# Syntax-based identification of light-verb constructions

**Silvio Ricardo Cordeiro**
LLF, Université Paris Diderot
Paris, France
silvioricardoc@gmail.com

**Marie Candito**
LLF, Université Paris Diderot
Paris, France
marie.candito@linguist.univ-paris-diderot.fr

## Abstract

This paper analyzes results on light-verb construction identification, distinguishing between known cases that could be directly learned from training data from unknown cases that require an extra level of semantic processing. We propose a simple baseline that beats the best results of the PARSEME 1.1 shared task (Savary et al., 2018) for the known cases, and couple it with another simple baseline to handle the unknown cases. We additionally present two other classifiers based on a richer set of features, with results surpassing these best results by 7 percentage points.

## 1 Introduction

Light-verb constructions (LVCs), such as the expression *pay visit*, are a linguistic phenomenon coupling a verb and a stative or eventive noun, in which the verb itself is only needed for morphosyntactic purposes, its syntactic dependents being semantically related to the noun. For instance in the sentence *John paid me a visit*, the subject and object of *paid* play the roles of the visitor and the visited. The verb's semantics is either bleached or redundant with that of the noun (as in *commit crime*) (Savary et al., 2018).

This mismatch between syntax and semantics has to be taken care of for semantically-oriented tasks to recover the full predicate-argument structure of the noun, since at least one of its semantic arguments of the noun is generally attached to the verb in plain syntactic treebanks.[1] Moreover, the fact that the verb choice is conventionalized and semantically bleached makes LVC identification an important requirement in semantic tasks such as machine translation (Cap et al., 2015).

Because of their syntactico-semantic characteristics, LVCs are generally considered difficult to circumscribe and annotate consistently (Bonial and Palmer, 2016). Yet recently, the PARSEME 2018 shared-task has brought forth a collection of corpora containing verbal multiword expression (VMWE) annotations across 19 languages (Ramisch et al., 2018), including LVCs. The reported inter-annotator agreement is variable across languages, but the macro-averaged chance-corrected kappa is overall 0.69, which is generally considered to denote a good agreement. In the annotated corpora, the category of LVCs[2] accounted for a third of all expressions (Savary et al., 2018). The annotation was performed in a separate layer, largely independent from the underlying syntactic framework, and relied on semantic properties of the verb (bleached or redundant in the given context) and both semantic and syntactic properties of the noun: it should be stative or eventive and take at least one semantic argument, and it should be possible to have all the syntactic arguments of the verb realized within a NP headed by the noun (for instance out of *John paid me a visit*, one can create the NP *John's visit to me*).

A total of 13 systems participated in the PARSEME shared-task, predicting VMWEs occurrence in the test corpora. Results for each system varied across different systems and target languages, in which expressions that had been seen in the test corpus were predicted with variable accuracy. However, expressions that had never been seen in the test corpus were hardly ever predicted by most systems (the best F-score on unseen-

[1] For instance, Nivre and Vincze (2015) report that for the majority of the 18 UD languages at that time, in a structure like *X takes a photo of Y* in English, X is attached to the verb, but the Y argument is attached to the noun. In some annotation schemes, the Y would be attached to the verb too. Note though that some treebanks do annotate the LVC status (e.g. in Hungarian). Additional semantic annotation of LVC can be found e.g. in propbank (Bonial and Palmer, 2016).

[2] Unless otherwise stated, this paper refers to the 1.1 edition of the PARSEME shared task, and to the category LVC.full (as opposed to LVC.cause).



*Proceedings of the 22nd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 97–104
Turku, Finland, 30 September – 2 October, 

in-train expressions in the closed track is below 20%).

In this paper, we investigate the task of LVC identification in running text. The main contributions of this paper are: (1) we propose handling the task of LVC identification differently depending on whether it was seen in the training corpus; (2) we present a simple baseline that surpasses all systems for seen LVCs; (3) we propose and evaluate different techniques for the prediction of unseen LVCs, which we then compare to the state of the art.

The remainder of this paper is structured as follows: Section 2 presents the related work; Section 3 describes the methodology that will be employed; Section 4 describes the results; and finally, Section 5 presents our conclusions.

## 2 Related Work

LVC identification may follow one of two strategies: (a) LVC candidates are initially proposed based on lexicosyntactic patterns, and are then classified as LVC or non-LVC based on other criteria; (b) a variant of the BIO scheme (Ramshaw and Marcus, 1999) is employed so as to directly classify each token as belonging or not to an LVC. The former method allows the use of features that encompass the LVC as a whole, while the latter can be more easily implemented in the framework of some machine learning algorithms.

Most works in the literature concerning LVC identification focus on annotations in a particular language, often with a language-specific understanding of LVCs. Vincze et al. (2013) adapt a dependency parser so as to identify Hungarian LVC candidates as a byproduct of parsing, which they then evaluate on the Szeged Dependency Treebank with LVC annotations. Nagy T. et al. (2013) extract English LVC candidates involving a verb and a dependent noun with a specific dependency label. A J48 and an SVM classifier are then considered, using lexical and morphosyntactic features from the corpus, as well as semantic features from WordNet. The latter was found to contribute to better results when compared to earlier works that relied purely on morphosyntactic and statistical features (Tu and Roth, 2011). Chen et al. (2015) detect English LVCs in the BNC and OntoNotes corpora, using the PropBank layer to select LVC candidates composed of an eventive noun linked one of 6 known light verbs. The candidates are then filtered based on semantic features, including WordNet synsets and hypernym relations.

More recently, the PARSEME shared-task saw 13 system submissions that tried to predict LVCs along with other VMWEs for annotated corpora in 19 languages (Ramisch et al., 2018). Overall, the best $F_1$ scores across all languages in the open track were obtained by the SHOMA system, which employed a pipeline of CNNs, a Bi-LSTM, and and optional CRF layer (Taslimipoor and Rohanian, 2018). MWE prediction followed a variant of the BIO scheme that allowed multiple tags per token, with input features including a set of pretrained embeddings (leading the system to compete in the open track category), POS tags, and a set of word-shape features.

In the closed track, the TRAVERSAL system obtained the best overall results for MWEs in general as well as for LVCs. It uses a syntax-based approach, in which each node in the syntax tree was classified as part of an MWE or not (Waszczuk, 2018). The classifier resembles a second-order CRF, but rather than considering the previous 2 tokens at each point, it considers the parent and left-sibling. Features included the lemma, POS tag and dependency relation.

Rather than predicting each token as being part of an LVC or not, the varIDE system use a Naive Bayes classifier to tag LVC candidates (Pasquer et al., 2018). These were extracted based on all possible token combinations whole multi-set of lemmas corresponded to an LVC that had been seen in the training corpus (no attempts were made at predicting unseen LVCs). Classifier features included POS tags and morphological information.

Graph convolutional neural networks have also been used in the identification of VMWE candidates for subsequent classification (Rohanian et al., 2019). In this work, the network is combined with an attention mechanism so as to improve the accuracy of long-range predictions, and a Bi-LSTM layer is used to classify these predictions and produce the final output. The system uses contextualized embeddings (ELMo) and outperforms the state of the art for the four languages for which results are reported.

## 3 Methods and materials

Our LVC identification technique consists of two main stages: (1) extraction of LVC candidates based on syntactic patterns; and (2) classification

of candidates based on a set of lexical, morphological, syntactic and semantic features, concerning both the candidate as a whole and its components.

### 3.1 Extraction of candidates

The first step of LVC identification in a target corpus is to identify candidates. While LVCs are commonly thought of as a combination of a verb and noun acting as its direct object, other configurations can be attested in the PARSEME corpora. This may be due to morphosyntactic variations (e.g. passive voice), the presence of more complex noun phrases (instead of a single noun), non-standard analyses (e.g. verbs that are tagged as adjectives) or other language-specific idiosyncrasies. A robust candidate extraction method should handle this variation, and we do so by using the morphosyntactic patterns of the LVCs in the training corpora, using the provided UD parses.

So we start by extracting language-specific morphosyntactic patterns from the training LVCs. More specifically, for each LVC annotated in a training corpus, we retain a representation involving the POS tag and the syntactic relation between components (henceforth referred to as a "pattern"). If the LVC does not form a connected tree (e.g. *to give a series of lectures*), the pattern will additionally include the minimum number of nodes that makes the tree connected (if two nodes are only connected by the root node, we discard the occurrence instead). In the example above, the extracted pattern would be: $\texttt{VERB}_1 \xrightarrow{\texttt{obj}} (\texttt{NOUN}_2) \xrightarrow{\texttt{nmod}} \texttt{NOUN}_3$ (the components of the LVC being those not within brackets).

The number of extracted patterns ranges from 14 (Slovene) to 185 (Farsi), with an average of 90 patterns per language. We then sort the patterns based on how many occurrences of LVCs led to each pattern. As expected, the patterns follow a Zipfian distribution. For example, for the French training data, the most common pattern is $\texttt{VERB}_1 \xrightarrow{\texttt{obj}} \texttt{NOUN}_2$ with 977 occurrences; the second is $\texttt{NOUN}_1 \xrightarrow{\texttt{acl}} \texttt{VERB}_2$ with 150 occurrences (as in for instance *a picture taken yesterday*); the third is $\texttt{VERB}_1 \xrightarrow{\texttt{nsubj:pass}} \texttt{NOUN}_2$ with 58 occurrences (as in *this picture was taken yesterday*); and so on[3].

The most common patterns are then used to identify LVC candidates in the train, development and test data, using the Grew tool (Bonfante et al., 2018). Obviously, using unlexicalized patterns results in getting a vast majority of candidates that are not LVCs, and this is even more true for rare patterns. We experimented with two pattern selecting stategies: topN, in which we take the $N$ most common patterns (we considered values of $N \in \{1, 5, 10, 20, 50\}$ ); and atleastNoccurs, in which we take all patterns that originated from at least N occurrences in the training corpus (we considered $N \in \{2, 5, 10, 50\}$). Moreover, for each pattern $p$ containing $\xrightarrow{\texttt{label}} \texttt{NOUN}_i$, we add a pattern $p'$ replacing this subpattern by $\xrightarrow{\texttt{label}} (\texttt{NOUN}_j) \xrightarrow{\texttt{conj}} \texttt{NOUN}_i$[4].

Using the selected patterns, we identify LVC candidates in the development and test corpora, but also in the training corpora, so as to obtain positive and negative LVC candidates to train a binary classifier. For each identified candidate, we produce a set of features which may be related either to the whole LVC, or to its components.[5]

### 3.2 Features

The PARSEME 1.1 data contains test/development and training data for 19 languages. The training data contained an average of 1171 LVCs per language ($\sigma$=948, ranging from 78 for English to 2952 for Turkish). Most corpora contain morphosyntactic information (in most cases obtained by an external parser, and in most cases representing data using the POS and dependency tagsets recommended by UD).

For a given candidate $c$, we first extract the verb component $v$ and predicative noun component $n$. This is in general trivial, but in order to cover all cases, $v$ is taken to be the leftmost token that has POS tag VERB, or the leftmost AUX, or the leftmost ADJ, or the leftmost token in $c$, while $n$ is the leftmost NOUN, leftmost PROPN, or leftmost token that is not $v$. In all the features, we use the lemmas of $v$ and $n$. We then extract the following features:

- $F_1$: One-hot representing the pattern used to predict the candidate (see Section 3.1).
- $F_2$: Fraction of true LVCs among all candi-

[3]Note that the majority of LVCs has two components only, but some do contain additional components, such as prepositions when they are required to connect the verb and the noun.

[4]This alternative pattern would cover the expression *make adjustment* in *make an effort and an adjustment*, for which two occurrences of LVCs would be annotated according the PARSEME guidelines.

[5]For the training data, we take the union of gold LVCs and LVC candidates identified through syntactic patterns, since the patterns do not cover all gold LVCs.

| | BG | DE | EL | ES | EU | FA | FR | HE | HR | HU | IT | PL | PT | RO | SL | TR | $\mu$Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| %seen | 60 | 26 | 50 | 48 | 86 | 61 | 68 | 45 | 29 | 75 | 71 | 66 | 74 | 90 | 57 | 44 | 62 |
| Coverage (seen) | 98 | 100 | 100 | 95 | 94 | 91 | 99 | 82 | 93 | 86 | 90 | 97 | 95 | 96 | 100 | 98 | 95 |
| Coverage (unseen) | 73 | 84 | 91 | 98 | 98 | 90 | 91 | 78 | 96 | 86 | 72 | 90 | 93 | 33 | 92 | 95 | 90 |

Table 1: Fraction of LVC annotations that were *seen* in train, and LVC candidate coverage (highest recall achievable, if all candidates are predicted as LVC) — evaluated on the development sets.

dates in train that have the same pattern and lexical items (lemma-wise comparison) as $c$ (-1 if unseen in train).

- $F_3$: POS tag of $v$ and $n$.
- $F_4$: Dependency relation between $v$ and $n$ (NONE if not directly connected).
- $F_5$: One-hot for the number of components of $c$ (with the rationale that LVCs of length higher than 2 may display more non-standard behavior due to the additionally lexicalized words).
- $F_6$: One-hot for the number of gaps (extraneous words that do not belong to the LVC), between the leftmost and rightmost components of $c$, in the underlying sentence.
- $F_C$: Binary contextual features from the underlying UD parses. Features are defined for every observed ⟨key, value⟩ pair in the morphological CoNLLU column (e.g. ⟨Tense, Past⟩), as well as every observed ⟨column, value⟩ pair for the UD columns FORM, LEMMA, XPOS, UPOS and DEPREL (e.g. ⟨FORM, took⟩, ⟨LEMMA, picture⟩, ⟨POS, NOUN⟩). These features are binary in value, and indicate whether the ⟨key, value⟩ pair is present for $c$. A feature is considered present if it appears in at least one of the direct dependents of $n$ or $v$. We consider only the top $t$ features with the highest mutual information and whose underlying pairs appear in at least $\ell$ LVCs.

While it is clear that LVC identification would greatly benefit from fine-grained semantic clues such as noun predicativeness, such information is not readily available for most languages under study. We consider instead on a set of unsupervised features that can be constructed for all languages based on distributional semantic models. In particular, we consider the *fasttext* (Bojanowski et al., 2017) set of pretrained word embeddings (which is also used by the SHOMA system) as a basis for semantic features.

- $F_E$: Word embeddings for the lemma of the verb and noun (300 dimensions each).
- $F_k^1$ : k-nearest neighbors of the underlying noun $n$. Considered neighbors are all nouns that are paired up with the underlying verb $v$ in at least one LVC candidate in the training set, whether true LVC or not. We select the top $k$ neighbors whose embedding has highest cosine against $n$'s embedding. Each neighbor is either *seen-in-LVC* (it is part of at least one true LVC) or an *unseen-in-LVC* (it is part of false positives only). The final value of the feature is the sum of the scores of the $k$ neighbors, where a seen-in-LVC neighbor has score $+1$ and an unseen-in-LVC neighbor has score $-1$.
- $F_k^c$: Same as $F_k^1$, but each neighbor's score that is being summed up is additionally weighted by the underlying cosine.

### 3.3 LVC classifiers

We present below two LVC candidate binary classifiers based on the features above: SVM and FFN. We compare them against two simple baselines: Majority, which only predicts LVCs seen in train, and kNN, which we use either for all LVCs or for those unseen in train, in combination with Majority for the seen LVCs. Note we consider a predicted or gold LVC to be seen in train when the training corpus contains at least one gold LVC with same lemmas, in whatever order and with whatever syntactic pattern.

- Majority baseline: Predict a candidate as LVC if and only if it has been annotated more often than not in the training corpus (i.e. the value of feature $F_2$ is greater than $0.5$).
- kNN baseline: Predict a candidate as LVC if and only if the value of feature $F_k^c$ is positive,

| Configuration | BG | DE | EL | ES | EU | FA | FR | HE | HR | HU | IT | PL | PT | RO | SL | TR | $\mu$Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Maj (seen) | 72 | 84 | 76 | 89 | 81 | 83 | 93 | 68 | 91 | 91 | 87 | 92 | 86 | 90 | 70 | 38 | 81 |
| kNN (seen) | 62 | 84 | 76 | 89 | 79 | 81 | 92 | 66 | 91 | 91 | 83 | 89 | 82 | 37 | 60 | 46 | 78 |
| FFN (seen) | 74 | 78 | 82 | 94 | 87 | 87 | 94 | 67 | 92 | 92 | 86 | 90 | 88 | 87 | 68 | 71 | **85** |
| SVM (seen) | 70 | 74 | 84 | 86 | 86 | 89 | 93 | 64 | 91 | 91 | 84 | 89 | 90 | 72 | 76 | 57 | 84 |
| kNN (unseen) | 08 | 08 | 24 | 22 | 32 | 30 | 30 | 04 | 13 | 13 | 11 | 20 | 32 | 00 | 04 | 24 | 22 |
| FFN (unseen) | 15 | 18 | 27 | 29 | 22 | 55 | 31 | 05 | 21 | 33 | 12 | 22 | 25 | 00 | 08 | 27 | 28 |
| SVM (unseen) | 15 | 15 | 33 | 20 | 42 | 63 | 37 | 02 | 24 | 46 | 10 | 31 | 45 | 00 | 10 | 34 | **34** |

Table 2: $F_1$ scores on Majority and kNN baselines ($F^c_k$ with $k = 2$) , along with the best configuration for the SVM and FFN classifiers — on the development sets.

meaning that within the $k$ nominal neighbors of the noun $n$ of the candidate, the total cosine of seen-in-LVC surpasses that of the unseen-in-LVC neighbors.

- SVM: Support vector machine with RBF kernel. Positive and negative examples are balanced through compensating class weights. We use a 3-fold grid-search to select for the best combination of classifier hyperparameters for each language; we consider the values $C \in \{1, 10, 20, 50, 100\}$ and $\gamma \in \{0.5, 0.1, 0.05, 0.01\}$.
- FFN: Feed-forward network with a 100-neuron hidden layer, using tanh as an activation function and 50% dropout. The network uses an SGD optimizer[6] and negative log-likelihood loss. Positive training examples are duplicated as much as needed so as to be balanced against negative examples. The final list of examples is shuffled, and fed into the classifier in batches of size $B \in \{1, 2, 4, 8, 16\}$. Training is performed for a number $e$ of epochs, such that epoch $e + 1$ would have had higher loss on the validation set (10% of train). One-hot features are implemented as a layer of trainable embeddings instead (300 dimensions for lemmas; 5 dimensions for dependency relations, for $F_5$ and $F_6$).

### 3.4 Evaluation

We explore hyperparameters on the 16 languages that contained a development set, and evaluate the final systems on the test set for all 19 languages (using both training and development set for training). Evaluation of LVC predictions for each language uses the MWE-based $F_1$ score from of the PARSEME shared task (Ramisch et al., 2018). We modified its evaluation script so as to output scores for seen and unseen LVCs: it first labels a LVC (whether gold or predicted) as "seen" if there exists at least one gold LVC occurrence with the same set of lemmas in the training set, and unseen otherwise. The two labels are then evaluated separately.

We also present a micro-average score ($\mu$Avg), in which the $F_1$ scores of all languages are averaged with a weight that is proportional to the number of LVCs in that languages test (or development) set.[7] On test sets, we compare our results with SHOMA and TRAVERSAL, the two highest-scoring systems in the shared-task.[8]

## 4 Results

Table 1 presents the fraction of LVCs in the development set that can also be seen in the training set. In the lower end, German dev LVCs were only seen in train 26% of the time, mostly due to the small training set in this language. In the higher end, 90% of Romanian LVCs had a counterpart in the training set, suggesting that a simple baseline focusing on seen LVCs should already yield good results for this language.

The last two rows in Table 1 present the coverage (i.e. recall) in the initial step of LVC candidate extraction, for the strategy atleastNoccurs with $N = 2$. This strategy was found to yield the best results in both SVM and FFN settings during early experiments. It can be seen that, despite variation across languages, mainly due to training corpus size differences, the micro-averaged coverage is 95% for dev LVCs seen in train, and slightly

[6]Basic tuning of the learning rate led us to use 0.01.

[7]We chose to use micro-average, since the test sets across languages don't have the same number of sentences, for reasons that are independent of the linguistic properties of each.

[8]We used the predicted test sets of all participating systems (made available by the shared task organizers at `https://gitlab.com/parseme/sharedtask-data/tree/master/1.1/system-results`), filtering them to consider LVCs only. The best systems in open and closed tracks (SHOMA and TRAVERSAL) are the same when considering all verbal MWEs or LVCs only.

| Configuration | BG | DE | EL | EN | ES | EU | FA | FR | HE | HI | HR | HU | IT | LT | PL | PT | RO | SL | TR | $\mu$Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Maj (seen) | 74 | 67 | 85 | 64 | 60 | 87 | 84 | 85 | 70 | 90 | 87 | 94 | 82 | 48 | 87 | 89 | 81 | 75 | 53 | 80 |
| kNN (seen) | 71 | 67 | 78 | 64 | 61 | 83 | 84 | 87 | 64 | 89 | 89 | 92 | 80 | 47 | 88 | 87 | 38 | 62 | 58 | 78 |
| SVM (seen) | 75 | 67 | 80 | 65 | 60 | 86 | 87 | 89 | 66 | 94 | 80 | 94 | 83 | 38 | 89 | 91 | 67 | 79 | 77 | 81 |
| FFN (seen) | 82 | 69 | 79 | 67 | 64 | 87 | 86 | 89 | 69 | 92 | 85 | 94 | 82 | 48 | 88 | 93 | 74 | 69 | 87 | **83** |
| SHOMA (seen) | 64 | 00 | 79 | 00 | 39 | 88 | 88 | 66 | 73 | 88 | 43 | 78 | 66 | 49 | 69 | 87 | 93 | 52 | 79 | 76 |
| TRAV (seen) | 62 | 53 | 66 | 34 | 44 | 82 | 81 | 70 | 58 | 81 | 64 | 87 | 64 | 45 | 76 | 78 | 83 | 63 | 66 | 72 |
| kNN (unseen) | 14 | 09 | 23 | 23 | 17 | 16 | 21 | 34 | 05 | 44 | 18 | 24 | 13 | 08 | 19 | 28 | 00 | 05 | 36 | 22 |
| SVM (unseen) | 17 | 24 | 34 | 19 | 07 | 17 | 57 | 29 | 07 | 61 | 17 | 36 | 06 | 07 | 39 | 39 | 67 | 13 | 43 | **31** |
| FFN (unseen) | 20 | 18 | 18 | 33 | 20 | 19 | 45 | 25 | 06 | 64 | 16 | 29 | 12 | 15 | 33 | 31 | 00 | 07 | 34 | 29 |
| SHOMA (unseen) | 21 | 00 | 36 | 03 | 13 | 35 | 62 | 37 | 19 | 53 | 19 | 14 | 04 | 08 | 22 | 35 | 29 | 00 | 50 | **31** |
| TRAV (unseen) | 08 | 00 | 18 | 14 | 10 | 11 | 41 | 31 | 05 | 42 | 21 | 23 | 00 | 01 | 20 | 24 | 00 | 00 | 23 | 20 |
| Maj + kNN | 53 | 26 | 62 | 31 | 36 | 81 | 64 | 62 | 30 | 68 | 45 | 77 | 63 | 28 | 60 | 74 | 69 | 34 | 44 | 57 |
| kNN | 56 | 26 | 59 | 32 | 37 | 77 | 65 | 64 | 29 | 67 | 46 | 76 | 62 | 28 | 61 | 72 | 28 | 31 | 49 | 57 |
| SVM | 61 | 40 | 66 | 26 | 35 | 79 | 77 | 65 | 41 | 77 | 44 | 81 | 70 | 28 | 71 | 78 | 67 | 63 | 61 | **63** |
| FFN | 53 | 26 | 43 | 40 | 36 | 74 | 74 | 51 | 21 | 78 | 42 | 75 | 44 | 30 | 60 | 68 | 57 | 26 | 56 | 56 |
| SHOMA | 50 | 00 | 60 | 02 | 22 | 79 | 78 | 51 | 43 | 72 | 24 | 59 | 46 | 29 | 51 | 70 | 86 | 28 | 64 | 56 |
| TRAVERSAL | 44 | 15 | 47 | 18 | 26 | 70 | 65 | 52 | 30 | 62 | 32 | 68 | 51 | 23 | 52 | 62 | 73 | 38 | 44 | 50 |

Table 3: $F_1$ scores (split for seen LVCs, unseen LVCs and overall) for the Majority and kNN baselines, the best configuration of our SVM and FFN classifiers, and the highest-scoring systems in the shared-task (SHOMA and TRAV(ERSAL)) — evaluated on the test sets.

lower (90%) for unseen ones.

We tuned the hyperparameters on the development sets. For every system, the same configuration is used for all languages. The best kNN configuration is $F^c_k$ with k=2; the best SVM and FFN configurations are both $F_{1..6}$, $F_C$ ($t$=30, $\ell$=30), $F_E$.

Table 2 presents the scores obtained by these best configurations on the development sets. Across seen LVCs, both the Majority and kNN baselines have considerably high scores ($F_1$=81 and 78 respectively, but the highest results are obtained by FFN and SVM ($F_1$=85 and 84). For the unseen LVCs, results are quite lower, and there is a bigger gap between the kNN baseline ($F_1$=0.22) and the best system on unseen, namely the SVM ($F_1$=0.34).

Table 3 presents system results for the same configurations when evaluated against the test sets. On seen LVCs here again, the Majority baseline is slightly higher than the kNN baseline. However, both baselines beat the best systems from the shared-task (that we recomputed for LVCs only). Results for SVM ($F_1$=81) are comparable to the Majority baseline ($F_1$=81) while FFN obtains the highest score ($F_1$=83).

When we consider LVCs that were not seen in training data, results are much lower. The kNN baseline obtains an $F_1$=0.22, while SHOMA obtains $F_1$=0.31, as does our SVM, while results for FFN are slightly weaker. When predictions for both seen and unseen LVCs are taken together, FFN and SHOMA have comparable scores ($F_1$=56), while the baselines (either Majority+kNN or kNN alone) is slightly higher. The best system overall is the SVM ($F_1$=63).

## 5 Conclusion

In this paper, we considered the task of identifying LVCs in running text. We propose to use data-driven language-specific syntactic patterns for the extraction of LVC candidates out of syntactic parses, followed by a binary classification of the candidates into LVC or not.

We proposed a strong baseline combining different methods for LVC candidates depending on whether they were seen in the training set or not ("seen" meaning a LVC with same lemmas is annotated at least once in the training set). The baseline for seen cases tags a candidate as LVC if the training occurrences with same lemmas are more often tagged as LVC than not. The baseline for unseen cases uses the similarity of the predicative noun with the nouns of the training candidates, in a distributional semantic model. We also proposed supervised classifiers (a SVM and a feed-forward neural network) trained using internal and contextual morphosyntactic and semantic features, and working independently of the seen/unseen status.

Overall the SVM system is our best one, surpassing the best shared task system on LVCs (SHOMA, (Taslimipoor and Rohanian, 2018)) by 7 percentage points. When evaluating performance separately on seen and unseen LVCs, the feed-forward network performs a little better on seen LVCs, but less well on unseen ones. It

also appears that our results for seen LVCs surpass the best shared-task results even in the case of the baseline, in spite of a much simpler technique of supervised learning. For unseen LVCs, results are globally quite lower. The best performance is $F_1$=31%, achieved both by the SHOMA system and our SVM. Our kNN-inspired baseline achieves $F_1$=22% only, a performance that would rank second for unseen LVCs in the shared task.

Given the quality of predictions for seen LVCs, future works should focus on improving prediction for the unseen expressions. Such task could be achieved through an evaluation of different types of neural network. Other semantically-motivated language-independent features should also be considered, so as to estimate the candidate noun's abstractness and predicativeness, as well as the level of semantic bleaching in the use of the verb. Finally, future works should investigate using a model for contextualized word embeddings such as BERT (Devlin et al., 2018)), despite the difficulty of covering the 19 languages of the PARSEME datasets.

## Acknowledgements

We would like to thank Bruno Guillaume for advice in using Grew. This work was partially funded by the French National Research Agency (PARSEME-FR ANR-14-CERA-0001).

## References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Guillaume Bonfante, Bruno Guillaume, and Guy Perrier. 2018. *Application of Graph Rewriting to Natural Language Processing*. Wiley Online Library.

Claire Bonial and Martha Palmer. 2016. Comprehensive and consistent propbank light verb annotation. In *LREC*.

Fabienne Cap, Manju Nirmal, Marion Weller, and Sabine Schulte im Walde. 2015. How to account for idiomatic German support verb constructions in statistical machine translation. In *Proceedings of the 11th Workshop on Multiword Expressions*, pages 19–28, Denver, Colorado. Association for Computational Linguistics.

Wei-Te Chen, Claire Bonial, and Martha Palmer. 2015. English light verb construction identification using lexical knowledge. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2375–2381. AAAI Press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

István Nagy T., Veronika Vincze, and Richárd Farkas. 2013. Full-coverage identification of English light verb constructions. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 329–337, Nagoya, Japan. Asian Federation of Natural Language Processing.

Joakim Nivre and Veronika Vincze. 2015. Light verb constructions in universal dependencies. In *Poster at the 5th PARSEME meeting, Iasi, Romania*.

Caroline Pasquer, Carlos Ramisch, Agata Savary, and Jean-Yves Antoine. 2018. VarIDE at PARSEME shared task 2018: Are variants really as alike as two peas in a pod? In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Carlos Ramisch, Silvio Ricardo Cordeiro, Agata Savary, Veronika Vincze, Verginica Barbu Mititelu, Archna Bhatia, Maja Buljan, Marie Candito, Polona Gantar, Voula Giouli, Tunga Güngör, Abdelati Hawwari, Uxoa Iñurrieta, Jolanta Kovalevskaitė, Simon Krek, Timm Lichte, Chaya Liebeskind, Johanna Monti, Carla Parra Escartín, Behrang Qasemi-Zadeh, Renata Ramisch, Nathan Schneider, Ivelina Stoyanova, Ashwini Vaidya, and Abigail Walsh. 2018. Edition 1.1 of the PARSEME shared task on automatic identification of verbal multiword expressions. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 222–240, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Omid Rohanian, Shiva Taslimipoor, Samaneh Kouchaki, Le An Ha, and Ruslan Mitkov. 2019. Bridging the gap: Attending to discontinuity in identification of multiword expressions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2692–2698, Minneapolis, Minnesota. Association for Computational Linguistics.

Agata Savary, Marie Candito, Verginica Barbu Mititelu, Eduard Bejček, Fabienne Cap, Slavomír

Čéplö, Silvio Ricardo Cordeiro, Gülşen Cebiroğlu Eryiğit, Voula Giouli, Maarten Van Gompel, Yaakov HaCohen-Kerner, Jolanta Kovalevskaitė, Simon Krek, Chaya Liebeskind, Johanna Monti, Carla Parra Escartín, Lonneke Der, Behrang Qasemi Zadeh, Carlos Ramisch, and Veronika Vincze. 2018. *PARSEME multilingual corpus of verbal multiword expressions*. Berlin: Language Science Press.

Shiva Taslimipoor and Omid Rohanian. 2018. SHOMA at PARSEME Shared Task on automatic identification of VMWEs: Neural multiword expression tagging with high generalisation. *arXiv preprint arXiv:1809.03056*.

Yuancheng Tu and Dan Roth. 2011. Learning English light verb constructions: contextual or statistical. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 31–39. Association for Computational Linguistics.

Veronika Vincze, János Zsibrita, and István Nagy T. 2013. Dependency parsing for identifying Hungarian light verb constructions. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 207–215, Nagoya, Japan. Asian Federation of Natural Language Processing.

Jakub Waszczuk. 2018. TRAVERSAL at PARSEME Shared Task 2018: Identification of verbal multiword expressions using a discriminative tree-structured model. In *Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*.