Semi-automatic Annotation of Event Structure, Argument Structure, and Opposition Structure to WordNet by Using Event Structure Frame

Seohyun Im
Automation and System Research Institute, Seoul National University
Seoul, South Korea
Seohyunim71@gmail.com

Abstract

In this paper, we present semi-automatic annotation of the Event Structure Frames to synsets of English verbs in WordNet. The Event Structure Frame is a sub-eventual structure frame which combines event structure (lexical aspect) with argument structure represented by semantic roles and opposition structure which represents the presupposed and entailed sub-events of a matrix event. Our annotation work is done semi-automatically by GESL-based automatic annotation and manual error-correction. GESL is an automatic annotation tool of the Event Structure Frame to verbs in a sentence. We apply GESL to the example sentence given for each synset of a verb in WordNet. We expect that our work will make WordNet much more useful for any NLP and its applications which require lexical semantic information of English verbs.

1 Introduction

This paper aims to present our work of linking the Event Structure Frame (henceforth, ESF) to WordNet to improve its usability for NLP applications such as multimodal (and textual) inference tasks which require the lexical semantic information of words.

WordNet represents the distinct senses of verbs very delicately and organizes the semantic relations such as synonymy and hypernymy of the verbs. The semantic relation is one of the major strengths of WordNet. However, WordNet lacks the following two factors which consist of the lexical meaning of verbs. First, the lexical aspect of verbs, which is represented as event structure, is essential lexical semantic information (Pustejovsky, 1995). Different lexical aspects have different event structure frames. Secondly, argument structure with semantic roles also is a necessary factor to represent the meaning of verbs.

We argue in this paper that the ESF, originally developed by Im & Pustejovsky (2009, 2010) and Im (2013), enriches WordNet. Linking ESF to WordNet makes it possible to provide information about sub-eventual structure and argument structure of English verbs together with original information about the semantic relation of verbs WordNet gives.

The ESF of a verb with its specific sense divides its sub-events into pre-state, process, and post-state. This will be a big help to any kind of inferencing or reasoning tasks which use the word meaning of verbs. For instance, the ESF of the English verb *arrive* in (1) gives the information required to derive the lexically entailed result state after the arriving event and the presupposed state before it.

(1) The Event Structure Frame of *arrive*
(arrive.v.01)
se1: pre-state: not_be_at (theme, goal)
se2: process: arriving (theme)
se3: post-state: be_at (theme, goal)

Given the sentence *John arrived at school at 9 am today,* we get the inferred statements from the ESF of arrive.v.01 by Word Sense Disambiguation (linking *arrive* to an appropriate WordNet synset arrive.v.01): 'John was not at school before 9 am today' and 'John was at school after 9 am today'.

We began the WordNet-ESF linking project around the end of last year (2018). The tagging work goes through the two steps: automatic annotation of the ESF for each verb sysnset in

WordNet by GESL and manual error correction. GESL is an automatic annotation tool of the ESF for verbs in a sentence developed by Im (2013). Since WordNet synsets have their example sentences, GESL is applied to the sentences for automatic ESF annotation.

In this paper, we present our main idea regarding the task and small annotated data focused on English motion verbs. The structure of this paper is as follows: in the next section, we briefly introduce the theoretical background of the Event Structure Frame and show the list of pre-defined ESFs in Im (2013). Section 3 describes our main task. First, we introduce GESL, the automatic ESF annotating system to verbs in text. Second, we explain how to assign ESFs to WordNet synsets. In section 4, we explain ESF-based verb classification and the extended list of ESFs for WordNet-ESF linking. In section 5, we show small size of data in which we annotated ESFs to WordNet synsets for a part of motion verbs. After that, we mention FrameNet and VerbNet and explain why we chose linking ESF to WordNet in the next section. Finally, we summarize our main idea and future work in section 6.

2 Event Structure Frame

In this section, we explain the theoretical background of the ESF. The idea is originated from Im and Pustejovsky (2009, 2010) and fully developed in Im (2013). The ESF is based on event structure and argument structure in Generative Lexicon Theory (Pustejovsky, 1995) and opposition structure (Pustejovsky, 2000). As shown in (1), the ESF is a merger of event structure, argument structure, and opposition structure.

A complex event has its sub-eventual structure which consists of temporally ordered sub-events. In (1), se1 precedes se2 and se3. The event structure of a complex event is composed of pre-state, process, and post-state. Pre-state is a presupposed sub-event. That is, it is a presupposition of the verb which denotes the main process (event). For instance, our common sense requires the presupposition that Kennedy was alive before killing him in order to use the word *kill*. On the other hand, post-state is temporally later than the killing process. The post-state is a lexical entailment of the verb *kill*. When Osswald killed Kennedy, it normally entails that Kennedy died and Kennedy is dead.

To sum up, the combination of pre-state, process, and post-state is a temporally ordered structure of lexical presuppositions, main process, and lexical entailments.

Based on the theoretical viewpoint about ESF, Im (2013) suggests 23 pre-defined ESF-dependent verb classes. As shown in Table 1, verb classification in GESL consists of three steps of classification.

aspectual	semantic	event type
state	state	state
process	process	process
	motion	motion
transition	change-of-location	leave, arrive, pass, transfer
	change-of-possession	lose, get, give
	change-of-state	come-into-existence, go-out-of-existence, become, begin, continue, end positive-causation, negative-causation, cos-leave, cos-arrive, cos-transfer, scalar-change change-state

Table 1. Verb classification in Im (2013)

The first step is to classify verbs according to the lexical aspect of verbs - state, process, and transition, based on Generative Lexicon Theory. State and process are simple events and transition is a complex event. Therefore, transition verbs have sub-eventual structure with more than one sub-event.

The next step is semantic classification of verbs. Im (2013) classifies process verbs into two groups – process and motion. It is because motion verbs have their own special lexical semantic properties. Their lexical aspect is heavily dependent on their contextual meaning. For instance, the motion verb *run* belongs to motion process but it changes into change-of-location class when it co-occurs with the prepositional phrases which denote goal, source, duration, etc. (e.g. *run to the store, run from the store, run for 30 minutes*). Transition verbs are classified into change-of-location, change-of-possession, or change-of-state verbs semantically.

The last step is to divide each semantic class into more specific ESF-dependent classes. Each verb class we finally get has its own ESF. Specifically, the change-of-location verb class has arrive, leave, pass, and transfer classes. The change-of-possession verbs are classified into lose, get, or give. Change-of-state verbs in-

clude aspectual classes (begin, continue, end), positive-/negative-causation (e.g. *cause_to / prevent_from*), become (e.g. *turn_red*), come_into_existence (e.g. *be_born*), go_out_of_existence (e.g. *die*), scalar_change (e.g. *increase, broaden*, etc.). COS-leave, COS-arrive, COS-transfer groups are for metaphorical or metonymical expressions of change-of-location which belong to change-of-state verb class semantically (e.g. *the water came to a boil*).

3 GESL-based Semi-Automatic Annotation of Event Structure Frame to WordNet

Our main task in WordNet-ESF linking is to assign a proper ESF to each synset of a verb in WordNet. We do the task semi-automatically via the two steps: automatic annotation of ESF with GESL and manual error correction. In section 3.1, we first introduce the automatic event structure tagging tool, GESL. Second, section 3.2 describes the procedure of WordNet-ESF linking.

3.1 The Generator of the Event Structure Lexicon (GESL)

GESL is the automatic event structure annotation tool developed by Im (2013) and Im and Pustejovsky (2009, 2010), which generates an appropriate event structure for each English event-denoting verb in text. Figure 1 shows the input and output of GESL.

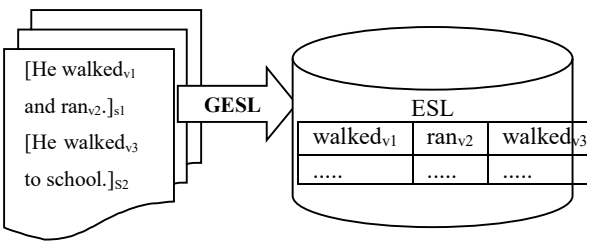

Figure 1. The input and output of GESL

As shown in Figure 1, the input of GESL is a text document. GESL gets English text data and generates the event structure of each event-denoting verb together with its lexical semantic information including its grammatical tense, aspect, and dependencies. For example, if GESL gets the sentence *Osswald killed Kennedy November 22, 1965*, the tool gives the ESL of the event-denoting verb *kill* as its output (Table 2).

verb	KILLED
vid	V1
tense	past
aspect	none
dependency	nsubj (killed, Osswald), dobj (killed, Kennedy), time (killed, November-4)
aspectual class	Transition
semantic class	change-of-state
event type	go_out_of_existence
event structure	se1: pre-state: not_be_killed (Kennedy) se2: pre-state: there_be (Kennedy) se3: process: killing (Osswald, Kennedy) se4: post-state: be_killed (Kennedy) se5: post-state: there_not_be (Kennedy)
sid	S1
sentence	*Osswald killed Kennedy November 22, 1965.*

Table 2. The Event Structure Lexicon of *kill*

Table 2 shows the GESL annotation result of the event-denoting verb *kill* in the special context the sentence generates. GESL classifies the contextual meaning of an English verb into one of the pre-defined event structure types via the three steps of classification – aspectual, semantic, and event type classification. The verb *kill* in the sentence above belongs to transition class aspectually and its semantic class is change-of-state (COS). Finally, the event type of the verb is go-out-of-existence.

GESL goes through several steps to derive the event structure of an event-denoting verb. We show the architecture of GESL in Figure 2.

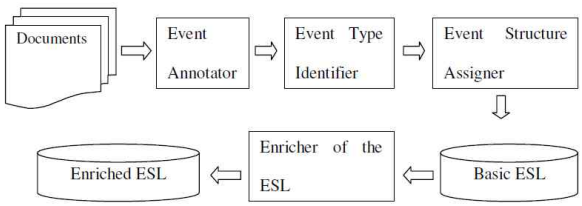

Figure 2. The architecture of GESL

GESL first determines whether a verb in text denotes an event or not. If it denotes an event, it classifies the verb into one of the pre-defined event types via the three classification steps and assigns the proper ESF to the verb. In addition, it links arguments to the semantic roles in the ESF by using the information from the given sentence. The last step is to enrich the event structure by adding synonyms, hypernyms, and antonyms[1].

[1] Refer to Im (2013) if you want to know in more detail about the enriching procedure of the ESL. We can infer additional information like 'Kennedy is dead', 'Kennedy died', 'Kennedy was alive', etc. by the enrichment.

3.2 WordNet-ESF Linking

Because WordNet synsets have their corresponding example sentences, we apply GESL to them in order to annotate the ESF to each synset in WordNet. After automatic annotation of ESF by GESL, we correct errors manually (Figure 3).

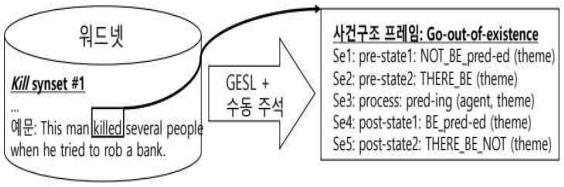

Figure 3. Annotation of ESF to WordNet synset

We have two reasons that we need manual error correction. First, many examples in WordNet synsets are not complete and thus GESL's performance is worse than its ordinary application to text documents. Second, quite many WordNet synsets do not have examples. In those cases, GESL is not applicable. Therefore, we need manual annotation of ESFs.

4 Verb Classes and Pre-defined Event Structure Frames

The ESFs and verb classes in GESL are designed as simple as possible, because it is an automatic annotation system. For instance, GESL does not distinguish between a verb class and its causative counterparts in terms of their ESFs. Instead, the issue is solved by the argument linking algorithm in GESL.

However, the ESFs linked to WordNet need to be more specific than the ESFs in GESL, since WordNet-ESF linking aims to make NLP applications like a textual inference system get the event structure-related inferences only by Word Sense Disambiguation with no other special NLP work.

First, we add its causative counterpart to each verb class (e.g. arrive – cause_arrive). This makes it easier to use the ESF of each synset of English verbs in WordNet without special difficulty in linking arguments to semantic roles in ESFs. Secondly, we separate semelfactive verb class from process class, although Im (2013) did not distinguish the two. The ESFs of the two verb groups are not different. However, we need to consider semelfactive verbs independently. The third change is to divide motion verbs into more specific groups considering motion_direction, motion, self_motion, move_backward, move_down, move_up, pull, push. self_motion verbs do not result in change-of-location. Fourth, the change-of-location verb class originally consists of arrive, leave, transfer but we added move_toward_speaker, move_from_speaker, bring, take, and carry. Fifth, scalar_change verb group is divided into: scale_up, scale_down, and scale_move . The sixth change is to add change_direction and change_posture. Finally, we added precede/follow, happen, maintain, skip, spread, info_transfer, performative (speech act verbs). Appendix A shows the list of verb classes for WordNet-ESF linking and their ESFs. ESFs and verb classes are not limited to the list but can be extended or modified. WordNet has more than 2100 verbs. Our final goal is to assign proper ESFs to all synsets of the verbs. In the next section, we show the examples of annotated ESF.

5 Data: Annotated WordNet Synsets

As of now, we have the ESFs for all synsets of verbs in WordNet by applying GESL to the example sentences in synsets of WordNet. We are working on manual error correction.

In this section, we present the result of experiment with the motion verbs which occur in the season 1 episodes of the drama named "Friends", which will be used in the Video Turing Test (VTT) Project we have been working on since 2017. We use the WordNet version 2.1 embedded in NLTK, Natural Language ToolKit developed at Stanford NLP Lab. The total number of verbs is 91 and they have 952 synsets. We assigned a proper ESF to each synset through automatic annotation by GESL and manual correction of the annotated ESF. We note that one verb can have several different ESFs since different synsets can have different ESFs. For instance, the 41 synsets of the verb *run* has 12 different types of ESF: motion, cause-motion, state, process, follow, leave, spread, change_state, cause-change_state, continue, become .

The scalar_change verbs need more consideration of the kinds of scales. We leave it as a future work.

motion [run.v.1, 6, 11, 28, 33, 34; play.v.18; ply.v.03], **cause-motion** [run.v.26], **change_state** [run.v.24, 41; melt.v.01; ladder.v.01], **cause-change_state** [run.v.31], **continue** [prevail.v.03], **follow** [hunt.v.01], **leave** [scat.v.01], **pass** [run.v.29], **process** [campaign.v.01; carry.v.15; move.v.13; operate.v.01; function.v.01; guide.v.05; race.v.02; run.v.13, 15, 16, 19, 21, 23, 25, 30, 32], **spread**

The target motion verbs are listed in Appendix B. Because the verbs used in the experiment are motion verbs, many synsets belong to motion or change-of-location-related classes. 30.6 % of the synsets (291 out of total 952 synsets) belong to motion or change-of-location-related verb classes. About 40% of the synsets are one of state, process, and change-of-state classes. It is a natural result because those groups have much more verbs than the others.

We additionally assigned the ESFs to the synsets of total 207 verbs including the 85 verbs used in the sentences which describe the scenes of Friends season 1 and their related phrasal verbs and idioms (Appendix C). The scene descriptions were automatically derived by the action recognition algorithm our co-workers developed in the field of Computer Vision. You can see the annotated data in GitHub.[4]

6 Related Work

Since lexical knowledge of words is crucial for various NLP applications including textual inference, computational lexical semanticists have been trying to build lexical resources which annotate many kinds of lexical knowledge. FrameNet, VerbNet, and WordNet, out of the built resources, are well-known and used in the field of NLP and its applications.

FrameNet is a lexical database of English that is both human- and machine-readable with manually annotated sentences, which is based on Frame Semantics (Fillmore, 1976). The basic idea is that the meaning of most words can be understood on the basis of a semantic frame: a description of a type of event, relation, or entity and the participant in it. The FrameNet project is still in progress. However, FrameNet's frames do not annotate the sub-eventual structure of verbs systematically, since it concentrates on semantic roles rather than event structure (Osswald and Van Valin, 2012).

Although VerbNet (Kipper, 2005), a hierarchical verb lexicon based on Levin's classes, also represents sub-eventual structure of verbs, its event structure annotation is neither complete nor consistent (Zaenen et al., 2008). More importantly, neither of the resources has much knowledge about semantic relations of verbs.

WordNet does not include the knowledge about the event structure of verbs but it has the other important factors of lexical semantic knowledge of verbs – semantic relations like synonym, antonym, hypernym, hyponym, etc. Therefore, adding event structure to WordNet will make the resource much more helpful to any NLP applications which need lexical knowledge of verbs. Especially, WordNet-ESF linking would allow us to derive event structure of a verb in text only by Word Sense Disambiguation which maps it to its proper synset, because the synset would have its ESF. In conclusion, WordNet-ESF linking is a good attempt of combining crucial lexical knowledge of verbs.

7 Conclusion

In this paper, we briefly described our semi-automatic annotation task of Event Structure Frames to WordNet synsets via the following two steps. GESL, an automatic event structure annotation tool, assigns a proper ESF to each WN synset of English verbs in WordNet and we correct errors manually. Since each WordNet synset has its own example sentence, GESL, which annotates event structure to verbs in a full sentence, can be applied to the target verb in the sentence so that it annotates an ESF to the verb. If a synset has no example sentence, GESL cannot annotate an ESF to the sysnset. It is one of the reasons that we need manual error correction.

Although WordNet is very useful to develop NLP application tools which require word meaning, it lacks event structure, argument structure, semantic role, and opposition structure. We expect that the enriched WordNet by WordNet-ESF linking will be a big help to NLP applications such as textual or multimodal inference tasks.

For WordNet-ESF linking, we extended ESF-dependent verb classes in GESL in order to represent the event structural meaning of each synset of verbs more specifically. GESL has 23 verb classes and each of them has its own event structure frame. We suggest 44 classes and their causative counterparts in this paper. The classes are not fixed. Since we still work on the WordNet-ESF linking task, verb classes can undergo change.

Acknowledgments

This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-01780, The

[run.v.27, 30], **state** [run.v.05, range.v.01, tend.v.01], **become** [run.v.14]

[4] https://github.com/ish97/VTT/blob/master/

technology development for event recognition/relational reasoning and learning knowledge based system for video understanding).

References

Fillmore, Charles J. 1976. Frame Semantics and the Nature of Language. *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech* 280: 20-32.

Im, Seohyun. 2013. *The Generator of the Event Structure Lexicon (GESL): Automatic Annotation of Event Structure for Textual Inference Tasks*. PhD Dissertation, Brandeis University, MA, USA.

Im, Seohyun and James Pustejovsky. 2010. Annotating Lexically Entailed Subevents for Textual Inference Tasks. In the *Proceedings of FLAIRS 23*, Daytona Beach, Florida, USA, 2010.

Im, Seohyun and James Pustejovsky. 2009. Annotating Event Implicatures for Textual Inference Tasks. In the *Proceedings of the 5th International Conference on Generative Approaches to the Lexicon*, Pisa, Italy, 2009.

Kipper, Karin Schuler. 2005. VerbNet: A Broad-coverage, Comprehensive Verb Lexicon. *PhD Dissertation*. University of Pennsilvenia.

Miller, George A. 1995. Wordnet: A Lexical Database for English. *Communications of the ACM* 38, no. 11.

Osswald, Rainer and Jr. Robert D. Van Valin. 2012. FrameNet, Fame Structure, and the Syntax-Semantics Interface (draft).

Pustejovsky, James. 2000. Events and the Semantics of Opposition, Pustejovsky and Tenny (eds.) *Events as Grammatical Objects*. CSLI Publications.

Pustejovsky. James. 1995. *The Generative Lexicon*. The MIT Press.

Zaenen, Annie, Cleo Condoravi, and Danny. Bobrow. 2008. The Encoding of Lexical Implications in VerbNet. *Proceedings of LREC 2008*. Morocco, 2008.

Appendix A. Verb Classes and Event Structure Frames

* CAUSATIVE counterparts: causer-argument added

STATE
se1: state: pred-ing_(prep) (theme)
PROCESS [cause_process]
se1: process: pred-ing_(prep) (agent)
SEMELFACTIVE [cause_semelfactive]
se1: process: pred-ing_(prep) (theme)
MOTION [cause_motion]
d-se1: pre-state: be_loc-prep (theme, source)
se1: process: pred-ing (theme)
d-se2: post-state: be_loc-prep (theme, goal)
MOVE_BACK [cause_move_back]
d-se1: pre-state: be_loc-prep (theme, source)
se1: process: pred-ing_back (theme)
d-se2: post-state: be_loc-prep (theme, goal)
d-se3: post-state: be_behind (goal, source)
d-se2 = d-se3
MOVE_UP [cause_move_up]
d-se1: pre-state: be_loc-prep (theme, source)
se1: process: pred-ing_up (theme)
d-se2: post-state: be_loc-prep (theme, goal)
d-se3: post-state: be_higher_than (goal, source)
d-se2 = d-se3
MOVE_DOWN [cause_move_down]
d-se1: pre-state: be_loc-prep (theme, source)
se1: process: pred-ing_downward (theme)
d-se2: post-state: be_loc-prep (theme, goal)
d-se3: post-state: be_lower_than (goal, source)
d-se2 = d-se3
MOVE_TOWARD_SPEAKER
[cause_move_toward_speaker]
d-se1: pre-state: be_loc-prep (theme, source)
se1: process: pred-ing (theme)
d-se2: post-state: be_loc-prep (theme, goal)
d-se3: post-state: be_near (goal, speaker's location)
d-se2 = d-se3
MOVE_FROM_SPEAKER
[cause_move_from_speaker]
d-se1: pre-state: be_loc-prep (theme, source)
se1: process: pred-ing (theme)
d-se2: post-state: be_loc-prep (theme, goal)
d-se3: post-state: not_be_near (goal, speaker's location)
PULL
d-se1: pre-state: be_loc-prep (theme, source)
se1: process: pred-ing (agent, theme)
d-se2: post-state: be_loc-prep (theme, goal)
PUSH
d-se1: pre-state: be_loc-prep (theme, source)
se1: process: pred-ing (agent, theme)
d-se2: post-state: be_loc-prep (theme, goal)
CARRY
se1: process: pred-ing (agent, theme)
se2: state: having (agent, theme)
se1 = se2
LEAVE [cause_leave]
se1: pre-state: be_loc-prep (theme, source)
se2: process: pred-ing (theme)
se3: post-state: not_be_loc-prep (theme, source)
PASS [cause_pass]
se1: pre-state: be_loc-prep (theme, source)
se2: process: pred-ing (theme)
se3: state: be_loc-prep (theme, path)
se4: post-state: be_loc-prep (theme, goal)
se2 = se3
ARRIVE [cause_arrive]
se1: pre-state: not_be_loc-prep (theme, goal)
se2: process: pred-ing (theme)
se3: post-state: be_loc-prep (theme, goal)

TRANSFER [cause_transfer]
se1: pre-state: be_loc-prep (theme, source)
se2: process: pred-ing (theme)
se3: post-state: be_loc-prep (theme, goal)
SPREAD [cause_spread]
se1: pre-state: not_be_over (theme, ground)
se2: process: pred-ing (agent, theme, ground)
se3: post-state: be_over (theme, ground)
BRING
se1: pre-state: not_be_loc-prep (agent & theme, goal)
se2: process: pred-ing_goal-prep (agent, theme, goal)
se3: post-state: be_loc-prep (agent & theme, goal)
TAKE
se1: pre-state: be_loc-prep (agent & theme, source)
se2: process: pred-ing_source-prep (agent, theme, source)
se3: post-state: not_be_loc-prep (agent & theme, source)
LOSE [cause_lose]
se1: pre-state: have (possessor, theme)
se2: process: pred-ing (possessor, theme)
se3: post-state: not_have (possessor, theme)
GET [cause_get]
se1: pre-state: have (recipient, theme)
se2: process: pred-ing (recipient, theme)
se3: post-state: not_have (recipient, theme)
GIVE
se1: pre-state: have (possessor, theme)
se2: process: pred-ing (possessor, recipient, theme)
se3: post-state: have (recipient, theme)
EXCHANGE
se1: pre-state: have (possessor, theme1)
se2: pre-state: have (recipient, theme2)
se3: process: pred-ing (possessor, recipient, theme1, theme2)
se4: post-state: have (possessor, theme2)
se5: post-state: have (recipient, theme1)
INFO_TRANSFER
se1: pre-state: have (possessor, theme:info)
se2: process: pred-ing (possessor, theme:info)
se3: post-state: have (possessor & recipient, theme:info)
COME_INTO_EXISTENCE
[cause_come_into_existence]
se1: pre-state: not_be_pred-ed (theme)
se2: pre-state: there_be_not (theme)
se3: process: pred-ing (theme)
se4: post-state: be_pred-ed (theme)
se5: post-state: there_be (theme)
GO_OUT_OF_EXISTENCE
[cause_go_out_of_existence]
se1: pre-state: not_be_pred-ed (theme)
se2: pre-state: there_be (theme)
se3: process: pred-ing (theme)
se4: post-state: be_pred-ed (theme)
se5: post-state: there_be_not (theme)
BECOME [cause_become]
se1: pre-state: not_be_pred-ed (theme, state)
se2: pre-state: not_be (theme, state)
se3: process: pred-ing (theme, state)
se4: post-state: be_pred-ed (theme, state)
se5: post-state: be (theme, state)
BEGIN [cause_begin]
se1: pre-state: not_in_progress (event)
se2: process: pred-ing (event)
se3: post-state: in_progress (event)
CONTINUE [cause_continue]
se1: pre-state: in_progress (event)
se2: process: pred-ing (event)
se3: post-state: in_progress (event)
END [cause_end]
se1: pre-state: in_progress (event)
se2: process: pred-ing (event)
se3: post-state: not_in_progress (event)
POSITIVE_CAUSATION
se1: pred-ing (causer, event)
se2: happen (event)
NEGATIVE_CAUSATION
se1: pred-ing (causer, event)
se2: not_happen (event)
SCALE_UP [cause-scale_up]
d-se1: pre-state: be_loc-prep (theme, source_scale)
se1: process: pred-ing (theme)
d-se2: post-state: be_loc-prep (theme, goal_scale)
d-se3: post-state: be_higher_than (goal, source_scale)
d-se2 = d-se3
SCALE_DOWN [cause-scale_down]
d-se1: pre-state: be_loc-prep (theme, source_scale)
se1: process: pred-ing (theme)
d-se2: post-state: be_loc-prep (theme, goal_scale)
d-se3: post-state: be_lower_than (goal, source_scale)
d-se2 = d-se3
SCALE_MOVE [cause-scale_move]
se1: process: pred-ing (theme, scale)
CHANGE_DIRECTION [cause-change_direction]
se1: pre-state: not_be_pred-ed (theme)
se2: pre-state: be (theme, source_direction)
se3: process: pred-ing (theme)
se4: post-state: be_pred-ed (theme)
se5 = post-state: be (theme, goal_direction)
CHANGE_POSTURE [cause-change_posture]
se1: pre-state: not_be_pred-ed (theme)
se2: pre-state: be (theme, source_posture)
se3: process: pred-ing (theme)
se4: post-state: be_pred-ed (theme)
se5: post-state: be (theme, goal_posture)
CHANGE_STATE [cause_change_state]
se1: pre-state: not_be_pred-ed (theme)
se2: pre-state: be (theme, source_state)
se3: process: pred-ing (theme)
se4: post-state: be_pred-ed (theme)
se5: post-state: be (theme, goal_state)
COS_LEAVE [cause_cos_leave]
same as the ESF of LEAVE
COS_ARRIVE [cause_cos_arrive]
same as the ESF of ARRIVE
COS_TRANSFER [cause_cos_transfer]
same as the ESF of TRANSFER
PERFORMATIVE (speech act)

se1: pre-state: not_be_pred-ed_to_by (theme, addressee, speaker)
se2: process: pred-ing (speaker, addressee, theme)
se3: post-state: be_pred-ed_to_by (theme, addressee, speaker)
HAPPEN [cause_happen]
se1: state: there_be (event)
MAINTAIN
se1: pre-state: be (state)
se2: process: pred-ing (agent, state)
se3: state: be (state)
se2 = se3
PRECEDE
se1: state: pred-ing (theme1, theme2)
se2: state: be_before (theme1, theme2)
se1 = se2
FOLLOW
se1: state: pred-ing (theme1, theme2)
se2: state: be_after (theme1, theme2)
se1 = se2

Appendix B. The list of motion verbs in Friends Season 1 episodes

arrive, back, bail, barge, base, board, bring, brush, bury, camp, carry, chase, clean, come, conduct, creep, dance, dip, drag, draw, drift, drive, drop, dump, enter, erase, fall, fax, fling, float, flush, fly, follow, go, head, hike, hop, inch, invade, jump, kick, land, lay, lead, leave, load, move, park, pass, plunge, pop, pour, pull, push, put, raise, reach, remove, return, ride, roll, run, rush, send, ship, shove, shuffle, sit, ski, skip, slather, slide, slip, stand, step, stomp, sweep, swoop, take, throw, travel, tremble, turn, twist, usher, vacuum, walk, wave, wind, wipe, wobble

Appendix C. The list of verbs in the scene description sentences provided by a Computer Vision Action Recognition algorithm

apply, assemble, attack, bark, beat, box, burn, celebrate, cheer, clean, comb, cook, crash, cry, cut, decorate, demonstrate, drink, dunk, eat, explain, explode, fight, film, fish, fix, floor, fold, give, have, hit, hold, hug, hunt, install, interact, interview, involve, kiss, lick, lie, make, mix, paint, perform, pet, ping, place, play, pose, preform, prepare, punch, race, read, record, rub, scoop, score, scream, sew, shoot, show, sing, skate, ski, sleep, slice, smash, smile, solve, speak, spray, stretch, surf, swim, talk, teach, use, wash, watch, weave, work, wrestle, write