ISSN 1392–124X (print), ISSN 2335–884X (online) INFORMATION TECHNOLOGY AND CONTROL, 2015, T. 44, Nr. 2

Action Classification in Action Ontology Building Using Robot-Specific Texts

Irena Markievicz[1], Jurgita Kapočiūtė-Dzikienė[1], Minija Tamošiūnaitė[2], Daiva Vitkutė-Adžgauskienė[1]

[1] *Vytautas Magnus University, Faculty of Informatics, Vileikos str. 8, LT- 44404 Kaunas, Lithuania e-mail: irena.markievicz@gmail.com*

[2] *The University of Göttingen, Bernstein Center for Computational Neuroscience, Friedrich-Hund Platz 1, 37077 Göttingen, Germany*

crossref http://dx.doi.org/10.5755/j01.itc.44.2.7322

Abstract. Instructions written in human-language cause no perception problems for humans, but become a challenge when translating them into robot executable format. This complex translation process covers different phases, including instruction completion by adding obligatory information that is not explicitly given in human-oriented instructions. Robot action ontology is a common source of such additional information, and it is normally structured around a limited number of verbs, denoting robot specific action categories, each of them characterized by a certain action environment. Semi-manual action ontology building procedures are normally based on domain-specific human-language text mining, and one of the problems to be solved is the assignment of action categories for the obtained verbs. Verbs in English language are very polysemous, therefore action category, referring to different robot capabilities, can be determined only after comprehensive analysis of the verb's context. The task we solve is formulated as the text classification task, where action categories are treated as classes, and appropriate verb context – as classification instances. Since all classes are clearly defined, supervised machine learning paradigm is the best selection to tackle this problem.

We experimentally investigated different context window widths; directions (context on the right, left, both sides of analyzed verb); and feature types (symbolic, lexical, morphological, aggregated). All statements were proved after exploration of two different datasets. The fact that all obtained results are above random and majority baselines allow us to claim that the proposed method can be used for predicting action categories. The best obtained results were achieved with Support Vector Machine method using window width of only 25 symbols on the right and bag-of-words as features. This exceeded random and majority baselines by more than 37% reaching 60% of accuracy.

Keywords: verb sense disambiguation, verb context classification, supervised machine learning.

1. Introduction

"*Start conveyor. Pick-up a rotor cap from conveyor and place it on fixture mounted on robot platform. Pick-up until fixture is full or conveyor is empty. Stop conveyor.*" – a factual piece of text extracted from the instruction sheet (or manual). It does not cause any perception problems for humans, but becomes a challenge when translating into a robot executable format. A robot able to "understand" and utilize human-language instructions has to be equipped with a number of different capabilities. Firstly, a vision function should help to recognize and locate relevant objects such as "conveyor", "rotor cap", "fixture" or "robot platform" in the environment; physical abilities –in particular, movements of hand, grabbing, lifting or putting– to perform different actions. Besides, robot has to perform reasoning: e.g. solve anaphora resolutions problems (by replacing pronouns with appropriate nouns where needed: e.g. "it" $\rightarrow$ "rotor cap"); transform complex actions into the chain of simple executable ones (e.g. "pick-up" $\rightarrow$ "hand move to object", "grab", "lift"); to understand how exactly each simple action has to be performed (because "pen", "cup", "balloon" or "rotor cap" should be grabbed with the different power and finger positions). Moreover, some actions require additional tools (e.g. "cut an apple" action requires a "knife" which is not directly mentioned in the instruction) or decoding of hidden meaning about their repetitiveness (e.g. "slice and apple", "dice an apple" are repetitive actions, but the number of repetitions is not known in advance). Last-mentioned ones can be performed only if have necessary links to world-knowledge in terms of ontologies and databases.

Indeed the process from robot "reading an instruction" to robot "executing an instruction" must pass many different phases and each of those phases requires deeper research analysis. In this research we focus on one of those phases, namely on instruction completion by adding action-specific information that is not explicitly given in human-oriented instructions. Automation of the robot instruction completion phase is often related to the use of specific action ontology, describing active environment for possible robot actions [20], [11]. A robot action is normally structured around a limited number of action categories, and each action verb that is included in the ontology has to be associated with an appropriate action category. Domain-specific texts Semi-manual action ontology building procedures are normally based on domain-specific human-language text mining, and one of the problems to be solved is the assignment of action categories for the obtained verbs. In this research we focus on determining the action category for action verbs, obtained in the process of domain-specific text mining. The problem is that verbs are highly polysemous in the English language, e.g. "hold on" in "hold on tightly" means "to grasp", but in "hold on a minute" – "to wait"; "mix up" in "mix up tenses" means "confuse", but in "mix up two kinds of nuts in a bowl" – "to blend", etc. Since our robot can perform actions only aware of their categories (5 action categories used in the research were: *handling action*, *hand-only action*, *null action*, *pick-and-place action*, *unrecognized action*), disambiguation problem can be considered as action categorization problem. Therefore, due to our previous examples "hold on" could be either attached to *pick-and-place* or *null action*; "mix up" - either to *unrecognized* or *handling action*. As we can notice from these examples the context surrounding each verb is crucial for determining its action category. Consequentially, from Computational Linguistics perspective our solving task can be formulated as text classification task where classes are action categories and classification instances – the context around the analyzed verb.

2. Related work

The task that we solve in our paper is very specific, formulated within the frame of the project ACAT[1]; therefore we could not find any research works tackling exactly the same problem as we do. However, the task that we attempting to solve in this paper can be discussed from two different perspectives: i.e. text classification (task of assigning predefined categories to unknown text documents) and verb sense disambiguation (task of assigning a sense to a verb given its context).

Both text classification and verb sense disambiguation tasks can be solved using two different approaches – i.e. rule-based and machine learning. Rule-based methods require human-experts for manual construction of appropriate patterns (i.e. rules) capable to take text classification or verb sense disambiguation decisions. Unfortunately rule-based methods not only require a lot of human work, but also are hardly adjustable to the new domains or applications. Due to this reason machine learning gained a lot of interest and remained the dominant paradigm till nowadays. Machine learning does not require manual construction of rules, because rules (defined as a model) are built automatically by observing and generalizing the characteristics of a given text.

Since all categories (action categories in our case) are known in advance; moreover, instances belonging to those categories will also be available, the area of possible machine learning methods can be narrowed down focusing on the supervised machine learning techniques only (for a review see [13]). One of the earliest comparative works on text classification using supervised machine learning methods revealed that Support Vector Machines (SVM) and k-Nearest Neighbor (k-NN) are top-notch classifiers, compared to Decision Trees (DT) or Naïve Bayes (NB) [8]. Dumais et al. [5] also demonstrated the superiority of SVM over DT and NB. Later SVM method was chosen by many researchers even without any consideration and became the most popular technique for classifying the texts not only for English, but for many other languages. Despite the domination of SVM, Bayesian methods maintained their popularity and are often selected as the baselines. It is necessary to mention that NB with a multinomial model (NBM) is more often selected instead of a simple NB with Bernoulli model, because NBM performs obviously better on the larger feature sets [14]. Moreover, some researchers report that NBM can even outperform popular SVM [16].

However not only the machine learning method itself, but the feature type is the most crucial problem in every text classification task. The most common approach for English remains simple bag-of-words (set of words) interpretation, which is also very accurate (e.g. demonstrated by Pang et al. [17]) that can even outperform other more sophisticated feature types based on token or character n-grams, lemmas, stems, etc. For the detailed description of text classification methods see [18], for the recent works [12]. Text classification was heavily researched on English in the past; current trends more focus on the morphologically complex or resource-scarce languages.

Verb sense disambiguation based on the supervised machine learning approaches is a very similar task to

[1] The goal of the ACAT project is to provide machines (robots) with this type of tacit information and to generate internal knowledge about individual task by way of creating and storing all required action information into so-called "Action Categories" (http://www.acat-project.eu/).

the text (topic, sentiment) classification; the main difference is in the interpretation of classification instances and in the used feature types. Instead of entire text, only snippets of it, –in particular, sentences, verb with its context, verb with its modifiers, etc. – are extracted and analyzed. Besides, instead of only lexical, morphological or symbolic features (which are typical in the text classification tasks), different types of semantic features are often exploited (e.g. generalized features using three types of word representations [9], Dynamic Dependency Neighbors, where for a given noun, verbs are retrieved taking that noun as an object form the dependency-parsed corpus [4], etc.). Besides, such feature types as bag-of-synsets (set of synonyms) based on external resources (e.g *WordNet* [6]), or left/right-word features, memorizing words position next to the verb in the text (e.g. [22]) are also often included.

Our task by its nature is a bit closer to text classification. Consequentially, this work will be our first attempt at finding a good method for classifying verbs by their context into action categories.

3. Action ontology in robotic

The concept of action is described in many fields of computer science – it can be introduced as a part of knowledge-based system, lexical (e.g. *WordNet, VerbNet*) or ontological (*OpenCyc, SUMO*) resource. It can be classified into lexical aspectual classes: activity, state, accomplishment and achievement [19]. Actions can be interpreted as functions, which transform one environment state into another environment state. Preconditions and effects restrict the change of initial and final states. While action changes a feature-value of an object and the value of a role or relation, the two general forms of action concept can be defined: *change-feature-value* and *change-role-value* [10]. In this case, we can define an action environment as a set of action objects in a certain state, which can be described by properties (e.g. size, weight, material, location in the certain reference system) and relation between action object (roles of objects).

Each robot has a limited number of actions it can execute. The action ontology should be based on those actions and it should add related actions and action environment information in the process of *ontology growing*. Our action ontology for robotic scenarios has a predefined structure (Fig. 1), which is based on an assumption that all robotic actions can be divided into several sequential steps. When presenting the sequence of these steps, the same action primitives are always used: *no-act, locate, grasp, pick up* and *put down*.

Fig. 1 presents the conceptual model of a text-based ontology used in our robotic experiments. The presented action ontology model assigns an appropriate *action synset* for each action as well as action details required for execution that can be obtained from textual sources. The *action synset* contains verbs, having the same sense.

Each action can be parameterized with modifiers, which can affect action execution speed (e.g. *wash slowly, wash quickly*), method (e.g. *wash carefully, wash safely*), repeating sequence (e.g. *wash once, wash twice, wash repeatedly, wash again*) or the sequence of steps (e.g. *wash first, wash then*). This information is included as action properties. Each action class has the following set of environment elements: objects, tools, time span, location, object materials, etc. (Fig. 1).

By choosing different robot action scenarios, we have described several action categories, which can be used in action ontology: *null-actions, homing actions, hand-only actions, handling actions, tools actions, tools actions with movement* and *pick and place actions*.

Null-action category contains actions suspending robot processes until some other robot or non-robot process completes. This action class is associated with the processes execution time, described by time units (e.g. *wait for 10 minutes*), by abstract definition of the process completion (e.g. *wait till centrifugation completes*) or the state of involved action objects (e.g. *wait until incubator is dry*). Null-actions always contain just one primitive *no-act.*

Homing class contains actions for moving the robot to a predefined home position. This class is relevant to the movement trajectory class, which describes movement of robot arm/hand or a whole robot.

Hand-only category includes actions that are executed with robot arm/hand and single target object (e.g. *press the button, open centrifuge*). The primitive steps of this category are: *locate target object, act at target.*

The difference between *hand-only* and *handling* actions is related with the action object position – if a *hand-only* action affects target object just by touching it, a *handling* action involves source object, which has to be taken by a robot hand (e.g. *shake the tube*) for some period of time. *Handling* action primitives are: *locate source object, pick up object, act, put down object.*

Actions with primary tools (e.g. *stir with a spoon*) are categorized into the *tool* actions category. The description of such actions contains common verb and tool objects (e.g. *wash with a sponge*) and specific tool-oriented verbs (e.g. *screw (with screwdriver), hammer (with hammer)*). *Tool* action primitives are: *locate tool, pick up tool, locate target object, act at target object, remove and put down tool.*

The action category Tools with movement includes actions with transmissions, gears or motors (e.g. mix with mixer). Primitives of this action class are: locate tool, pick up tool, locate target object, act on source with motor, remove and put down tools.

Handling and *tools (+ with movement)* actions are grouped into the *pick and place* action category. It

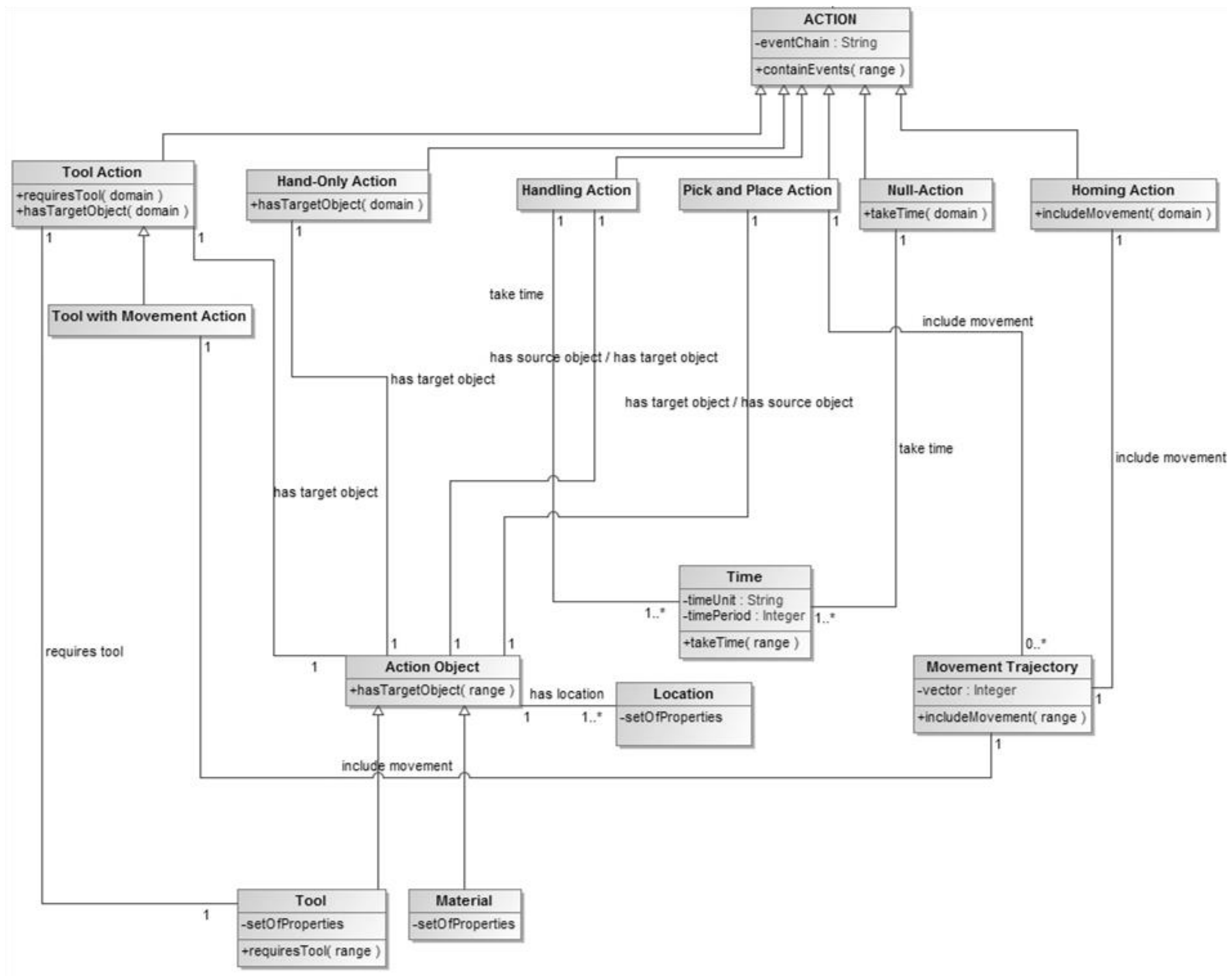

Figure 1. The conceptual model of an action ontology

Table 1. Action categories with the corresponding verbs (numbers in brackets indicate the frequencies of corresponding word appearance in the corpus)

Action category	Verbs (frequency of appearance in the corpus)
handling action	invert (88), mix (782), shake (244), stir (779)
hand-only action	close (265), open (403), press (112), pull (202), turn (866)
null action	stay (175), wait (371)
pick-and-place action	insert (372), pick (191), place (1,437), put (849)
unrecognized action	affect (1,021), afford (877), allow (2,098), appear (806), apply (796), associate (731), base (1,175), be (109,501), become (1,369), begin (710), bind (1,055), calculate (846), call (1,735), carry (780), cause (1,517), change (973), come (1,271), compare (932), consider (1,023), contain (3,179), continue (677), create (938), depend (872), describe (1,139), determine (1,966), develop (1,839), do (8,736), dry (701), explain (741), extract (688), find (3,234), follow (2,397), form (2,439), generate (709), get (2,182), give (3,632), go (1,334), grow (1,328), have (18,983), heat (715), help (1,346), hold (813), identify (1,271), improve (701), include (3,005), increase (1,597), indicate (841), involve (1,688), keep (1,140), know (2,927), lead (1,548), learn (881), leave (925), look (1,054), make (5,842), mean (706), measure (1,302), mix (782), need (2,271), observe (1,033), obtain (1,828), occur (1,503), perform (1,271), place (1,437), prepare (1,444), produce (2,736), provide (2,386), put (849), react (938), read (787), record (918), reduce (1,432), relate (743), remain (883), remove (1,715), report (886), require (2,033), result (1,114), run (907), say (688), see (2,656), set (910), show (2,730), start (1,321), study (805), take (2,720), test (855), think (917), treat (753), try (914), turn (866), understand (821), use (14,356), want (900), work (1,754), write (990)

contains variants of putting and pushing action objects together (e.g. *put down, insert, put together, put on top*). It is also relevant to the movement trajectory class. Action category primitives: *locate source object, pick up object, locate target object, put down source object at target.*

While defining *tools* actions and *tools actions with movement* can be easily done based on action object recognition, it is necessary to involve action classification algorithms for associating actions into the remaining predefined other action categories. In our experiments, we used only these actions categories, which do not include tools specification and cannot be recognized with particular tool-based patterns recognition. All relevant results describe these action categories: *handling actions, hand-only actions, null actions, pick and place actions* and *unrecognized actions* (see Table 1).

4. Dataset for action classification

A domain-specific corpus containing chemical laboratory texts was used for our experiments. The corpus texts describe chemistry laboratory experiments, basic rules, instruments and techniques. The texts that were crawled from the Internet, were subject to boilerplate removal (undesired content, such as HTML tags, scripts, styles advertisements, etc., removed) and filtering (corpus contains only texts, longer than 1600 symbols). The overall size of this experimental chemistry lab corpus (further referred to as the CHEMLAB corpus) is 3,821,073 running words. Collected texts were morphological annotated and lemmatized using Stanford University NLP tools for English language (http://nlp.stanford.edu/software/). The dataset for action classification experiment consists of 100 most frequent actions of the CHEMLAB corpus.

The fact that entire corpus contains 5,079 different lemmatized verbs where almost 36% (i.e. 1,825 words) appeared only once (such as *platinize* or *favorise*) led us to a decision not to analyze the verbs themselves, but better to zoom into their context. 111 most frequent verbs[2], which tend to be monosemous, were selected for the further experiments and were manually labeled with one of 5 action categories (see Table 1).

After labeling with the action categories, the pure context (not cleaned, not lemmatized) around each of these verbs (in Table 1) was extracted; the verb itself was eliminated, thus forming a new text document. The arbitrary selected primary window width for the context extraction was 200 symbols (including whitespaces) to both sides of the analyzed verb. The context window could be slightly increased so that the word at the left-most or right-most edge would be totally covered and included.

As we can see from Table 1, *unrecognized action* category is strongly dominated over the rest ones. To avoid the distortion of results, our dataset was balanced by randomly selecting 1,500 text documents for each action category (except for *null action* category, because only 546 cases were found in the entire corpus). See Table 2 for more detailed statistics about the created dataset.

Table 2. Number of text documents, tokens and distinct tokens in lowercase; average number of tokens in text document for each of the action categories in the created dataset

Action category	Documents	Tokens	Distinct tokens (in LC)	Avg. of tokens in doc.
handling	1,500	109,046	7,085	72.7
hand-only	1,500	106,850	9,663	71.2
null	546	38,581	5,362	70.7
pick-and-place	1,500	108,462	9,167	72.3
unrecognized	1,500	101,778	12,496	67.9
ALL	6,546	464,717	20,294	71.0

5. Methodology

5.1. Formal description of the task

The mathematical formulation of the verb context classification task, that we are attempting to solve, is given below.

Let $d \in D$ be a text document, belonging to a document space D. Each document contains verb context except verb itself, whose action category has to be determined.

Let C be a finite set of classes (action categories):

$$C=\{c_1, c_2, \ldots, c_N\}. \quad (1)$$

In our case $2 < N=5 << \infty$, thus we have multi-class classification problem.

Let D^L be a training set, containing instances I – i.e. document feature vectors d' (where d' corresponds to document d) with their appropriate class labels: $I=\langle d', c\rangle$. We selected only the verbs which tend to be monosemous, thus we have single-labeled instances only: i.e. the text document d cannot be attached to more than one class c.

Let function γ be a classification function mapping text documents to classes, $\gamma : D \rightarrow C$. Function γ determines the logic how d is labeled with c.

Let Γ denote a method which given training D^L as the input, can return a learned classification function γ' (defined as a model) as the output:

$$\Gamma(D^L) \rightarrow \gamma'. \quad (2)$$

5.2. Experimental setup

The main purpose of this research is to prove experimentally that verbs can be effectively classified into action categories by their context. The classification results will be considered reasonable, if achieved accuracy will outperform random ($\sum_i P(c_i)^2$) and majority ($\max(P(c_i))$) baselines ($P(c_i)$ is the

[2] Phrasal verbs were not considered.

Table 3. Feature type groups and feature types (with their description) used in our experiments

Feature group	Feature type	Description
Symbolic	Document-level character n-gram (*chrn*)	Succession of *n* characters including spaces and punctuation marks. We investigated sliding window of $n \in [3; 7]$. E.g. if *n*=5 phrase "verb context" would be split into "verb_", "erb_c", "rb_co", "b_con", "_cont", "conte", "ontex", "ntext".
Lexical	Bag-of-words (*bow*)	N-grams (interpolation of *n* from 1 up to 3) based on word tokens. E.g. if *n*=1 "verb context classification" would be transformed into single words "verb", "context", "classification"; if *n*=2 it would be transformed into singe words plus pairs of words "verb context", "context classification"; if *n*=3 it would be transformed into single, pairs of words plus triplets of words "verb context classification".
	Lemmas (*lem*)	N-grams (interpolation of *n* from 1 up to 3) based on word lemmas. All texts were lemmatized beforehand. Lemmatization transformed words into their main form, does not changing part-of-speech tag, e.g. "better" → "good", etc.
	Stems (*stem*)	N-grams (interpolation of *n* from 1 up to 3) based on word stems. All texts were stemmed beforehand. Stemming reduced inflected words to their stem: e.g. "friendly" → "friend", etc.
Morphological	Part-of-speech tags (*pos*)	N-grams (interpolation of *n* from 1 up to 3) based on part-of-speech tags. All texts were part-of-speech tagged beforehand. For part-of-speech tagging, as well as for lemmatization and stemming Stanford parser ([3]) was used.
Aggregated: Lexical + Morphological	Lemmas + part-of-speech tags (*lempos*) Stems + part-of-speech tags (*stempos*) Bag-of-words + part-of-speech tags (*bowpos*)	N-grams (interpolation of *n* from 1 up to 3) based on aggregated features which involved concatenated lexical and syntactic information. E.g. "filtration_NN" is an example of *lempos* feature, where "filtration" is a word lemma and "NN" is a part-of-speech tag for determining singular nouns.

probability of class c_i). For the dataset described in Section 3 calculated random baseline is 0.2170 and majority baseline is 0.2291.

Besides we wanted to determine how the results are affected by the different window widths and directions. Therefore we experimentally investigated context window widths of 15, 20, 25, 50, 100 and 200 symbols and directions of context to the both/left/right side(s) of the verb.

Since an appropriately selected feature type is the keystone of any text classification method, we investigated 26 different feature types (see the summary in Table 3) to find out which one gives the best results.

Before starting the experiments, we formulated two hypotheses.

Our first hypothesis states that a context on the right of the verb should be more informative, because, typically, in the instructions verbs go in the imperative mood, e.g. "do something", "do something with", etc. The laconic instruction language and rigid word-order in the English sentences allows us to expect reasonable narrow windows (covering 3-5 words) to be much more effective compared to the longer ones.

Our second hypothesis states that English language should benefit the most from the lexical features, but we still do not expect stemming and lemmatization to give much boost in accuracy: English language is not very highly inflective; moreover, inflections contain some extra information which also might be important.

5.3. Classification methods

We attempt to find a method Γ which could create the model γ' the best approximating γ (see Section 4.1, formula (2)). For this reason, two supervised machine learning approaches– in particular, Naïve Bayes Multinomial (NBM) [2] and Support Vector Machine (SVM) [1] – were selected and experimentally investigated.

NBM method is often selected for the text classification tasks mostly due its simplicity: Naïve Bayes assumption about the feature independence allows parameters of each feature to be learned separately. This method performs especially well when the number of features having equal significance is large; is very fast and does not require huge data storage resources. Besides, this Bayesian method is often selected as the baseline approach.

SVM was selected because it is the most popular technique for text classification, which can effectively cope with high dimensional feature spaces (e.g. 20,294 word features in our dataset (see Table 2)); sparseness of the feature vectors (e.g. among 20,294, each instance would have only ~3.1 non-zero word feature values); and instances do not sharing any common features (common for short texts, e.g. average length of instance in our dataset is ~71.0 words). Besides, SVM does not perform aggressive feature selection which may result in a loss of information.

In our experiments we used NBM and SMO for SVM implementations in WEKA ([7]) machine learning toolkit, version 3.6. All parameters were set to their default values, except for SVM: the parameters of SVM were tuned, giving the best results with the polynomial kernel.

6. Results

Our initial experiments involved investigation of different window widths (15, 20, 25, 50, 100, and 200, still including words on the edge) and directions (both/left/right) (see Fig. 2 for results with NBM and Fig. 3 for results with SVM) using the bag-of-words (*bow*) as the feature type. All experiments were performed with 10-fold cross-validation [21]). For the results evaluation we used accuracy and micro/macro F-score metrics.

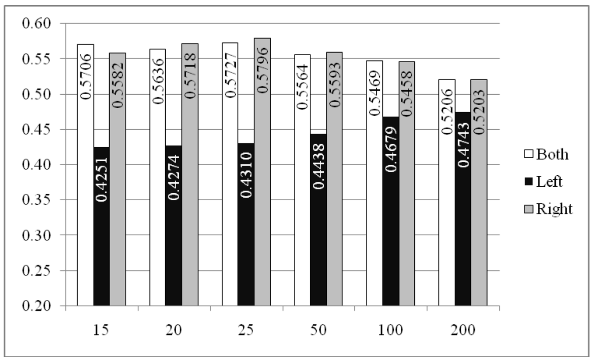

Figure 2. Results obtained with NBM method: X axis represents different window widths, Y axis – obtained accuracy. The best results where obtained with the right context of 25 symbols

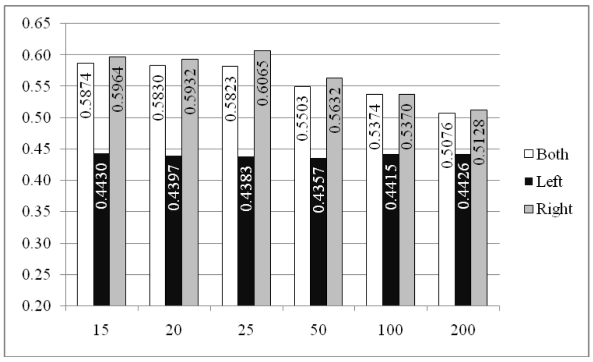

Figure 3. Results obtained with SVM method: X axis represents different window widths, Y axis – obtained accuracy. The best results where obtained with the right context of 25 symbols

To investigate the influence of the feature types, we kept window width and direction parameters stable. However, instead of investigating the best, we investigated three derived datasets: the best obtained with the context to the both sides (Both-15: window width = 15 and direction = both), with the context on the left (Left-200: window width = 200 and direction = left) and on the right (Right-25: window width = 25 and direction = right) contexts. For detailed statistics about derived datasets see Table 4. For the classification results with NBM see Table 5; with SVM see Table 6.

Table 4. Total number of instances, tokens, distinct tokens (in lowercase), lemmas, stems; and average number of tokens in instances in our derived datasets

Derived dataset	**Both-15**	**Left-200**	**Right-25**
Numb. of instances	6,546	6,546	6,546
Numb. of tokens	42,798	232,778	33,107
Numb. of distinct tokens (in LC)	6,509	15,25	5,669
Numb. of distinct lemmas	5,781	13,004	5,092
Numb. of distinct stems	5,491	11,942	4,841
Avg. numb. of tokens in inst.	5.6	35.6	5.1

7. Discussion

Due to the fact that all obtained results were above random and majority baselines, we can state that the proposed method is effective and can be used to determine action categories from the verb context.

In line with the dataset described in Section 3, we also built a very small one, containing only 500 instances for each action category (the dataset was composed using the same principles as the first one). We performed the same control experiments with this small dataset as with the larger one. It was done to ascertain if the findings generalize over both datasets, meaning that they are not accidental.

Thus, despite the size of the dataset, our first hypothesis was confirmed: i.e. the context on the right was the most informative and gave the biggest boost in accuracy compared to the context on the left or lying in both directions (see Fig. 1 and Fig. 2). The assumption that the best results should be obtained with a relatively small window was confirmed as well: the best results were obtained with 25 symbols (~5 words) only using bag-of-words as features. Besides, SVM gave much better results with smaller window widths and was capable to outperform NBM.

In our further experiments we investigated the effectiveness of the token feature types. Overall, the best *n* seems to be equal to 1, *n*=2 often slightly boosts the accuracy, but *n*=3 – usually degrades the performance. We performed McNemar test [15] to see if the differences are statistically significant. Despite the fact that differences are not statistically significant between different lexical feature types, bag-of-words can be considered as marginally the best feature type. Hence, our assumption that English language does not benefit from lemmatization or stemming was confirmed as well. Not surprisingly, part-of-speech tags give the overall worst classification results, and, correspondingly, do not help to solve disambiguation problems.

Table 5. Accuracy, micro and macro average F-scores for all derived datasets obtained with NBM. Number next to the feature type indicates *n* of n-gram. The best results with each evaluation metric are underlined

		Both-15			**Left-200**			**Right-25**		
Feature type group	**Feature type**	**Acc.**	**MicroF**	**MacroF**	**Acc.**	**MicroF**	**MacroF**	**Acc.**	**MicroF**	**MacroF**
Symbolic	chr3	0.5370	0.5370	0.5140	0.4607	0.4530	0.4316	0.5478	0.5480	0.5236
	chr4	0.5280	0.5280	0.5062	0.4577	0.4500	0.4286	0.5267	0.5280	0.5070
	chr5	0.5134	0.5130	0.4920	0.4458	0.4370	0.4150	0.5121	0.5120	0.4932
	chr6	0.4795	0.4760	0.4570	0.4323	0.4240	0.4020	0.4843	0.4830	0.4638
	chr7	0.4464	0.4410	0.4226	0.4248	0.4150	0.3892	0.4387	0.4340	0.4134
Lexical	bow	0.5706	0.5610	0.5186	0.4743	0.4640	0.4406	0.5796	0.5690	0.5286
	bow2	0.5701	0.5670	0.5372	0.4684	0.4560	0.4318	0.5657	0.5600	0.5270
	bow3	0.5686	0.5650	0.5352	0.4670	0.4540	0.4300	0.5661	0.5600	0.5256
	lem	0.5724	0.5650	0.5294	0.4760	0.4660	0.4448	0.5747	0.5650	0.5242
	lem2	0.5735	0.5700	0.5418	0.4681	0.4570	0.4342	0.5733	0.5690	0.5392
	lem3	0.5666	0.5630	0.5362	0.4653	0.4540	0.4320	0.5648	0.5610	0.5294
	stem	0.5755	0.5680	0.5328	0.4846	0.4750	0.4536	0.5823	0.5730	0.5314
	stem2	0.5695	0.5660	0.5400	0.4737	0.4630	0.4400	0.5680	0.5640	0.5346
	stem3	0.5697	0.5670	0.5394	0.4711	0.4610	0.4386	0.5695	0.5650	0.5340
Morpho-logical	pos	0.3488	0.3340	0.2912	0.3385	0.3140	0.2738	0.3837	0.3600	0.3142
	pos2	0.3989	0.3870	0.3438	0.3665	0.3520	0.3192	0.4193	0.4000	0.3530
	pos3	0.4148	0.4100	0.3794	0.3702	0.3620	0.3384	0.4276	0.4150	0.3792
Aggregated: Lexical + Morphological	lempos	0.5503	0.5420	0.5044	0.4630	0.4540	0.4322	0.5732	0.5620	0.5196
	lempos2	0.5516	0.5490	0.5190	0.4575	0.4470	0.4228	0.5548	0.5500	0.5176
	lempos3	0.5455	0.5420	0.5106	0.4534	0.4430	0.4190	0.5558	0.5510	0.5172
	stempos	0.5500	0.5410	0.5022	0.4639	0.4540	0.4330	0.5724	0.5610	0.5190
	stempos2	0.5443	0.5410	0.5124	0.4595	0.4490	0.4260	0.5559	0.5510	0.5182
	stempos3	0.5434	0.5400	0.5092	0.4546	0.4450	0.4214	0.5530	0.5480	0.5150
	bowpos	0.5536	0.5460	0.5076	0.4678	0.4580	0.4382	0.5687	0.5570	0.5154
	bowpos2	0.5432	0.5400	0.5106	0.4551	0.4450	0.4230	0.5519	0.5470	0.5136
	bowpos3	0.5428	0.5400	0.5088	0.4490	0.4390	0.4160	0.5501	0.5450	0.5116
	Random baseline	0.2170								
	Majority baseline	0.2291								

The context around e.g. "turned into" in "liquid turned into gold" and "robot turned into conveyor" has absolutely the same part-of-speech tags and it is a bit confusing. For this reason, lexical features used in concatenation with part-of-speech tags give slightly worse results compared with the lexical features used alone. Character n-grams represent the second worst feature type. All these findings allow us to conclude that words alone in the form as they appear in the text are the most informative for solving verb disambiguation problem from the text context.

8. Conclusions and future work

We have formulated and experimentally confirmed two hypotheses:

- The context on the right of the verb is the most informative for determining its action category. The most informative context is in a window of 25 symbols.
- The most accurate feature type is bag-of-words interpretation, thus lemmatization or stemming does not give obvious improvements in accuracy.

The best results are obtained with SVM method using bag-of-words as features. The results boost random and majority baselines more than 37% and achieve 60% of accuracy.

The obtained results are promising and motivate us to continue on trying to increase classification accuracy even more. These solutions could probably be found after comprehensive error analysis. Besides, we are planning to expand the set of verbs by including phrasal verbs and not binding to the monosemous words only. But this will definitely require manually annotated data.

Table 6. Accuracy, micro and macro average F-scores for all derived datasets obtained with SVM

		Both-15			**Left-200**			**Right-25**		
Feature type group	**Feature type**	**Acc.**	**MicroF**	**MacroF**	**Acc.**	**MicroF**	**MacroF**	**Acc.**	**MicroF**	**MacroF**
Symbolic	chr3	0.5371	0.5350	0.5010	0.4216	0.4180	0.3904	0.5548	0.5520	0.5190
	chr4	0.5254	0.5230	0.4932	0.4247	0.4210	0.3936	0.5445	0.5420	0.5102
	chr5	0.5168	0.5140	0.4852	0.4227	0.4180	0.3938	0.5394	0.5360	0.5072
	chr6	0.5058	0.5020	0.4738	0.4227	0.4180	0.3866	0.5180	0.5130	0.4808
	chr7	0.4659	0.4610	0.4376	0.4134	0.4080	0.3778	0.4924	0.4850	0.4580
Lexical	bow	0.5874	0.5840	0.5540	0.4426	0.4370	0.4070	0.6065	0.6020	0.5646
	bow2	0.5862	0.5830	0.5486	0.4522	0.4470	0.4204	0.5990	0.5950	0.5584
	bow3	0.5886	0.5860	0.5506	0.4497	0.4450	0.4188	0.5985	0.5950	0.5582
	lem	0.5784	0.5750	0.5430	0.4513	0.4450	0.4152	0.5920	0.5890	0.5522
	lem2	0.5840	0.5800	0.5440	0.4407	0.4360	0.4078	0.5991	0.5950	0.5580
	lem3	0.5807	0.5770	0.5422	0.4444	0.4400	0.4118	0.5943	0.5900	0.5544
	stem	0.5752	0.5720	0.5422	0.4606	0.4550	0.4248	0.5976	0.5950	0.5592
	stem2	0.5826	0.5790	0.5448	0.4496	0.4450	0.4170	0.5998	0.5960	0.5604
	stem3	0.5805	0.5770	0.5434	0.4546	0.4500	0.4202	0.5988	0.5950	0.5586
Morphological	pos	0.3734	0.3550	0.3098	0.3642	0.3440	0.3006	0.4102	0.3890	0.3392
	pos2	0.4305	0.4200	0.3810	0.3702	0.3630	0.3292	0.4545	0.4390	0.3952
	pos3	0.4290	0.4240	0.3918	0.3636	0.3590	0.3294	0.4543	0.4430	0.4026
Aggregated: Lexical + Morphological	lempos	0.5720	0.5680	0.5372	0.4384	0.4330	0.4044	0.5984	0.5940	0.5582
	lempos2	0.5697	0.5670	0.5322	0.4421	0.4360	0.4056	0.5877	0.5830	0.5458
	lempos3	0.5691	0.5650	0.5316	0.4424	0.4370	0.4078	0.5837	0.5790	0.5420
	stempos	0.5698	0.5660	0.5344	0.4352	0.4300	0.4002	0.5973	0.5930	0.5568
	stempos2	0.5735	0.5700	0.5364	0.4447	0.4390	0.4078	0.5836	0.5790	0.5426
	stempos3	0.5703	0.5660	0.5324	0.4482	0.4430	0.4128	0.5800	0.5750	0.5378
	bowpos	0.5720	0.5690	0.5366	0.4418	0.4360	0.4060	0.6037	0.5990	0.5616
	bowpos2	0.5684	0.5650	0.5310	0.4455	0.4400	0.4094	0.5836	0.5790	0.5436
	bowpos3	0.5691	0.5650	0.5306	0.4481	0.4430	0.4122	0.5796	0.5740	0.5370
	Random baseline	0.2170								
	Majority baseline	0.2291								

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Programme and Theme: ICT-2011.2.1, Cognitive Systems and Robotics) under grant agreement No. 600578, ACAT.

References

[1] **C. Cortes, V. Vapnik.** Support-vector networks. *Machine Learning*, 1995, Vol. 20, Issue 3, 273–297.

[2] **D. L. David, A. G. William.** A sequential algorithm for training text classifiers. In: Proceedings of Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR-94), 1994, pp. 3–12.

[3] **M. C. De Marnee, C. D. Manning.** *Stanford typed dependencies manual*. Stanford, USA, 2008.

[4] **D. Dligach, M. Palmer.** Novel Semantic Features for Verb Sense Disambiguation. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies (ACL'2008): Short Papers, 2008, pp. 29-32.

[5] **S. Dumais, J. Platt, D. Heckerman, M. Sahami.** Inductive learning algorithms and representations for text categorization. In: *Proceedings of the seventh international conference on Information and knowledge management*, Bethesda, Maryland, USA, 1998, pp. 148–155.

[6] **C. Fellbaum (edt.)** *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA, 1998.

[7] **M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten.** The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 2009, Vol. 11, No. 1, 10–18.

[8] **T. Joachims.** Text categorization with support vector machines: learning with many relevant features. In: *Proceedings of ECML-98, 10th European Conference on Machine Learning*, 1998, pp. 137–142.

[9] **D. Kawahara, M. Palmer.** Single Classifier Approach for Verb Sense Disambiguation based on Generalized

Features. In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 2014, pp. 4210–4213.

[10] **C. Kemke.** An action ontology framework for natural language interfaces to agent systems. *Artificial Intelligence Review, Springer*, 2007.

[11] **S. Kobayashi, S. Tamagawa, T. Morita, T. Yamaguchi.** Intelligent Humanoid Robot with Japanese Wikipedia Ontology and Robot Action Ontology. In: *Proceedings of the 6th International Conference on Human Robot Interaction, HRI 2011*, Lausanne, Switzerland, March 6-9, 2011, pp. 417-424.

[12] **V. Korde, C. N. Mahender.** Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications*, 2012, Vol. 3, No. 2, 85–99.

[13] **S. B. Kotsiantis.** Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, 2007, Vol. 31, 249–268.

[14] **A. McCallum, K. Nigam.** A comparison of event models for Naive Bayes text classification. In: *Proceedings of AAAI-98 workshop on learning for text categorization*, 1998, pp. 41–48.

[15] **Q. M. McNemar.** Note on the sampling error of the difference between correlated proportions or percenttages. *Psychometrika*, 1947, Vol. 12, Vol. 2, 153–157.

[16] **A. Pak, P. Paroubek.** Twitter for Sentiment Analysis: When Language Resources are Not Available. In: *Proceedings of Database and Expert Systems Applications* (DEXA 2011), 2011, pp. 111–115.

[17] **B. Pang, L. Lee, S. Vaithyanathan.** Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, (EMNLP), 2002, pp. 79–86.

[18] **F. Sebastiani.** Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 2002, Vol. 34, 1–47.

[19] **R. Trypuz, L. Vieu.** An ontology of the aspectual classes of actions. In: ILCLI International Workshop on Logic and Philosophy of Knowledge, Communication and Action (LogKCA-07), 2007, pp. 393-409.

[20] **R. C. Wales, V. L. Shalin, D. S. Bass.** An Ontology for Requesting Distant Robotic Action: A Case Study in Naming and Action Identification for Planning on the Mars Exploration Rover Mission. NASA Report No. AIAA Paper, 2004, 99-1247. Public domain license. Online source: http://wayback.archive-it.org/1792/20100419102035/http://hdl.handle.net/2060/20040086918.

[21] **S. M. Weiss, C. Kulikowski.** *Computer systems that learn*. San Francisco, California, USA, Morgan Kaufmann, 1991.

[22] **P. Ye, T. Baldwin.** Verb Sense Disambiguation Using Selectional Preferences Extracted with a State-of-the-art Semantic Role Labeler. In: *Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006)*, 2006, pp. 139–148.

Received June 2014.