begin to do the <noun>' in to enter into marriage, to acquire a right, to assume a role, to reach the age, to fall into a trance, to develop ability.

ContOper$_1$ means 'to continue to do the <noun>' in to cultivate a relationship, to develop cooperation, to nourish friendship, to nurture an attitude, to maintain the balance.

CausFunc$_{01}$ means 'to cause that the <noun> functions' in to stir up interest, to raise hope, to inspire love, to assign a rank, to bring luck, to impose a fine.

2.3 Lexical Functions for Verb-Noun Collocations

As it is said in Section 2.1, 64 LFs have been defined to represent a wide range of collocations of different structure: adjective-noun, noun-noun, verb-noun, verb-adverb, etc. These LFs are simple. Our experiments on automatic detection of lexical functions has been done for Spanish verb-noun collocations.

Due to data we have at our disposal (see Section 4.1), we have got a sufficient number of verb-noun collocations only for eight lexical funcitons listed in Table 1. Therefore, these LFs were chosen for machine learning experiments.

2.4 Lexical Functions as Semantic Classes of Collocations

It was mentioned in Section 1 that lexical functions may be applied to build a semantic typology of collocations. This typology will be rather fine-grained and include at least 64 classes equal to the number of simple LFs as indicated in Section 2.2, not to mention complex LFs. Each class can be characterized by the semantics of its corresponding LF and a semantic pattern (like 'to do the <noun>) which also stores syntactic information. The latter gives quite a comprehensive description of a group of collocations which desambiguates them and distinguishes them from any other class since such description is unique for each class.

3 Automatic Detection of Lexical Functions: State-of-the-Art

Wanner (2004) suggested to regard the task of LF automatic detection as classifying collocations according to LF typology and applied nearest neighbor technique to resolve it. Experiments were done on two groups of Spanish verb-noun collocations: one with emotion nouns and the other with field-independent nouns. LF learning was performed using hypernym information from the Spanish part of EuroWordNet [22].The maximum F1-measure achieved for field-independent collocations was 0.76 on CausFunc$_0$. The average F1-measure was about 0.70.

The same average result was shown on more data in [23] over four learning methods: nearest neighbor, Naïve Bayes, tree-augmented network, and ID3-algorithm. This time the best F1-measure on field-independent verb-noun collocations (0.76) was achieved by ID3-algorithm.

Archer [2] performed experiments on extracting collocations of the adjectival/adverbial lexical function Magn, from Lat. *magnus*, big, great, using Conceptual Vectors [14]. Magn values are intensifiers, for example: Magn(*rain*) = *heavy*, Magn(*temperature*) = *high*, Magn(*storm*) = *sever*, Magn(*to weep*) = *bitterly*. The research was done on French material. The reported result is a precision of 83%. However, the process of extracting Magn collocations was semi-automatic and included human intervention, which improved the result significantly.

Alonso Ramos et al. [1] proposed an algorithm to retrieve collocations of the type 'support verb + object' from the semantically annotated FrameNet corpus of examples [13]. Their interest was to see whether the collocations they extracted were of Oper_n. The authors assumed that certain syntactic, semantic, and collocation annotations in the FrameNet corpus could signal that a particular collocation belonged to Oper_n. The proposed algorithm was tested on 208 instances and showed an accuracy of 76%.

In [4] it is discussed how to use dictionary definitions for extracting actants, which is very similar to LF and can be used in our future work.

4 Our Experiments on Supervised Learning of Lexical Functions

4.1 Data

Spanish verb-noun collocations were extracted automatically from the Spanish Web Corpus via SketchEngine [8], web-based software for automatic text processing. The Spanish Web Corpus is compiled of texts found in the Internet. The texts are not limited to particular themes so the corpus represents the general Spanish lexis, therefore its lexical data can be considered field-independent. The collocations were manually annotated with LFs and Spanish WordNet senses [21]. Our data is available on the web[1].

We found that in our data eight LFs were represented by the number of collocations sufficient for machine learning experiments, as it was mentioned in Section 2.3. Table 1 presents these LFs together with the number of their instances, corresponding semantic patterns, and Spanish examples followed by English translations.

4.2 Data Representation

Hypernyms of all words in collocations were used as features for LF learning. The source of hypernym information was the Spanish WordNet [21]. It is an online lexical resource and is used widely in the natural language processing community. We used a vector space model with binary representation of features assigning each feature the value of 1 if a hypernym was present among hypernyms of the verb or the noun in a collocation, and the value of 0 if it was absent.

Thus, every collocation was represented as a vector of binary values having the size equal to the number of all hypernyms extracted for all words in all LF samples. Each of eight lexical functions was assigned its training set. All eight training sets included the same collocations, with the difference of marking collocations belonging to the LF in question as positive examples (class "yes") and the collocations belonging to the other seven LFs as negative examples (class "no").

4.3 Methodology

WEKA 3-6-2 toolkit [6] was used for experimenting with many supervised learning algorithms. We supplied the training sets of eight LF to 68 classifiers and evaluated predictions of the positive class on the same sets applying 10-fold cross-validation technique and F1-measure. Below all classifiers tested for LF prediction are listed. The classifiers are grouped according to their classes.

Class `bayes`: AODE, AODEsr, BayesianLogisticRegression, BayesNet, HNB,

[1] http://148.204.58.221/okolesnikova/index.php?id=lex/ or http://www.gelbukh.com/lexical-functions

Table 1. LFs and semantic patterns of collocations in our data

LF	# of instances	Semantic pattern	Spanish collocation	English translation
$Oper_1$	266	to carry out the <noun>, to experience the <noun>	*prestar atención* *tener una duda* *celebrar la reunión*	to pay attention to have doubt to have a meeting
$Oper_2$	28	to undergo the <noun>, to be source of the <noun>	*obtener beneficio* *sufrir un ataque* *recibir la respuesta*	to get a benefit to have an attack to get the answer
$IncepOper_1$	24	to begin to do, perform, experience, carry out the <noun>	*asumir la responsabilidad* *iniciar un proceso* *tomar la iniciativa*	to take on the responsibility to start a process to take the initiative
$ContOper_1$	16	to continue to do, perform, experience, carry out the <noun>	*mantener el contacto* *seguir el curso* *guardar silencio*	to keep in contact to follow the course to keep silent
$Func_0$	16	the <noun> exists, takes place, occurs	*la posibilidad existe* *el día pasa* *la duda cabe*	the possibility exists the day passes by a doubt arises
$CausFunc_0$	109	the agent of the <noun> causes the <noun> to occur	*poner un ejemplo* *dar explicación* *provocar la reacción*	to give an example to give an explanation to provoke a reaction
$CausFunc_{01}$	89	a person/object, different from the agent of the <noun>, causes the <noun> to occur	*dar importancia* *abrir paso* *producir daño*	to give importance to make way to inflict damage
$Real_1$	60	to fulfill the requirement of the <noun>, to act according to the <noun>	*utilizar la herramienta* *corregir el error* *alcanzar el nivel*	to use a tool to correct an error to reach a level

NaiveBayes, NaiveBayesSimple, NaiveBayesUpdateable, WAODE.

Class `functions`: LibSVM, Logistic, RBFNetwork, SimpleLogistic, SMO, VotedPerceptron, Winnow.

Class `lazy`: IB1, IBk, KStar, LWL.

Class `meta`: AdaBoostM1, AttributeSelectedClassifier, Bagging, ClassificationViaClustering, ClassificationViaRegression, CVParameterSelection, Dagging, Decorate, END, EnsembleSelection, FilteredClassifier, Grading, LogitBoost, MultiBoostAB, MultiClassClassifier, MultiScheme, OrdinalClassClassifier, RacedIncrementalLogitBoost, RandomCommittee, RandomSubSpace, RotationForest, Stacking, StackingC, ThresholdSelector, Vote.

Class `misc`: HyperPipes, VFI.

Class `rules`: ConjunctiveRule, DecisionTable, JRip, NNge, OneR, PART, Prism, Ridor, ZeroR.

Class `trees`: ADTree, BFTree, DecisionStump, FT, Id3, J48, J48graft, LADTree, RandomForest, RandomTree, REPTree, SimpleCart.

4.4 Experimental Results and Discussion

As it is mentioned in Section 4.3, the performance of supervised learning algorithms was evaluated in

Table 2. Best classifiers on LF detection

LF	#	Baseline	Classifier	F1
$Oper_1$	266	0.437	BayesianLogisticRegression	**0.873**
			Id3	0.870
			SMO	0.864
$Oper_2$	28	0.046	J48	**0.706**
			LogitBoost	0.686
			LADTree	0.667
$IncepOper_1$	24	0.040	Prism	**0.774**
			NNge	0.727
			SMO	0.722
$ContOper_1$	16	0.026	DecisionTable	**0.833**
			FilteredClassifier	0.800
			Ridor	0.783
$Func_0$	16	0.026	BFTree	**0.696**
			HyperPipes	0.636
			AttributeSelectedClassifier	0.636
$CausFunc_0$	109	0.179	JRip	**0.725**
			EnsembleSelection	0.699
			REPTree	0.695
$CausFunc_{01}$	89	0.146	END	**0.762**
			LogitBoost	0.756
			AttributeSelectedClassifier	0.733
$Real_1$	60	0.099	FT	**0.598**
			NNge	0.593
			Id3	0.587
Total #	608		Average best	**0.746**

terms of F1-measure using 10-fold cross-validation technique. Table 2 presents the results of three best classifiers for each LF, # stands for the number of LF positive instances, F1 stands for F1-measure, and the highest F1-measure for each LF is in boldface.

Usually, in machine learning experiments, ZeroR is chosen as the baseline. ZeroR is a trivial classifier that assigns the majority class to all examples. In our experiments, the majority class is always "no" since the number of negative examples for each lexical function is bigger than the number of its positive examples. For example, the number of positive examples for $Oper_1$ is 266 and the number of its negative examples is 608 – 266 = 342. The number of positive examples for the rest seven lexical functions is even less than for $Oper_1$ as it is seen from Table 2, therefore, ZeroR has no sense as the baseline.

However, the baseline can be a random choice of a positive or a negative answer to the question "Is this collocation of this particular lexical function?" In such a case we deal with the probability of a positive and negative response. Since we are interested in only assigning the positive answer to a collocation, we calculate the probability of "yes" class for eight lexical functions in the experiments according to the formula:

probability of “yes” = (the number of positive examples of a given lexical function) / (the number of all examples). These probabilities will be results of a classifier that assigns the class “yes” to collocations at random assuming a uniform probability distribution. Since we will compare the probabilities of the random choice with the results obtained in our experiments, we present the former as numbers within the range from 0 to 1 in Table 2.

As it is seen from Table 2, no single classifier is the best one for detecting all LFs. For each LF, the highest result is achieved by a different classifier. The maximum F1-measure of 0.873 is achieved by BayesianLogisticRegression classifier for Oper$_1$. The lowest best F1-measure of 0.598 is shown by FT for Real$_1$. The average best F1-measure (calculated over only the eight best results, one for each LF) is 0.746.

No correlation was observed between the number of instances in the training set and the results obtained from the classifiers. For example, a low result of 0.598 is shown for Real$_1$ which has more positive instances (60) than ContOper$_1$ (only 16), but for ContOper$_1$ a high result of 0.833 was attained. At the same time, Func$_0$ with the equal number of positive instances as ContOper$_1$ (16) was detected with an F1-measure of 0.696 which is low. Similar results were showed for CausFunc$_{01}$ (F1-measure of 0.762) and IncepOper$_1$ (F1-measure of 0.774), though CausFunc$_{01}$ has 89 positive instances and IncepOper$_1$, only 24.

It is also seen from Table 2 that the best classifiers for all LFs except for Oper$_1$ are based on symbolic, not statistical, learning. `J48`, the best method for detecting Oper$_2$, is a rule-based classifier algorithm that generates C4.5 decision trees, which in their turn implement ID3 algorithm. `Prism`, the best classifier for IncepOper$_1$, is based on the inductive rule learning and uses separate-and-conquer strategy.

`DecisionTable`, the best for ContOper$_1$, is an induction algorithm whose task is to find the optimal list of attributes such that the decision table created from this list will have the lowest possible error on the training data. `BFTree`, the best method for Func$_0$, is a best-first decision tree learner – an alternative approach to standard decision tree techniques such as C4.5 algorithm since they expand nodes in best-first order instead of a fixed depth-first order. `JRip`, the best algorithm for CausFunc$_0$, implements a propositional rule learner RIPPER. `END`, the best for CausFunc$_{01}$, is a meta-classifier for handling multi-class datasets with two-class classifiers by building an ensemble of nested dichotomies. `FT`**,** the best for Real$_1$, is a classifier for building functional trees, which are classification trees that could have logistic regression functions at the inner nodes and/or leaves. More details on classifiers can be found in [23].

Since almost all best classifiers for LF detection are symbolic, we can suppose that collocational semantics is better distinguished by rules and decision-making procedures than based on probabilistic knowledge learned from the training data. Statistical methods are limited by the assumption that all features in data are equally important in contributing to the decision of assigning a particular class to an example and as well independent of one another. This is a rather simplified view of data, because in many cases, data features are not equally important or independent and this is certainly true for linguistic data, especially for such a language phenomenon as hypernyms. Graphically, hypernyms form a tree, a hierarchic structure where every hypernym has its ancestor (except for the hypernym at the root of the tree) and a child or children (except for hypernyms in the leaf nodes of the tree).

Table 3 presents some of the best results reported in [22, 23], and our best results. No comparison is fair because the experiments were not run on the same dataset though in all cases verb-noun collocations were dealt with. In Table 3, W04 stands for [22], W06 stands for [23], F1 stands for F1-measure, # signifies the number of instances for a given LF, NB is the nearest neighbor technique, NB is Naïve Bayesian network, BLR is BayesianLogisticRegression

Table 4 shows performance of classifiers commonly used for natural language processing: Naive Bayes, C4.5 decision tree learner, nearest-neighbor instance-based learner, and support vector machine. In this table, W06 stands for [23], BLR stands for BayesianLogisticRegression, and DT stands for DecisionTable.

The column ‘Best’ presents the best result achieved in our research by classifiers specified after the **|** symbol.

Table 3. Best results in [22, 23] and our best results for some LFs

LF	# in W04	Result in W04,F1	# in W06	Result in W06		# in our data	Our result	
				F1	Method		F1	Method
Oper$_1$	50	**0.609**	87	**0.737**	NB	266	**0.873**	BLR
Oper$_2$	48	**0.759**	48	**0.662**	NN	28	**0.706**	J48
CausFunc$_0$	53	**0.766**	53	**0.676**	NN	109	**0.725**	JRip
Real$_1$	52	**0.741**	52	**0.500**	NN	60	**0.598**	FT
Average		**0.719**	**0.644**		**0.726**			

Table 4. Performance of four classifiers most frequently used in natural language processing on LF predicting. The numbers are F1-measure values

LF	NB in W06	NB	J48	IB1	SMO	Average	Best
Oper$_1$	0.737	0.711	0.844	0.620	0.864	0.760	0.873 | BLR
Oper$_2$	0.304	0.000	0.706	0.327	0.595	0.407	0.706 | J48
IncepOper$_1$		0.000	0.571	0.357	0.722	0.413	0.774 | Prism
ContOper$_1$		0.000	0.800	0.333	0.750	0.471	0.833 | DT
Func$_0$		0.000	0.636	0.348	0.583	0.392	0.696 | BFTree
CausFunc$_0$	0.589	0.308	0.609	0.387	0.643	0.487	0.725 | JRip
CausFunc$_{01}$		0.077	0.762	0.462	0.681	0.496	0.762 | END
Real$_1$	0.452	0.040	0.533	0.364	0.627	0.391	0.598 | FT
Average			**0.477**				**0.746**

The best values are taken from Table 2 and displayed here for the sake of a more convenient viewing.

It can be seen from Table 4 that the value of F1-measure averaged over four classifiers for each LF is significantly lower than the best result for this LF with the only exception of Oper$_1$ for which the difference between the average result and the best result is not so great (0.760 vs. 0.873).

It is rather surprising that Naive Bayes performed so poorly except for Oper$_1$ (F1-measure of 0.711). Naive Bayes was not able to predict Oper$_2$, IncepOper$_1$, ContOper$_1$, and Func$_0$ at all. However, [23] reports better results achieved by Naive Bayes. Unfortunately, they cannot be compared for all LFs in our experiments because not all of them were dealt with in [23]. Better results for Naive Bayes may be explained by a more informative data representation in [23] where Basic Concepts and Top Concepts of EuroWordNet were used as features for LF learning [21] together with hypernym information.

5 Conclusions

It is widely acknowledged that collocations can be classified according to their structure. *Oxford Collocations Dictionary for Students of English* [12] groups collocations with similar syntactic patterns like ‘verb + noun’, ‘verb + adverb’, ‘preposition + noun’, etc. We have shown that collocations can be classified semantically according to the typology of lexical functions. The formalism of lexical functions encodes semantic information of collocations representing it by abstract meanings like ‘do’, ‘cause’, ‘begin’, ‘continue’, etc. Lexical functions can serve as semantic descriptions of collocational classes. Still a more detailed description of classes is given by means of semantic and syntactic patterns typical for each class, like 'to do the <noun>', 'to cause the <noun> to exist', 'to begin to do the <noun>', etc.

It has been demonstrated in our experiments that verbal lexical functions can be detected automatically applying supervised machine learning techniques on the dataset of Spanish verb-noun collocations. We tested 68 supervised learning algorithms and evaluated their performance on automatic detection of lexical functions. The maximum F1-measure of 75% was achieved in our experiments. The best state-of-the-art result is 70% [22, 23].

Our experiments have also demonstrated that symbolic learning methods significantly outperform statistical methods on the task of LF detection. Such a classical statistic method as Naive Bayes failed to distinguish four out of eight LFs in the experiments; the average F1-measure of statistical methods is 0.477.

In future, we plan to apply the concept of syntactic n-grams [16, 17] for extraction of LF, i.e., to use additional syntactic information for this task.

Acknowledgements

We are grateful to Adam Kilgarriff and Vojtěch Kovář for providing us a list of most frequent verb-noun pairs from the Spanish Web Corpus of the Sketch Engine[2].

The work was done under partial support of Mexican Government: SNI, COFAA-IPN, BEIFI-IPN, and SIP-IPN grants 20201948 and 20200859.

References

1. **Alonso-Ramos, M., Rambow, O., & Wanner, L. (2008).** Using semantically annotated corpora to build collocation resources. *Proceedings of the International Language Resources and Evaluation Conference (LREC)*, Vol. 19, No. 4, pp. 1154–1158.
2. **Archer, V. (2007).** Using conceptual vectors to get Magn collocations (and using contrastive properties to get their translations). **Gerdes, K., Reuther, T. & Wanner, L. (eds.),** *Proceedings of the Third International Conference on Meaning-Text Theory,* Wiener Slawistischer Almanach, Sonderband 69, München-Wien, pp. 57–65.
3. **Bolshakov, I.A. & Gelbukh, A. (2001).** A very large database of collocations and semantic links. *Proceedings of NLDB'2000: 5th International Conference on Applications of Natural Language to Information Systems,* pp. 103–114. DOI: 10.1007/3-540-45399-7_9.
4. **Castro-Sánchez, N.A. & Sidorov, G.** (2010). Analysis of definitions of verbs in an explanatory dictionary for automatic extraction of actants based on detection of patterns. *Lecture Notes in Computer Science* 6177, pp 233–239.
5. **Delač D., Krleža, Z., Šnajder, J., Bašić B.D., & Šarić, F. (2009).** TermeX: A tool for collocation extraction. *Computational Linguistics and Intelligent Text Processing*, pp. 149–157. DOI: 10.1007/978-3-642-00382-0_12.
6. **Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten I. H. (2009).** The WEKA data mining software: An update. *SIGKDD Explorations*, Vol. 11, No. 11, pp. 10–18. DOI: 10.1145/1656274.1656278.
7. **Kahane, S. (2003).** The Meaning-Text theory, dependency and valency. *Handbooks of Linguistics and Communication Sciences*, Vol. 25, pp. 1–2.
8. **Kilgarriff, A., Rychly, P., Smrz, P., & Tugwell, D. (2004).** The sketch engine. *Proceedings of EURALEX,* Lorient, pp. 105–116.
9. **Longman Dictionary of Contemporary English (1995)**. Essex, UK: Longman Group Ltd.
10. **Manning, C. & Schütze, H. (1999).** *Foundations of statistical natural language processing.* The MIT Press, Cambridge, US.
11. **Mel'čuk, I. (1996).** Lexical functions: A tool for the description of lexical relations in a lexicon. *Lexical functions in lexicography and natural language processing. John Benjamins, Amsterdam/ Philadelphia,* pp. 37–102.
12. **Oxford Collocations Dictionary for Students of English (2003).** Oxford University Press, Oxford, UK.
13. **Ruppenhofer, J., Ellsworth, M., Petruck, M.R., Johnson, C.R., & Scheffczyk, J. (2006).** FrameNet II: Extended theory and practice. pp. 1–115.
14. **Schwab, D., Lafourcade, M., & Prince, V. (2002).** Antonymy and Conceptual Vectors. **Shu-Chuan, T., Tsuei-Er, C., & Yi-Fen, L. (eds.)** *Coling 2002: Proceedings of the 19th International Conference on Computational Linguistics*, Howard International House, Taipei, Taiwan, pp. 904–910.

[2] www.sketchengine.co.uk

15. **Seretan, V. (2011).** Survey of extraction methods. *Text, Speech and Language Technology*, Vol. 44, pp. 29–58.
16. **Sidorov, G. (2019).** *Syntactic n-grams in computational linguistics*. Springer.
17. **Sidorov, G. (2013).** *Construcción no lineal de n-gramas en lingüística computacional*. SMIA, México.
18. **Siepmann, D. (2005).** Collocation, colligation and encoding dictionaries. Part I: Lexicological aspects. *International Journal of Lexicography*, Vol. 18, No. 4, pp. 409–443.
19. **Smadja, F. (1993).** Retrieving collocations from text: Xtract. *Computational Linguistics*, Vol. 19, No. 1, pp. 143–178.
20. **Vossen, P. (1998).** EuroWordNet: A multilingual database with lexical semantic networks. *Kluwer Academic Publishers, Dordrecht, the Netherlands.* Vol. 32, No. 1-2, pp. 1–79.
21. **Vossen, P., Bloksma, L., Rodriguez, H., Climent, S., Roventini, A., Bertagna, F., & Alonge, A. (1998).** *The EuroWordNet base concepts and top ontology.* Technical Report Deliverable D017, D034, D036, WP5 EuroWordNet, LE2-4003. University of Amsterdam.
22. **Wanner, L. (2004).** Towards automatic fine-grained classification of verb-noun collocations. *Natural Language Engineering*, Vol. 10, No. 2, pp. 95–143. DOI: 10.1017/S1351324904003328.
23. **Wanner, L., Bohnet, B., & Giereth, M. (2006).** What is beyond collocations? Insights from machine learning experiments. *Proceedings of the EURALEX Conference*.
24. **Witten, I.H. & Frank, E. (2005).** *Data mining: Practical machine learning tools and techniques.* Morgan Kaufmann.

Article received on 29/01/2019; accepted on 02/09/2019.
Corresponding author is Alexander Gelbukh.