Machine Translation Based on WordNet and Dependency Relations

Luchezar Jackov
Institute for Bulgarian Language
Bulgarian Academy of Sciences
lucho@skycode.com

Abstract

The proposed machine translation (MT) approach uses WordNet (Fellbaum, 1998) as a base for concepts. It identifies the concepts and dependency relations using context-free grammars (CFGs) enriched with features, role markers and dependency markers. Multiple interpretation hypotheses are generated and then are scored using a knowledge base for the dependency relations. The hypothesis with the best score is used for generating the translation. The approach has already been implemented in an MT system for seven languages, namely Bulgarian, English, French, Spanish, Italian, German, and Turkish, and also for Chinese on experimental level.

1. Introduction

Any translation must properly convey the concepts and the relations between them from the source to the target language. This includes correct identification of the concepts (i.e., word sense disambiguation) and correct identification of the relations between them (their dependency relations). These concepts and relations must be properly projected into the target language so that they can be correctly identified (understood) by the recipient of the translation.

The article proposes an approach for generation and semantically driven evaluation of interpretation hypotheses as part of an MT system. The derived hypotheses embed and evaluate the morphological, syntactic and semantic information simultaneously instead of in a pipeline. Recent developments (Bohnet et al., 2013) show the advantages of performing morphological and syntactic analysis jointly, obviating the use of a part-of-speech tagger. Our approach goes further by performing morphological, syntactic and semantic analysis jointly. The best hypotheses are chosen by using a semantic scoring mechanism that works on the relations that each hypothesis identifies. A method for performing parse selections based on semantic knowledge has been proposed in (Fujita et al., 2010).

The article presents work in progress, and no extensive comparison of the translation results has been done yet. However, the proposed MT approach is used in the SkyCode machine translation system. It has been implemented in C++ and has a very compact binary data representation, approx. 60MB for 7 languages and 42 language translation directions. It has been used in offline translation applications for mobile devices, outperforming Google Offline Translator in both quality and size (the latter needs about 1.05GB of data for 7 languages). The system has also participated successfully in the *iTranslate4* project, and can be tested online at http://itranslate4.eu (the SkyCode vendor). The system consists of a lemmatizer, a concept binder, a hypothesis generator, a dependency relations scorer and a synthesis unit.

2. Lemmatizer

The lemmatizer analyzes the smallest bits that the system works on: the tokens. For every token the lemmatizer yields a list of all lemmas that have a word form equal to the token. Each entry in the list consists of the lemma identifier in the database and the morphological features of the word form.

The lemmatizer database consists of entries where each entry holds an identifier, a lemma (or a base form of the word), an inflection group identifier, and a paradigm identifier. Each inflection group is a set of inflection entries consisting of a suffix and its respective features. In this way, all word forms of the lemma are defined. The input word form can be lemmatized with the inflection features extracted, and any word form can be generated by specifying the lemma and the respective features.

The result of applying the lemmatizer over each token is a list of lemma entries. Each entry consists of a lemma identifier and a set of features. The lemma entries list is used by the concept binder to yield initial interpretation hypotheses for the token. For instance, "water" will yield two lemma entries, one for the noun and one for the verb. The Bulgarian surface form of ми (*mi*, "me") will yield an entry for the dative/genitive/possessive form of the personal pronoun аз (*az*, "I") and another two for the second and third person past forms of the verb мия (*miya* "to wash").

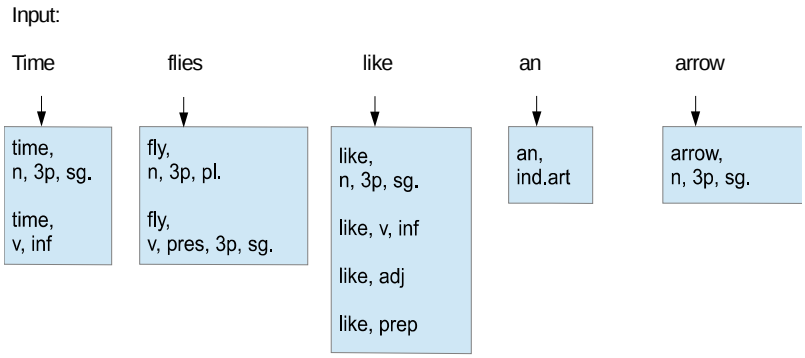

Figure 1: Lists of lemma entries resulting from the application of the lemmatizer over each token. The lemma and language identifiers for each entry are omitted for brevity.

The proposed approach considers every possible lemmatization of the token producing one or more interpretation hypotheses. The lemmatization disambiguation occurs naturally when scoring the different hypotheses and disregarding the low-scored ones. This obviates the use of part-of-speech taggers, which are known to introduce errors that cannot be handled further in the process.

We have developed dictionaries containing 115,735 lemmas for English, 102,393 for Bulgarian, 38,445 for Turkish, 135,171 for German, 68,026 for Spanish, 65,866 for French, and 59,883 for Italian as part of the SkyCode MT system.

3. Concept Binder

The concept binder database links each WordNet concept (its synset identifier) to a list of one or several lemmas that observe agreement restrictions. The database is used by the concept binder to identify concepts in the input language and to generate translations in the output language.

3.1. Database Structure

The concept binder database consists of entries having the following fields:

- a language identifier;
- a base form (a descriptive string, usually matching the base form of the constituting lemmas);
- a hypothesis type identifier (HTI);
- a list of lemma identifiers;
- a WordNet synset identifier;
- restrictions on features of each lemma;
- unification of features of each lemma;
- a list of additional features.

The base form is used only for easy lookup and management of the database.

The hypothesis type identifier (HTI), as used in this article, corresponds to some extent to the non-terminal symbols of a classical CFG. Here are some of the HTIs used in the system: *Verb, Adjective, Noun, Personal_pronoun, Demonstrative_pronoun, Direct_object, Indirect_object, Verb_phrase, Noun_phrase, Prepositional_phrase, Subject_phrase, Sentence*, etc.

The list of lemma identifiers is used for both concept identification and translation generation. The restrictions on features of each lemma allow identifying concepts that are defined by a specific word form and not by all of the word forms, which is usually encountered in multiword expressions (MWEs). The unification of features of each lemma is used for MWEs. The list of additional features is used to define sub-categorization frames, mass/plural count nouns, etc.

The SkyCode MT system currently has 285,171 concept binder entries for English, 166,948 for Bulgarian, 118,832 for Turkish, 213,421 for German, 162,545 for Spanish, 183,479 for French, and 140,836 for Italian. The concept binder data for English has been automatically imported from the Princeton WordNet 3.0, while the rest has been developed independently. Similar resources exist for some of the languages (e.g., Bulgarian – cf. (Koeva, 2010), but they were either not available or not freely accessible when the development of the system started.

3.2. Identifying Concepts

The concept binder works on the lists of lemmatized tokens created by the Lemmatizer. It generates all the possible interpretations for one or more consecutive tokens and the result comprises the initial interpretation hypotheses on which the hypothesis generator works. For instance, running the concept binder over the token “water” (lemmatized to [water, n, English] and [water, v, English]) will yield the following interpretation hypotheses: 6 instances with HTI of “noun” bearing the respective WordNet synset identifiers and 5 instances with HTI of “verb” bearing the respective WordNet synset identifiers.

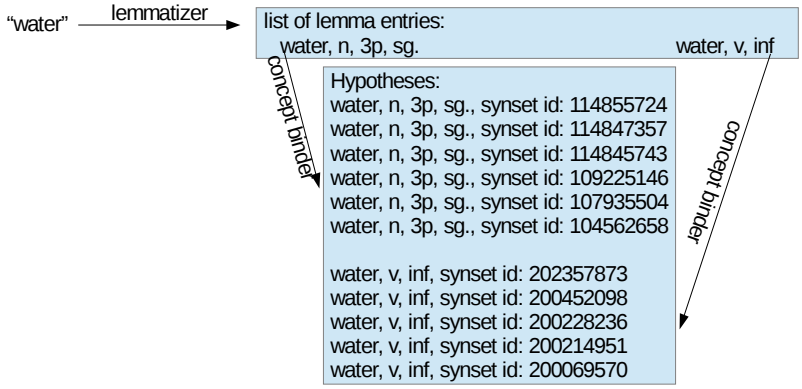

Figure 2: Concept binder being run over the output of the lemmatizer. The lemma, language and concept binder identifiers are omitted for brevity.

The concept binder is run for spans up to 9 tokens to find multiword expressions (such as “guinea pig”) and yield interpretation hypotheses for them. Each hypothesis comprises a particular WordNet concept and a particular projection (translation) of the WordNet concept in the target language if there is more than one translation of the concept.

The hypotheses derived from several language units by the concept binder are considered along with the hypotheses created by applying the rules over the single-lemma hypotheses. For instance, “to kick the bucket” will be considered as a hypothesis for a single concept (“to die”) having HTI of “Verb”. It will also be considered as a hypothesis with HTI of “Verb_phrase” and roles and dependencies identified in concert with the literal meaning of *to kick a bucket*.

3.3. Generating Translations for Concepts

The concept binder database is also used to generate translations in the target language. For each source language concept one or several translations are retrieved from the database by filtering the entries that

match the target *language id* field and the *WordNet synset id* field of the source concept. Each translation is generated by looking for the lemmas in the lemmatizer database and inflecting each of them into the appropriate word form.

4. Hypotheses Generator and Parsing Rules

The hypotheses generator groups hypotheses of adjacent spans of the input text by trying to apply each of the parsing rules (based on enriched CFGs) over them. A parsing rule can be applied if the hypotheses to be grouped meet the parsing rule criteria, thus yielding new interpretation hypotheses for the span that includes the adjacent spans whose hypotheses are grouped. The hypothesis generator (parse generator) uses the Cocke–Younger–Kasami (CYK) algorithm (Cocke et al., 1970; Younger, 1967; Kasami, 1965), modified with scoring and pruning to prevent search space explosion.

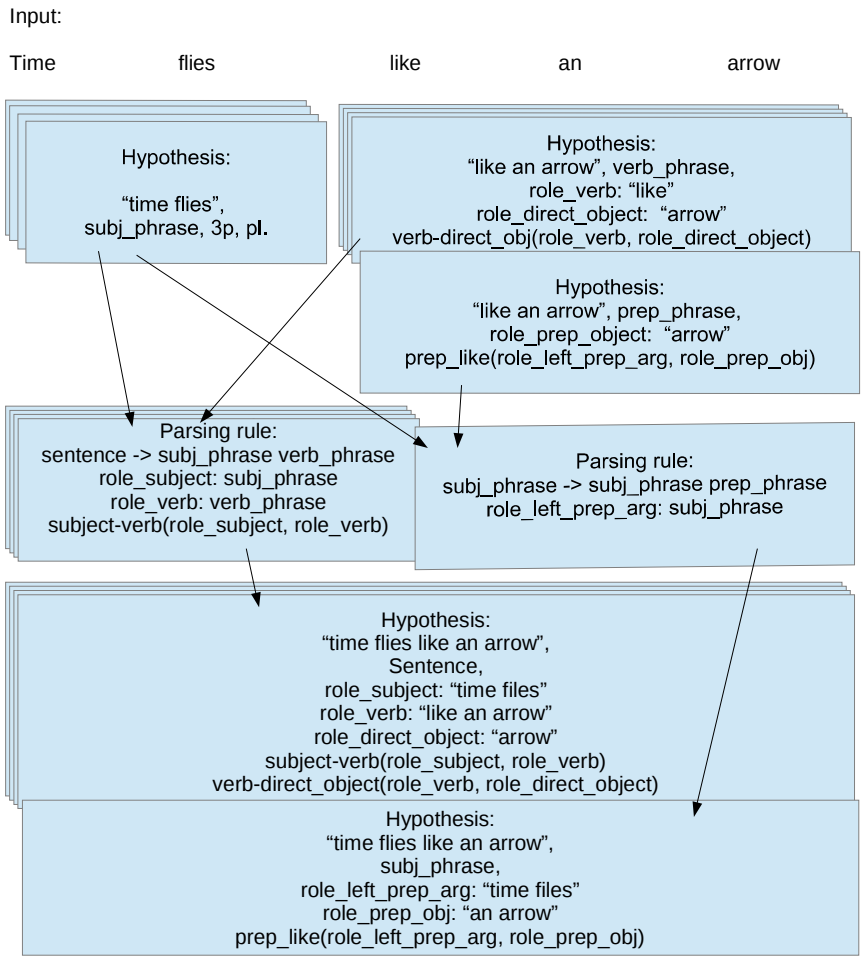

Figure 3: Parsing rules being applied to hypotheses yield hypotheses for broader spans. Even though the illustrated hypotheses seem unlikely for the sample input text, this may not be so for other input text (e.g., “time travels seem an illusion”). The likeliness is evaluated as a hypothesis score by looking up for the identified dependency relations in the knowledge base. Note that Figure 3 shows just one of the possible splits but other splits are also considered, such as the correct one, [S → SP VP (“time”, *subj_phrase*) (“flies like an arrow”, *verb_phrase*)]. When having good knowledge base, the latter hypothesis will receive the best score.

Each parsing rule used by the hypothesis generator assumes roles and dependency relations. The result of the successful application of a parsing rule is a new interpretation hypothesis that includes the assumed roles and dependency relations as part of it. The hypothetical dependency relations between the assumed roles are scored by the dependency relations scorer using the dependency relations knowledge

base. Thus, each interpretation hypothesis is scored and the worst hypotheses are pruned to prevent search space explosion. Currently, the system identifies the following relations and roles (inexhaustive): *subject-verb, verb-direct_obj, verb-indirect_obj, modal_verb-verb, adjective-noun*, etc. It also identifies a number of language-dependent prepositional relations, such as *prep_in(left_prep_argument, prep_object)*.

4.1. Parsing Rules

The parsing rules are the equivalent to the rewriting rules of classical CFG. A classical CFG rewriting rule, when used for analysis, selects or restricts the non-terminals that would build the resulting non-terminal. Unlike CFG, the parsing rules hold data for additional restrictions over the features of the constituent hypotheses. Such restrictions are used to define rules for specific sub-categorizations, agreement rules, etc. The parsing rules are manually developed. Each parsing rule can be either unary or binary. It consists of:

1. A list of one (unary rule) or two (binary rule) entries. Each entry defines the restrictions on the hypothesis that would take the entry position. The following data restricts the candidate hypothesis:
    - A hypothesis type identifier (HTI);
    - A list of restrictions over the features of the hypothesis;

Example: *VP → V NP should be restricted only for transitive verbs. Such verbs have the "transitive"feature defined in the concept binder. The restriction for transitiveness is in this list.*

    - A list of features being inherited (i.e., feature unification data);

Example: *In composite past tenses in Bulgarian, the auxiliary verb does not have a gender feature, but has person and number features. The past participle has gender and number features. Gender is inherited from the past participle, while person and number features are inherited from the auxiliary verb. The resulting hypothesis has unified gender, number, and person features that will be used later to account for the subject-verb agreement on these features.*

    - Role markers: *(e.g., role_subject, role_verb, role_direct_object, role_indirect_object, role_prep_object, role_left_prep_argument)*;
    - A list of role markers (to be inherited).

Complex interpretation hypotheses may identify more than one role. When grouping such hypotheses, the parsing rule inherits the pointers to the role markers from the hypotheses that are being grouped.

Example: *A unary rule for the preposition "in" introduces the relation prep_in(role_left_prep_argument, role_prep_object). Another parsing rule groups the preposition hypothesis with a noun phrase hypothesis and sets its role to prep_object to yield a prepositional phrase hypothesis. This hypothesis carries the preposition_role and the prep_object role pointing to the particular concepts within the hypothesis. Another rule binds a noun phrase to the prepositional phrase. This rule inherits the role pointers to the preposition and to the prepositional object.*

    - A list of features that the hypothesis should agree with any of the roles that the parsing rule identifies.

Example: *In S → NP VP the verb phrase should agree with the subject noun phrase. The rule marks the first entry (the NP) with "role_subject" and defines that the feature list [gender, person, number] of the second entry should agree with "role_subject". If the agreement is not met, the rule is not applied.*

2. A list of dependency relations where each entry holds:
    - A relation identifier;
    - Role markers for the first argument and for the second argument.

Example: *A parsing rule for a subject phrase with a verb phrase subcategorized for possession. The rule introduces a possession dependency relation between the subject and the direct object. A general rule for non-possession verbs would introduce only the subject-verb and verb-direct_obj relations.*

3. Resulting HTI and features

Example: *VP → V NP will have HTI of "V" for the first entry, HTI of "NP" for the second entry, and a resulting HTI of "S". The resulting features are used to add information on what the parse tree lying under the hypothesis contains. For instance, a verb phrase with that-clause is unlikely to be bound to a*

prepositional phrase. This can be described by having a resulting feature "+that-clause" on the VP → V CP rule, and having a "not(+that-clause)" restriction on the VP → VP PP rule.

4. A list of languages that the parsing rule can be applied on.
5. A list of languages that the rule can be used to translate into.
6. Rule score that is added to the total hypothesis score.

Example 1: *Rules that handle commonly encountered but grammatically incorrect constructions.*
Example 2: *Rules that handle inverse word order in free word order languages. Such rules are defined with a lower score, giving precedence to the rules that would handle the canonical word order.*

A parsing rule is applied to adjacent interpretation hypotheses if they obey the feature and agreement restrictions. When the feature and agreement restriction lists are empty, the rule will not apply any feature restrictions.

The data structure holding each newly yielded interpretation hypothesis preserves pointers to its constituents, the rule that has been applied, the roles that have been identified, and the dependency relations that have been introduced, so that the hypothesis can be scored.

Example: *(DO → NP) a unary rule for a noun in accusative case (for case languages) that generates a new hypothesis with HTI of "Direct_object".*
Example: *(DO → Ppr) a unary rule for a personal pronoun in accusative case (e.g., in Bulgarian) that generates a new hypothesis with HTI of "Direct_Object".*
Example: *(VP → Vtr DO) a binary rule that binds a transitive verb with the direct object. The first entry has the following data:*

- HTI is "Verb".
- It must have a sub-categorization feature "transitive_verb".
- Its role marker is set to "role_verb".

The second entry has the following data:

- HTI is "Direct_object".
- Its role marker is set to "role_Direct_object".

The parsing rule introduces the dependency relation verb-direct_obj (role_verb, role_direct_object). It will group hypotheses with HTI of "Verb" with hypotheses having HTI of "Direct_Object" to yield a new hypothesis with HTI of "Verb_Phrase" (Verb_Phrase → Verb Direct_Object). This parsing rule is common for English, German, Spanish, French, Italian, and Bulgarian. This specific rule will not cover all cases for all languages, as the direct object can stand before the verb in German, and in Bulgarian for cases where a pronoun is the direct object.

There are 5,598 parsing rules, of which 2,085 rules are shared by more than one language.

4.2. Hypothesis Generator

The hypothesis generator is a modified version of the CYK algorithm. Given a list of language units from 1 to n, it sequentially derives hypotheses for spans starting from 1 and having length of 1, then, length of 2, then length of 3, and so on to length of n-1 by applying the parsing rules on every possible split of the span being considered. Each interpretation hypothesis for each span is stored in a three-dimensional array where the first index denotes the span start, the second index denotes the span length, and the third index denotes the hypothesis position in the hypotheses list.

4.2.1. General Algorithm Description

Let's assume that the input text is "Time flies like an arrow". The hypothesis generator will first derive interpretation hypotheses for span of length 1 starting at position 1 ([1,1]), i.e., for the token "time" by running the concept binder over the lemmatizer output of "time". Then it will derive hypotheses for "Time flies" by first deriving hypotheses for "flies", i.e., span of length 1 starting at position 2 ([2,1]). Then it will try to apply parsing rules over the two spans [1,1] and [2,1], yielding hypotheses for span [1,2] ("time flies"). It will continue by deriving hypotheses for span [3,1] ("like"), [2,2] ("flies like"), [1, 3] ("time flies like"). Eventually it will generate hypotheses for the span [1,4] ("Time flies like an arrow").

Multiple hypotheses are derived for each span (see Figure 3). For instance, “flies” is the third person singular present form of the verb “fly”, but it is also the plural of the noun “fly”. The verb “fly” has 14 WordNet senses and for each sense the concept binder yields an interpretation hypothesis. Each hypothesis holds particular bindings to the WordNet concepts and the presumed relations between them, which makes it possible for the dependency scorer to look up the dependency relation instances in the knowledge base.

4.2.2. Application of the Parsing Rules

The parsing rules are applied on adjacent spans by trying to apply each parsing rule over the Cartesian product of the hypotheses for the two spans. Let's assume that “flies” yields two hypotheses, one as a noun and one as a verb. Let's have two parsing rules, S → NP VP, NP → N N. Applying the parsing rules over the two fragments [(“time”, N)] and [(“flies”, N), (“flies”,V)] will yield [(“time flies”, S), (“time flies”, NP)]. Even though the second hypothesis is unacceptable from a semantic point of view, it is a legitimate syntactic parse and a legitimate hypothesis. However, this hypothesis will receive a low score and will eventually be pruned, since the hypothesized dependencies between the hypothesized concepts do not have a match in the dependency relations knowledge base.

4.2.3. Telling the Good Hypotheses from the Bad Ones

The data structure behind each interpretation hypothesis stores the roles and the dependency relations identified as part of the hypothesis. Each dependency relation that has its arguments (role markers) bound to particular concepts, is scored by the dependency relations scorer.

5. Dependency Relations Knowledge Base and Scoring

The dependency relations knowledge base consists of quadruples containing a relation identifier, two concept identifiers for the relation arguments, and scoring weight. The weight can be positive or negative. The scorer evaluates each hypothesis by looking in the knowledge database for all of the dependency relations between the particular concepts that the hypothesis has identified and summing the weights, thus forming the hypothesis score.

By applying the parsing rules, the hypothesis generator defines particular dependency relations between the concepts of each generated interpretation hypothesis. For instance, it hypothesizes the relation *subject-verb(time, fly)* for the hypothesis (“time flies”, S), and *attrib_english(time, fly)* for the hypothesis (“time flies”, NP). The knowledge base consists of entries giving scores for such instances (e.g., *relation*(subject-verb, time, fly) = 1, *relation*(attrib_english, time, fly) = 0). The scorer looks up for the particular dependency relations entries in the knowledge base and adds the entry score to the hypothesis score whenever it finds a matching entry. Thus, each hypothesis receives a score, and low-scored hypotheses are pruned to prevent search space explosion.

5.1. Knowledge Base over WordNet Synsets

Having a knowledge base over WordNet synsets allows reusing it for analyzing different languages that have WordNets bound to the Priceton WordNet synsets. Each knowledge base entry consists of a relation identifier, two synset identifiers, and relation score (usually 0, 1 or -1). Each hypothesis has a number of hypothesized relations, namely a relation identifier and two concepts (i.e., two synset identifiers). For each such relation instance, the scorer looks up for matches of the triple (*rel_id, synset_id1, synset_id2*) in the knowledge base and adds the resulting score to the hypothesis score.

5.2. Knowledge Base over Lemmas

Having a knowledge base over lemmas allows making fine distinctions between members of the same WordNet synset in the translation synthesis. Each knowledge base entry consists of a relation identifier, two concept binder base forms and relation score. For each relation identified by a given hypothesis, the scorer retrieves the concept binder base forms of the translated concepts and forms a triple having (*rel_id, arg1_base_form, arg2_base_form)*. The scorer looks up for matches of this triple and adds the resulting score to the hypothesis score.

5.3. Data Sparseness

The main challenge to the proposed system is the data sparseness of the dependency relations knowledge base. One way of overcoming this challenge is to use the WordNet hypernym relations and manually populate relation instances over hypernyms. For instance, the relation *verb-direct_obj(play, musical instrument)* can yield the same relation for the "musical instrument" hyponyms. Unfortunately, this approach is not productive enough.

Another way is to use the MT system for automatic collection of lemma-based dependency relations knowledge from monolingual corpora. This can be achieved by translating sentences of the corpora and recording the dependency relations over the particular source language lemmas identified by the best interpretation hypothesis. This data can be used in the hypothesis scoring by using the translated concepts as relation arguments when looking up the lemma-based knowledge base. The data can be used to infer relation instances between WordNet synsets by running the system over a set of several languages (e.g., English, French, Spanish, German, Italian, and Bulgarian) and picking the most complete synset clusters.

6. Translation Synthesis

Each hypothesis is a parse tree consisting either of sub-trees or of concept binder entries. Creating a translation of the hypothesis includes constituent reordering, various agreements, etc. for each parsing rule. There is a set of synthesis rules for each parsing rule that takes care of word reordering, insertion, deletion, etc. when creating the translation output. The rules are manually written. For instance, when translating "I gave him the book", the hypothesis generator identifies the structure [I (*subj*) [[gave him (*verb-ind_obj*)] the book (*v_ind_obj-dir_obj*)]. When translating it into Bulgarian, there is a synthesis rule for the *verb-ind_obj* rule that checks whether the indirect object is a pronoun, whether the rest of the translation has a missing subject, or whether it is negative, to achieve the correct word order:

I gave him the book. = Дадох му книгата. (Dadoh mu knigata)
I gave John the book. = Дадох книгата на Джон (Dadoh knigata na Dzhon)
John gave him the book. = Джон му даде книгата. (Dzhon mu dade knigata)
I haven't given him the book. = Не му дадох книгата. (Ne mu dadoh knigata)

The leaves of the hypothesis parse tree are concept binder entries and are translated by looking up the concept binder database for entries that match the source concept synset in the target language and inflecting them (see 3.3).

More than one synthesis rule can be defined for each parsing rule. The competing synthesis rules add language-specific relation dependencies that are also scored by the Dependency relations scorer.

Example: *A noun phrase with a simple prepositional phrase can be expressed in English as an attributive, e.g.,"months of spring" and "spring months". There are two competing synthesis rules, one introducing prep_of relation and the other introducing attrib_english relation. The rule that gets the higher score is chosen over the other rule.*

7. Conclusion

The article provides an overview of a machine translation system based on WordNet and dependency relations. There is a working prototype of this system implemented in C++ for seven languages (42 language directions): English, French, German, Spanish, Italian, Turkish, and Bulgarian that can be tested online at `http://itranslate4.eu` (SkyCode translation vendor).

One of the main challenges to the proposed system is the populating of the knowledge base and mitigating the data sparseness. There are several approaches to overcome this.

One approach involves manual population of the knowledge base; it has proven to yield very good results in terms of parsing accuracy for any given sentence. This is further improved by populating relations over the WorNet concepts (hypernyms) and the Dependency scorer is modified to look for relations between the hypernyms of the arguments when no direct match is found.

Another approach includes automatic collection of relation instances over lemmas. The system produces scored hypotheses with dependency relations over the lemmas. The best hypothesis can be used

to populate language-specific lemma-based knowledge base. This knowledge base can be reused when translating into the language that the knowledge base is for. Running the system over the Europarl corpus yielded some 33 million knowledge entries for six languages (English, French, German, Italian, Spanish, and Bulgarian).

A third approach employs automatic derivation of WordNet-based dependency relations by picking a lemma-based relation, generating all possible WordNet-based hypotheses, and choosing the one that is most consistent with the lemma knowledge base in different languages and WordNet synonyms. A test version of the relation inference module over the 33 million lemma-based knowledge entries yielded some 1.32 million synset-based knowledge entries.

Acknowledgements

The present paper was partially prepared within the project *Integrating New Practices and Knowledge in Undergraduate and Graduate Courses in Computational Linguistics* (BG051PO001-3.3.06-0022) implemented with the financial support of the Human Resources Development Operational Programme 2007 – 2013 co-financed by the European Social Fund of the European Union. The author takes full responsibility for the content of the present paper.

References

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Bohnet, B., Nivre, J., Boguslavsky, I., Farkas, R., Ginter, F., and Hajic, J. (2013). Joint Morphological and Syntactic Analysis for Richly Inflected Languages. *Transactions of the Association for Computational Linguistics*, 1:415–428.

Fujita, S., Bond, F., Oepen, S., and Tanaka, T. (2010). Exploiting Semantic Information for HPSG Parse Selection. *Research on Language and Computation.* Online publication date: 20-Oct-2010.

Koeva, S. (2010). Bulgarian Wordnet – Current State, Applications and Prospects. In *Bulgarian-American Dialogues,* pages 120-132. Sofia: Prof. M. Drinov Academic Publishing House.

Cocke, J. and Schwartz, J. T. (1970). *Programming Languages and Their Compilers: Preliminary Notes. Technical report.* Courant Institute of Mathematical Sciences, New York University.

Kasami, T. (1965). *An Efficient Recognition and Syntax-analysis Algorithm for Context-free Languages*. Scientific report AFCRL-65-758. Bedford, MA: Air Force Cambridge Research Lab.

Younger, D. H. (1967). Recognition and Parsing of Context-free Languages in Time n3. *Information and Control,* 10 (2):189–208.