

PBML



The Prague Bulletin of Mathematical Linguistics

NUMBER 94 SEPTEMBER 2010

EDITORIAL BOARD

Editor-in-Chief

Eva Hajičová

Editorial staff

Ondřej Bojar
Eduard Bejček
Martin Popel
Pavel Schlesinger
Pavel Straňák

Editorial board

Nicoletta Calzolari, Pisa
Walther von Hahn, Hamburg
Jan Hajič, Prague
Eva Hajičová, Prague
Erhard Hinrichs, Tübingen
Aravind Joshi, Philadelphia
Philipp Koehn, Edinburgh
Jaroslav Peregrin, Prague
Patrice Pognan, Paris
Alexander Rosen, Prague
Petr Sgall, Prague
Marie Těšitelová, Prague
Hans Uszkoreit, Saarbrücken

Published twice a year by Charles University in Prague

Editorial office and subscription inquiries:

ÚFAL MFF UK, Malostranské náměstí 25, 118 00, Prague 1, Czech Republic

E-mail: pbml@ufal.mff.cuni.cz

ISSN 0032-6585

PBML



The Prague Bulletin of Mathematical Linguistics

NUMBER 94 SEPTEMBER 2010

CONTENTS

Editorial 5

Articles

Dependency Parsing as a Sequence Labeling Task 7

Drahomíra "johanka" Spoustová, Miroslav Spousta

**Identifying Different Meanings of a Chinese Morpheme through Semantic
Pattern Matching in Augmented Minimum Spanning Trees** 15

Bruno Galmar, Jenn-Yeu Chen

**Pragmatically-Motivated Utterance Fine-Tuning in Human-Computer
Dialogue** 35

Vladimir Popescu, Jean Caelen, Corneliu Burileanu

MT Server Land: An Open-Source MT Architecture 57

Christian Federmann, Andreas Eisele

CorporaI: a Method and Tool for Handling Overlapping Parallel Corpora 67

Mark Fishel, Heiki-Jaan Kaalep

**Asiya: An Open Toolkit for Automatic Machine Translation
(Meta-)Evaluation** 77

Jesús Giménez, Lluís Màrquez

An Experimental Management System 87

Philipp Koehn

A Toolkit for Visualizing the Coherence of Tree-based Reordering with Word-Alignments	97
<i>Gideon Maillette de Buy Wenniger, Maxim Khalilov, Khalil Sima'an</i>	
Instructions for Authors	107

PBML



The Prague Bulletin of Mathematical Linguistics
NUMBER 94 SEPTEMBER 2010

EDITORIAL

The present issue of The Prague Bulletin of Mathematical Linguistics continues in the fruitful tradition started with the special issues of PBML 91 (2009) and 93 (2010) to publish papers accepted for presentation at the regular Machine Translation Marathon events organized by the EuroMatrix and EuroMatrixPlus projects. Not to postpone the other already reviewed and accepted submissions to PBML, we publish in the current issue 5 papers (the contributions by Christian Federmann, Andreas Eisele, Mark Fishel, Heiki-Jaan Kaalep, Jesús Giménez, Lluís Màrquez, Philipp Koehn and Gideon Maillette de Buy Wenniger et al.). The rest of accepted contributions to Marathon 2010 held in Le Mans will be published in PBML 95, to be out by May 2011.

We are most grateful to the group of reviewers of the Marathon event who in a very restricted period of time have presented their highly appreciated comments to the submitted papers and especially to Philipp Koehn, Loïc Barrault and Ondřej Bojar who have taken over most of the editorial work with the Marathon part of this issue.

Eva Hajičová
Editor-in-Chief
hajicova@ufal.mff.cuni.cz



The Prague Bulletin of Mathematical Linguistics
NUMBER 94 SEPTEMBER 2010 7-14

Dependency Parsing as a Sequence Labeling Task

Drahomíra "johanka" Spoustová, Miroslav Spousta

Institute of Formal and Applied Linguistics, Charles University in Prague

Abstract

The aim of this paper is to explore the feasibility of solving the dependency parsing problem using sequence labeling tools. We introduce an algorithm to transform a dependency tree into a tag sequence suitable for a sequence labeling algorithm and evaluate several parameter settings on the standard treebank data. We focus mainly on Czech, as a high-inflective free-word-order language, which is not so easy to parse using traditional techniques, but we also test our approach on English for comparison.

1. Introduction

Dependency parsing became very popular in the recent years and many different algorithms were suggested and evaluated for this task.

Dependency structure is a rooted tree where each node (except the root) corresponds to one word of the underlying sentence. The dependency relation between two nodes (parent and child) is captured by an edge between these two nodes. The actual type of the relation is given as a function label of the edge.

The aim of a dependency parser is to assign correct parent node index to each child node, optionally also with the dependency relationship label. The standard method of evaluating dependency parser accuracy is computing the percentage of children that got the correct parent index (and optionally the dependency relationship label), among all words in a test data set.

Sequence labeling is a process where each token in a sentence is assigned a label from a fixed set. Common examples include Part-of-Speech (POS) tagging, Named

entity recognition and Semantic Role Labeling tasks, where we label every word with a tag. There are many well-studied algorithms that proved to be successful, many of them based on the Hidden Markov models and the Viterbi algorithm, such as Conditional Random Fields (Lafferty et al., 2001) or Averaged Perceptron (Collins, 2002).

In the area of parsing, the sequence labeling techniques were applied mainly to the NP-chunking (shallow parsing) task.

In the following sections, we would like to explore the possibility to turn the dependency parsing into a sequence labeling task.

Our approach is somewhat similar to LTAG supertagging introduced by (Bangalore and Joshi, 1999), but their approach has been deeply explored mainly for English and cannot be directly applied to free-word-order non-projective languages with rich inflection, like Czech.

The paper is organized as follows: Section 2 introduces the data used for our experiments. In section 3, we propose a tagset and conversion algorithm for the sequence labeling task and evaluate the performance of selected settings of the proposed algorithm on the English and Czech data sets. Section 4 presents current results and concludes.

2. The data

In order to demonstrate that our approach is not restricted to a specific language, we performed our data conversion experiments on two languages with completely different morphological characteristics and tagsets, English and Czech.

For English, we used CoNLL Shared Task 2009 data, which is a dependency representation of (a part of) the Penn Treebank (Marcus et al., 1994). We used the columns PLEMMA, PPOS (automatically assigned part-of-speech tag and morphological lemma), HEAD and DEPREL (manually annotated labeled dependency relationship). For Czech, we selected the corresponding data (i.e. manual annotation of the dependency relationships and automatic POS tagging) from the Prague Dependency Treebank 2.0 (Hajič et al., 2006), analytical layer. The Czech morphological tagset is described in (Hajič, 2004).

Main characteristics of the data¹ can be found in Table 1.

3. The data conversion algorithm

Standard dependency tree representation assigns a parent node index to every word in a sentence. The process of turning dependency parsing into a sequence la-

¹All experiments were performed using only the Train and the Development data sets. We save the Evaluation data set for the final evaluation of the (hypothetical) parser.

Table 1. The data characteristics

data set	tokens	sentences	tagset size
English train	958,167	39,278	46
English dev	33,368	1,335	45
Czech train	1,172,299	68,562	1330
Czech dev	158,962	9,270	1041

Table 2. The example sentence

	FORM	PLEMMA	PPOS	HEAD	DEPREL	DEPTAG
1	The	the	DT	4	NMOD	NN_R3 NMOD
2	luxury	luxury	NN	3	NMOD	NN_R1 NMOD
3	auto	auto	NN	4	NMOD	NN_R1 NMOD
4	maker	maker	NN	7	SBJ	VBD_R1 SBJ
5	last	last	JJ	6	NMOD	NN_R1 NMOD
6	year	year	NN	7	TMP	VBD_R1 TMP
7	sold	sell	VBD	0	ROOT	00_00 ROOT
8	1,214	1,214	CD	9	NMOD	NNS_R1 NMOD
9	cars	car	NNS	7	OBJ	VBD_L1 OBJ
10	in	in	IN	7	LOC	VBD_L1 LOC
11	the	the	DT	12	NMOD	NNP_R1 NMOD
12	U.S.	u.s.	NNP	10	PMOD	IN_L1 PMOD

being task must turn this “relative” information into an absolute label, which is to be assigned to the words. Let us call this label *deptag*².

We designed the *deptag* to contain the following parts:

1. parent node’s POS tag
2. additional information needed to decide between more possible parent-candidates with the same POS tag: direction and distance³.
3. The DEPREL label is added to the *deptag* (no transformation is needed here) to be also predictable by the sequence labeling algorithm.

An example sentence transformation can be found in Table 2.

²Unfortunately, the word *supertag* was already taken in (Bangalore and Joshi, 1999).

³We use L (R) for left (right) direction, respectively, and a positive number as a distance marker. E.g. VB_R2 means “second VB token on the right” (i. e. the first VB candidate on the right is omitted)

3.1. Tagset reduction

Naturally, a too large *deftagset* (tagset of the *deftags*) leads to the data sparseness problem. It turns out that the size of the tagset may be reduced by adjusting two parameters of the algorithm:

1. Considering only a fragment of the full POS tag during the *deftag* construction. This is mainly useful for a rich POS-tag set language (such as Czech with more than 4000 possible POS-tags and only 1330 of them appearing in the training data) and is much less important for English (with only 46 tags). We tried to reduce the 15-positions tagset to 5 or 2 most important positions for Czech⁴. At this point, the assigned *deftag* corresponds to exactly one parent node. In other words, the transformation from the tree into a *deftag* sequence remains lossless.
2. Constraining maximum *distance* to reduce the *deftagset*. Here the *distance* does not mean the absolute number of tokens between parent and its child, but the maximum number of possible candidates with the same POS tag to be distinguished. For example, if the maximum is set to 3, then all third, fourth and further candidates will be labeled with 3 and translated back to the third candidate.

This constraint, however, makes the transformations lossy.

Let us define the *transformation error rate* of the selected algorithm configuration as following: For each node n let p_n^1 be the parent node index assigned to the node n in the source dependency tree, D_n a *deftag*, to which p_n^1 is translated using the selected algorithm configuration, and p_n^2 a parent node index, to which the the *deftag* D_n is translated back (the process of translating back is unambiguous). *Transformation error rate* is the percentage of nodes, where $p_n^2 \neq p_n^1$.

Table 3 shows the *deftagset* size (labeled, unlabeled) and transformation error rates for a few selected configurations. Considering usual performance of dependency parsers lying between 80 % and 90 %, setting the *maximum distance* to 3 introduces only a small decrease of performance (lower than 1% transformation error rate).

3.2. The list of possibilities

Usually, the sequence labeling algorithm chooses one label from a list of plausible labels for every token. There are several methods how to generate such a list. We can compare the methods for generating the list of possible *deftags* using a standard measure: precision and recall.

⁴5 — Part of speech, Detailed part of speech, Gender, Number, Case; 2 (originating from (Collins et al., 1999)) — first letter is the main POS, second letter is the Case field if the main POS is the one that displays case, while otherwise the second letter is the detailed POS.

Table 3. deptagset size (labeled, unlabeled) and transformation error rate. (POS - Part-of-speech tag size, max - maximum distance.)

language	POS	max. distance	tagset size		error
			labeled	unlabeled	
en	full	-	1523	244	0.00
en	full	3	1305	159	0.52
en	full	1	809	78	7.56
cz	full	-	10449	1979	0.00
cz	full	3	9949	1882	0.30
cz	full	1	8679	1601	2.29
cz	5	-	7820	1319	0.00
cz	5	3	7299	1214	0.30
cz	5	1	6016	958	2.45
cz	2	-	3760	389	0.00
cz	2	3	3127	244	0.36
cz	2	1	2054	122	3.32

In order to enable sequence labeling algorithm to work well on our transformed data, we aim to keep recall as high as possible, while introducing a reasonable precision.

- To achieve 100% recall, we need to consider every token in the sentence to be possible parent of all other tokens. I.e. in a n -token (incl. root) sentence, every token has $n - 1$ possible parents, thus translated into $n - 1$ unlabeled *deptags*. Every *deptag* can (theoretically) be combined with every *deprel* label. Let *dep* be size of the set of the *deprel* labels. So the final list of possibilities contains $(n - 1) * \text{dep}$ labeled *deptags* for every token. This approach achieves 100% recall, but very poor precision.
- The list of possible combinations *deptag* + dependency relationship label (*labeled deptags*) can be simply derived from the training data. To avoid the data sparseness problem, we choose to discriminate the tokens only through their morphological tags, i.e. to ignore their form and lemma. For example, if a token has the *NNP* POS tag, we add to its list of possible *labeled deptags* manually annotated parents (translated into the *labeled deptags*) of all *NNP* tokens found in the training data. This approach achieves 100% recall only for the training data. Its precision is much better than the precision of previous approach, but still remains low.
- Precision of the previous approach can be increased by restricting the maximum "length of relationship" (in absolute number of tokens) between parent and child. We omit every possible relationship which is "longer" (in terms of

Table 4. Generating a labeled list of possibilities; recall2 = recall including conversion back to the tree representation

basic options	list options	precision	recall	recall2
en full -	full	0.06	100	100
en full 3	full	0.06	100	99.48
en full 3	train	1.42	99.77	99.26
en full 3	train+length	2.45	99.43	98.93
cz 5 3	full	0.05	100	99.70
cz 5 3	train	1.28	98.21	97.93
cz 5 3	train+length	2.28	96.86	96.58
cz 2 3	full	0.05	100	99.64
cz 2 3	train	0.82	99.53	99.18
cz 2 3	train+length	1.46	99.01	98.67

absolute value of the subtraction between the nodes indexes) than the longest relationship seen in the training data between nodes with the same POS tags as the ones of the considered tokens.

Precision and recall of selected variants of the approaches described above can be found in Tables 4 (labeled) and 5 (unlabeled), as measured on the development data set.

4. Results and Conclusion

We have shown that our data conversion algorithm can fully represent a labeled dependency tree, both for a rich morphology language with large tagset and for a language with very small tagset.

The parameters of the conversion can be theoretically set in the manner that keeps the conversion absolutely lossless. We have proposed various kinds of (slightly lossy) reductions of the solution space.

As we have proposed in the introduction section, this article focuses on the data conversion algorithm, i.e. data preparation for the sequence labeling algorithm.

As a proof-of-concept and possible baseline, we have processed the converted data with the averaged perceptron algorithm (Collins, 2002) with trivial trigram feature set, the results (accuracy of the final labeled tree) varied (depending on configuration) between 5-10 % below state-of-the-art (according to CoNLL Shared Task 2009 results and the <http://ufal.mff.cuni.cz/czech-parsing> page). Additional up to about 1 % will be loosed if we use as a postprocessing some sort of greedy algorithm which will fix the cycles to ensure the result is tree.

Our approach can thus be used as a simple and fast way to build an "approximative" parser in case there is no better solution available (e.g. due to license restrictions).

Table 5. Generating an unlabeled list of possibilities; recall2 = recall including conversion back to the tree representation

basic options	list options	precision	recall	recall2
en full -	full	3.24	100	100
en full 3	full	3.52	100	99.48
en full 3	train	4.80	99.92	99.39
en full 3	train+length	6.68	99.75	99.24
cz 5 3	full	4.17	100	99.70
cz 5 3	train	6.52	99.08	98.79
cz 5 3	train+length	9.76	98.20	97.91
cz 2 3	full	4.27	100	99.64
cz 2 3	train	5.18	99.91	99.55
cz 2 3	train+length	7.26	99.67	99.32

The performance can be further improved by selection of the sequence labeling algorithm and its configuration (such as feature set selection).

Acknowledgments

The research described here was supported by the project GA405/09/0278 of the Grant Agency of the Czech Republic.

Bibliography

- Bangalore, Srinivas and Aravind K. Joshi. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25:237–265, 1999.
- Collins, Michael. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, volume 10, pages 1–8, Philadelphia, PA, 2002.
- Collins, Michael, Jan Hajič, Lance Ramshaw, and Christoph Tillmann. A Statistical Parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 505–512, College Park, Maryland, USA, June 1999. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P99-1065>.
- Hajič, Jan. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*. Nakladatelství Karolinum, Prague, 2004. ISBN 80-246-0282-2.
- Hajič, Jan, Eva Hajičová, Jarmila Panevová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, and Marie Mikulová. Prague Dependency Treebank v2.0, CDROM, LDC Cat. No. LDC2006T01. Linguistic Data Consortium, Philadelphia, PA, 2006.

- Lafferty, John, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, USA, 2001.
- Marcus, Mitchell P., Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1994. ISSN 0891-2017.

Address for correspondence:

Drahomíra "johanka" Spoustová
johanka@ucw.cz
Institute of Formal and Applied Linguistics
Charles University in Prague
Malostranské náměstí 25
118 00 Praha 1, Czech Republic



The Prague Bulletin of Mathematical Linguistics
NUMBER 94 SEPTEMBER 2010 15-34

Identifying Different Meanings of a Chinese Morpheme through Semantic Pattern Matching in Augmented Minimum Spanning Trees

Bruno Galmar^a, Jenn-Yeu Chen^b

^a National Cheng Kung University, Institute of Education

^b National Cheng Kung University, Institute of Cognitive Science

Abstract

Galmar and Chen (2010) introduced the first corpus-based computational approach to the problem of identifying the different meanings of a polysemous Chinese morpheme embedded in polymorphemic words. The approach is based on the successive application of a dimensionality reduction method - Latent Semantic Analysis, a graph-theory algorithm - Prim's algorithm and a semantic pattern recognition search used for meaning inference. Our present work adds major changes and contributions to Galmar and Chen (2010). Firstly, we theoretically defined and detailed what are the Chinese semantic patterns to be searched in the augmented minimum spanning trees. Then, we modified the computational approach to include the use of Nearest Neighbors lists. This change allows for a major contribution: a proof is given that all the possible outputs of Prim's algorithm in our experiment - the minimum spanning trees - contain all the same amount of semantic information to be used for meaning inference. Thus, our final meaning inference results are optimal.

Practically, this work could serve as a first step to add a new feature to current Chinese Wordnets: the listing of all the Chinese words embedding a certain polysemous morpheme with a fixed identified meaning. Finally, future directions are sketched.

KEYWORDS: Chinese Polysemous Morphemes, Meaning Inference, Dimensionality Reduction, Graph-Theory Analysis, Semantic Patterns.

1. Introduction

The character 公 is a Chinese polysemous morpheme with more than sixteen dimensions of meaning according to an etymological dictionary¹. The Chinese words 公鹿 (male deer), 外公 (maternal grandfather) and 公平 (fair) are polymorphemic Chinese words that all embed 公 as a common morpheme. However, for a Chinese native speaker, the meaning of 公 in each of the three words is different:

1. In 公鹿 (male deer), the meaning of 公 is male.
2. In 外公 (maternal grandfather), the meaning of 公 is grandfather.
3. In 公平 (fair), the meaning of 公 is fair.

In the three above examples, the meaning of 公 is defined each time by only one word: “male”, “grandfather” and “fair”. These one-word definitions have a psychological reality for the Chinese native speaker. An interesting fact is that these one-word definitions exist as the etymological dimensions of meaning of 公² given in an etymology dictionary of Chinese. Each etymological dimension of meaning can be represented by a Chinese word – which we call a meaning dimension word. Thereafter, we retain the linguistically simplified definition of the meaning of 公 in a polymorphemic word as being one meaning dimension word.

In (Galmar and Chen, 2010) and in the present study, the proposed computational approaches aim at identifying the meaning of 公 in a polymorphemic 公 word using knowledge of the etymological dimensions of meaning for the Chinese monomorphemic word 公. Basically, the approaches consist in detecting in a data structure - here a minimum spanning tree - some semantic patterns which are instances of pre-defined abstract semantic patterns. These patterns can be interpreted through pre-defined meaning inference rules to identify which meaning dimension word can serve as the meaning of 公 in a polymorphemic 公 word captured in a semantic pattern. So, both (Galmar and Chen, 2010) approach and ours can be viewed as light knowledge-based approaches. We used the adjective “light” to emphasize that the amount of knowledge fed to the meaning inference system is kept as low as possible: a dozen of meaning dimensions words and a few meaning inference rules. Our approach can also discover other potential meaning dimensions words and can eventually update the initial list of meaning dimensions words. Galmar and Chen (2010)’s work and our present study are both generalizable to all Chinese polysemous monomorphemic words. Galmar and Chen (2010) focused on 公 polymorphemic words because experimental data from a categorization label task experiment are available for the 公 words.

¹ source: <http://www.chineseetymology.org/www.chineseetymology.org/> Some dimensions are overlapping.

²The dimensions of meaning for 公 are: unselfish / unbiased / fair / to make public / open to all / public / the first of old China’s five-grades of the nobility / an old Chinese official rank / the father of one’s husband (one’s husband’s father) / one’s father-in-law / one’s grandfather / a respectful salutation / the male (of animals) / office / official duties / a Chinese family name

Galmar and Chen (2010) observed that there is currently no Chinese dictionary or database which lists for each meaning of a polysemous morpheme all the Chinese words embedding the morpheme with this meaning. For example, the Chinese Wordnet of the Academia Sinica³ offers a list of some of the different meanings of 公 but provides no listing of all the 公 words with a same given meaning of 公 e.g. “fair”.

Galmar and Chen (2010) reviewed the literature on Chinese computational morphology and Chinese word sense disambiguation. They found no prior work proposing a computational approach to the task of meaning identification of a polysemous morpheme embedded in Chinese words. Therefore, Galmar and Chen (2010) proposed the first corpus-based computational approach to this task. In an example, they showed how the approach can lead to extract correctly the meaning of 公 in 公鹿 (Galmar and Chen, 2010).

Galmar and Chen (2010)’s work suffers from the following main limitations:

1. The authors found that there was not a unique minimum spanning tree to serve as a solution. However, only one solution was presented without indication about its selection. The authors gave neither information about possible criteria to use to select the best solution, neither a proof that the presented solution was the best one.
2. The concept of augmented minimum spanning trees did appear briefly but was not detailed and emphasized.
3. Some primary considerations for future directions were absent.

The present work adds the following major contributions and changes to (Galmar and Chen, 2010):

1. The novel computational approach has been designed in a way that allows to prove its optimality. Now, computing the Nearest Neighbors lists plays a major role in helping to prove optimality of the results. We described the novelties of our approach carefully.
2. A new way of formalizing Chinese free morphemes and words interaction for semantic inference is presented for the first time through the concepts of semantic pattern matching and semantically augmented minimum spanning trees.
3. Proofs that our results are optimal are given using solid reasoning based on the work of Prim in graph-theory (Prim, 1957).
4. New detailed examples of meaning inference for a Chinese polysemous morpheme are given to illustrate the results.
5. Future enhancements, directions and generalizations of this work are given.

This work is intended to serve as a first step in:

1. Designing tools for Chinese cognitive scientists and linguists who study the semantic interaction between Chinese morphemes and polymorphemic words. It can help in preparing experimental materials for lexical decision tasks and relatedness judgment tasks involving the repetition of a same Chinese polyse-

³<http://cwn.ling.sinica.edu.tw/>

mous morpheme embedded with a fixed identified meaning in different Chinese words (Chen et al., 2009; Galmar and Chen, 2007).

2. Enhancing Chinese Wordnets with the listing for each meaning of a polysemous morpheme of all the Chinese words embedding the morpheme with this meaning.

The new computational approach to infer the meaning of 公 in polymorphemic words can be unfolded in six steps:

1. The first step encompasses the creation of semantic spaces and their dimensionality reduction. We built three nested lists of words. The Academia Sinica Balanced Corpus (ASBC) was then filtered by these three lists to output three term-document matrices (TDM). The matrices were weighted and then their dimensionality was reduced through the computation of the reduced Singular Value Decomposition (SVD). The final matrices represent the three nested Latent Semantic Analysis (LSA) semantic spaces.
2. The second step consists in computing for each LSA semantic space the cosine matrix and the dissimilarity matrix for all terms.
3. In the third step, each dissimilarity matrix is viewed as the adjacency matrix of a complete weighted undirected graph and is used to build a minimum spanning tree (MST) by applying Prim's algorithm.
4. Besides building the MST, a nearest neighbors (NN) list is built for each graph. The NN list serves to augment the MST with neighboring information.
5. In the fifth step, it is proven that even if the outputted MSTs are not unique, they embed the maximum amount of information useful for meaning inference.
6. In the last step, the paths of the MSTs are browsed in search of patterns to extract the meaning of 公 in the polymorphemic 公 words.

The remainder of this paper is organized as follows. In Section 2, firstly, we newly put forward which particular features of the Chinese language and the ASBC corpus are at the heart of the working of the computational approach. Then, we present briefly Steps 1 and 2 which are fully described in (Galmar and Chen, 2010). Section 3 encompasses Steps 3, 4 and 5. For Step 3, we emphasize the rationale of reducing the completeness of the dissimilarity matrix to obtain a MST. Step 4 and 5 are new steps. In Section 3, we also define and list the semantic patterns to be used for meaning inference. In section 4, the application of Step 6 is illustrated through new results. Then come the conclusion and the future directions sections.

2. The Nested Semantic Latent Semantic Analysis Spaces and Their Dimensionality Reduction

2.1. Philosophy of the Computational Approach and Creation of the Semantic Spaces

At the heart of the design of the present computational approach are the following peculiarities of the Chinese language and the ASBC corpus:

1. Some Chinese characters are free morphemes: they are not only embedded in compound words but can also be standalone words (Packard, 2000). 公 is such a free morpheme and is also polysemous.
2. In the ASBC corpus, 公 occurs as a stand-alone monomorphemic word with more than one part-of-speech (POS) tag: it has five different POS tags⁴.

The plurality of the occurrence of 公 with different POS tags is thought to help the capture of the meaning of 公 in 公 polymorphemic words. We imagined that the 公 monomorphemic words could be captured in a structure - e.g. a graph - with other polymorphemic 公 words forming together semantic patterns. These semantic patterns could be used to infer some different meanings of the polysemous 公 morpheme.

To employ a metaphor inspired by the software hacking culture, we viewed polysemous free morphemes like 公 as a potential backdoor to the semantics of Chinese polymorphemic words. Hence, the work is an attempt to crack the semantics of the Chinese language using some peculiarities of the Chinese language and of the annotated ASBC corpus.

There is also a flavor of whole-part thinking in our approach: the part - the free morpheme - and the whole - the compound word - are both necessary to conduct semantic inference about the identification of the meaning of a Chinese polysemous morpheme.

2.2. The Building of Nested Semantic Latent Semantic Analysis Spaces

Galmar and Chen (2010) built three nested lists of 公 words extracted from the 5 million words Academia Sinica Balanced Corpus (ASBC). Full details about how were built these three lists is given in Galmar and Chen (2010). These three lists are used to filter the ASBC corpus to obtain three term-document matrices (TDM). The three TDM will serve to build the three semantic LSA spaces by application of Latent Semantic Analysis.

The main idea is to create three semantic spaces of increasing semantic richness:

1. The smallest semantic space contains only 公 words. This space is thought to be the poorest representation of the semantic relationships between the 192 公 words. The 5 公 words and 187 additional polymorphemic 公 words constitute the initial list of 192 公 words under study. The TDM was made of 192 words and 3716 documents.
2. The second semantic LSA space contains words that represent ten etymological dimensions of the meaning of 公. These meaning dimension words could attract or be attracted by semantically similar 公 words in semantic patterns in a graph structure. From a cluster-based viewpoint, these words could serve as centroids of 公 words clusters. They could also be used later to infer the meaning of 公 in 公 words. The TDM was made of 202 words and 4327 documents. The twelve

⁴“ 公 (Vh)”, “ 公 (Nb)”, “ 公 (Nc)”, “ 公 (Na)” and “ 公 (A)”

meaning dimension words are: (公正, 公平, 公開, 公共, 無私, 貴族, 爵位, 父, 岳父, 雄性, 機關, 機構)⁵.

3. The third and biggest semantic space is thought to be semantically rich enough to embed meaningful semantic relationships between the words it contains. Such a micro-size space could be an alternative start to a whole corpus semantic space to investigate the different meanings of 公 in 公 words. The TDM was made of 283 words and 6798 documents. Among the additional words which were inserted in the third list, there were:
 - (a) words which are key-words occurring in a Chinese dictionary's definitions of some of the 187 polymorphemic 公 words.
 - (b) non-公 compound words that share some common morphemes with the 187 公 compound words.
 - (c) a few words (e.g. 國家 (country)) which have been chosen as category labels by two Taiwanese participants in a pilot study of the subjective sorting of the 187 polymorphemic 公 words.

From a language acquisition standpoint, our approach is thought to bear the two following realistic traits: both the size of the lexicon and its semantic richness increase - as they do during language learning -.

2.2.1. The Three Weighting Schemes

To each of the three term-document matrices, Galmar and Chen (2010) applied a total of three weighting schemes:

1. A local weighting scheme. The TDM containing the term frequencies m_{ij} is logarithmised. Hence, the effect of frequency differences between terms in a same document is reduced.
2. A global weighting scheme. The classic Inverse Document Frequency scheme (Landauer and Dumais, 1997; Landauer et al., 2007) is used. It gives more weight to words with a global low frequency.
3. At the document level - the columns of the term-document matrix - a weighting scheme is applied. To reduce the effect of the size difference between documents, each column of the term-document matrix is multiplied by:

$$\log_2 \left(\frac{\text{Max document size}}{\text{Document size}} + 1 \right) . \quad (1)$$

More weight is given to small documents. Such a choice is motivated by the heuristic that especially for news articles - the documents of the ASBC corpus -, it is easier to extract the gist of a short article than a very long one: it means that small documents are generally better informative than long ones about their

⁵In the twelve words list, the first four words are 公 words already present in the first semantic space. These twelve words capture ten relatively different dimensions of meaning of 公. Two of the words capture the same dimension meaning. See (Galmar and Chen, 2010) for more details.

inherent meaning⁶. This document level weighting scheme is preferred to resizing the corpus's meaning unit from the original entire document to paragraph of a given size. Resizing could result in splitting meaningful units in different documents.

2.2.2. Singular Value Decomposition and Reduced SVD

After being weighted, the three TDM have their reduced Singular Value Decomposition (rSVD) computed. Galmar and Chen (2010) explained why they empirically decided to reduce the dimensionality of the LSA spaces by taking into account only the first one hundred singular values. This rSVD can be written:

$$A \simeq A_{100} = U_{100} \Sigma_{100} V_{100}^T \quad (2)$$

where A is the original TDM, A_{100} the reduced TDM, U_{100} and V_{100} two orthogonal matrices⁷ and where $\Sigma_{100} = \text{diag}(\sigma_1, \dots, \sigma_{100})$ and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{100} > 0$ are the 100 first non-zeros singular values.

Galmar and Chen (2010) termed $A_{192,100}$, $A_{202,100}$ and $A_{283,100}$ the three reduced LSA spaces containing respectively 192, 202 and 283 words. $A_{192,100}$, $A_{202,100}$ and $A_{283,100}$ are the Chinese LSA spaces that will be used to compute the cosine and dissimilarity matrices.

2.2.3. Cosine Matrix and Dissimilarity matrix

In $A_{192,100}$, $A_{202,100}$ and $A_{283,100}$, words are represented as vectors. Similarity between two words of a Chinese LSA space is measured by calculation of the cosine value between the two corresponding vectors. For each of the three Chinese LSA spaces, Galmar and Chen (2010) built the whole cosine matrix C defined by $c_{ij} = \cos(v_i, v_j)$, where i and j are two words of a Chinese LSA space and where v_i and v_j their representing vectors. C is symmetric due to $\cos(v_i, v_j) = \cos(v_j, v_i)$. The three cosine matrices C_{192} , C_{202} and C_{283} were computed.

From the cosine matrix C , the dissimilarity matrix D is derived with:

$$d_{ij} = 1 - c_{ij} = 1 - \cos(v_i, v_j) \geq 0 \quad (3)$$

The three dissimilarity matrices D_{192} , D_{202} and D_{283} whose dimensions are respectively 192*192, 202*202 and 283*283 were computed.

⁶An illustrative comparison would be that of politicians' speeches. Politicians are sometimes criticized to drown the meaning of their speech in their length. After listening to a long politician's speech one could have the feeling of having lost its gist. Politicians could not achieve to induce this feeling with a very short speech while attention of listeners is remaining high.

⁷ U_{100} and V_{100} are the truncated matrices of the orthogonal matrices occurring in the SVD of A .

3. The Graph-Theory Based Approach

3.1. A Few Definitions

Galmar and Chen (2010) introduced all the basic concepts from graph theory which will be used thereafter. Here, we just define again the most crucial concepts and add two definitions.

The *adjacency matrix* A of a complete weighted graph G is the matrix whose entry A_{ij} is 0 if $i = j$ and otherwise is w_{ij} the weight assigned to the edge $\{V_i, V_j\}$ (Gross and Yellen, 2006).

A *nearest neighbor (NN)* V_{NN} of a vertex V_i is a vertex for which the weight of the edge $\{V_{NN}, V_i\}$ is minimum among all the edges joining V_i . A *second nearest neighbor (SNN)* V_{SNN} of a vertex V_i is the second vertex of smallest weight and follows V_{NN} in the NNs list. Two vertices V_i and V_j are said to be *reciprocal nearest neighbors (RNN)* if V_i is the nearest neighbor of V_j and vice versa. In the same way, two vertices V_i and V_j are said to be *secondary reciprocal nearest neighbors (SRNN)* if V_i is the second nearest neighbor of V_j and vice versa.

A *spanning tree (ST)* of a graph G is a tree of G which contains all the vertices of G .

A *minimum spanning tree (MST)* of a graph G is a spanning tree (ST) of G whose the sum of edges is minimum (Foulds, 1995). This can be written:

$$\sum_{e \in MST} w(e) = \min_{ST \in G} \left(\sum_{e \in ST} w(e) \right) . \quad (4)$$

In a *short-path tree (SPT)*, the length of the paths between the root vertex and all the other vertices are minimum.

3.2. Applying Graph Theory to the Dissimilarity Matrix

3.2.1. Building the Adjacency Matrix.

Galmar and Chen (2010) viewed the dissimilarity matrix D defined in §2.2.3 as the adjacency matrix of a complete weighted undirected graph G . Each word of a given Chinese LSA space is a vertex of G and each edge of G linking two vertices v_i and v_j is weighted by d_{ij} . Thus we have:

$$\forall i, d_{ii} = 0 \text{ and } \forall (i, j) \text{ with } i \neq j, d_{ij} = 1 - \cos(v_i, v_j) . \quad (5)$$

3.2.2. Building the Nearest Neighbors Lists.

From each of the dissimilarity matrices D_{192} , D_{202} and D_{283} , we computed the nearest neighbors lists NN_{192} , NN_{202} and NN_{283} . The i -th member of the list NN_X is the NN of the i -th member of the original list of X $\hat{\sphericalangle}$ words.

3.2.3. Building One Minimum Spanning Tree.

From D_{192} , D_{202} and D_{283} we can directly build three minimum spanning trees MST_{192} , MST_{202} and MST_{283} using Prim's algorithm (Prim, 1957; Graham and Hell, 1985). Hence, each of our semantic LSA space $A_{192,100}$, $A_{202,100}$ and $A_{283,100}$ has an associated minimum spanning tree.

In (Prim, 1957), Prim enunciated two construction principles for building a MST of an undirected weighted graph G :

1. Any unconnected vertex can be directly connected to a nearest neighbor (P1).
2. Any unconnected sub-MST⁸ can be connected to a nearest neighbor by a shortest available path (P2).

Prim went on by validating these two construction principles by proving the two following necessary conditions (NC1 and NC2) for a MST:

1. Every vertex in a MST is directly connected to at least one nearest neighbor (NC1).
2. Every subgraph of a MST is connected to at least one nearest neighbor by a shortest available path (NC2).

In Prim's algorithm, P1 is first applied to build a first edge and then P2 is continuously applied to grow from the very first edge a subgraph that will be a MST once all the vertices of the initial graph G will have been connected.

3.2.4. Uniqueness of the Minimum Spanning Tree.

Uniqueness of the MST of a graph G is ensured if each edge of G has a different weight (Eppstein, 1995). In case of non-uniqueness of the MST, solutions provided by (Eppstein, 1992, 1995; Broder and Mayr, 1997; Wright, 1997, 2000) can be applied to count and enumerate all the MSTs.

3.2.5. Counting and Enumerating All the MSTs.

(Eppstein, 1995) proposed to build an equivalent graph EG of the graph G such that the ST of EG are the MSTs of G . The *Kirchoff's matrix-tree* theorem (de Abreu, 2007) served to compute the number of ST in EG corresponding to the number of MSTs in G . Listing all the STs of EG gives all the MSTs of G . (Wright, 1997, 2000) proposed another procedure to count the number of MSTs and build them.

3.2.6. Patterns to Be Observed in MSTs for Meaning Inference.

Once a MST is obtained for the semantic space $A_{X_{words},100}$, it will be used for meaning inference. Here, we are concerned by describing the expected patterns of vertices - to be found in the MST - that will serve for meaning inference. Each pattern is

⁸In the original paper, Prim used *isolated fragment* to refer to a subgraph of a MST. Here we use the word sub-MST.

paired with a set of assertions concerning the meaning inferences that can be formulated. Figure 1 presents such an ideal pattern. In Fig. 1, one meaning dimension word (*DimWord*) - introduced in §2.2 - shares an edge with a $\hat{\text{公}}$ monomorphemic word $\hat{\text{公}}_{\text{mono}}$. The latter shares one or more edges with $\hat{\text{公}}$ polymorphemic words. The weight of the edge $\{\hat{\text{公}}_{\text{mono}}, \text{DimWord}\}$ is assumed to be the smallest one among the edges joining $\hat{\text{公}}_{\text{mono}}$. In other words, *DimWord* is the NN of $\hat{\text{公}}_{\text{mono}}$. This is represented on Fig.1 by a left-to-right arrow from $\hat{\text{公}}_{\text{mono}}$ to *DimWord* labeled with the symbol NN. In the semantic space $A_{X_{\text{words}},100}$, for the pattern represented by Fig.1, we will assert that the primary meaning of $\hat{\text{公}}_{\text{mono}}$ is the meaning of *DimWord*. $\hat{\text{公}}_{\text{Wordtarget}}$ is a $\hat{\text{公}}$ polymorphemic word directly connected to *DimWord* and shares no other edge with other words. Therefore, in $A_{X_{\text{words}},100}$, the meaning of $\hat{\text{公}}$ in $\hat{\text{公}}_{\text{Wordtarget}}$ is the meaning of $\hat{\text{公}}_{\text{mono}}$ whose primary meaning is the meaning of *DimWord*.

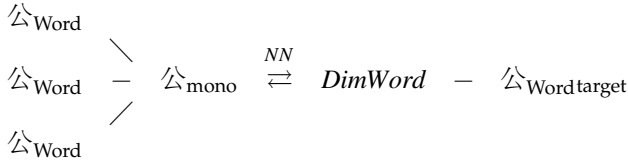


Figure 1. An ideal component of a MST for meaning inference.

The pattern presented in Fig. 2 is an extension of Fig. 1 where *DimWord* and $\hat{\text{公}}_{\text{Wordtarget}}$ are connected through a number N of non- $\hat{\text{公}}$ Word. In that case, we will also assert that in $A_{X_{\text{words}},100}$, the closest meaning to the meaning of $\hat{\text{公}}$ in $\hat{\text{公}}_{\text{Wordtarget}}$ is the meaning of $\hat{\text{公}}_{\text{mono}}$ which itself has for meaning the meaning of *DimWord*.

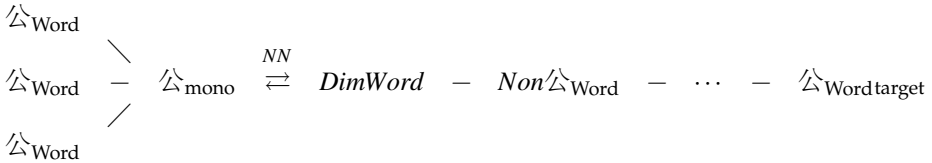


Figure 2. Another ideal component of a MST for meaning inference

In Fig. 3, *DimWord* and $\hat{\text{公}}_{\text{mono}}$ are reciprocal nearest neighbors (RNN). We will assert that the meaning of $\hat{\text{公}}_{\text{mono}}$ in $A_{X_{\text{words}},100}$ is the meaning of *DimWord*. Of all the edges joining respectively $\hat{\text{公}}_{\text{Wordtarget1}}$ and $\hat{\text{公}}_{\text{Wordtarget2}}$, the edges $\{\text{DimWord}, \hat{\text{公}}_{\text{Wordtarget1}}\}$

and $\{\hat{\text{公}}_{\text{mono}}, \hat{\text{公}}_{\text{Wordtarget2}}\}$ are assumed to be of smallest weight. This means that $\hat{\text{公}}_{\text{Wordtarget1}}$ and $\hat{\text{公}}_{\text{Wordtarget2}}$ have respectively *DimWord* and $\hat{\text{公}}_{\text{mono}}$ as NN. In this case, we will assert that in $A_{X_{\text{words},100}}$, the meaning of $\hat{\text{公}}$ in $\hat{\text{公}}_{\text{Wordtarget1}}$ and in $\hat{\text{公}}_{\text{Wordtarget2}}$ is the meaning of $\hat{\text{公}}_{\text{mono}}$ which itself has for meaning the meaning of *DimWord*.

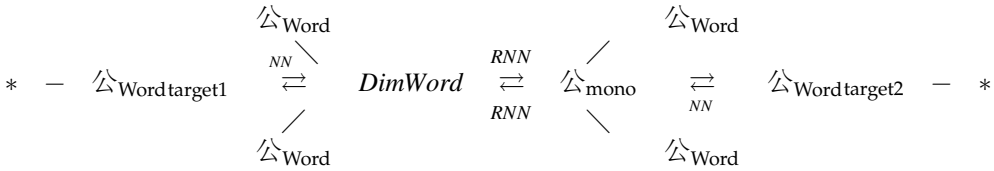


Figure 3. A more complex ideal component

The above list of presented patterns for meaning inference is not exhaustive. It was given to illustrate the logic of how meaning inference is conducted from sequences of paths in the MST.

3.2.7. Walking the MST.

Lemma 1. (Wu and Chao, 2004)

Any two vertices in a tree are connected through a unique path.

Following Lemma 1, in a MST, the path connecting two vertices is unique. The length of the path between two vertices could be measured by:

1. summing the weights of all the edges composing the path.
2. combining the precedent sum with the total number of intermediary nodes.
3. qualitatively summing the number of concepts composing the path. This requires human judgment.
4. quantitatively summing the number of concepts composing the path. This requires to feed the meaning inferring program with some additional knowledge about the relationships between the non- $\hat{\text{公}}$ words added into the third semantic space. Some of the non- $\hat{\text{公}}$ words are synonyms, antonyms or hyponyms of other non- $\hat{\text{公}}$ words. The MST does not embed originally such knowledge. Therefore the MST is to be augmented. That can be easily done by augmenting the structure of vertices in the MST for non- $\hat{\text{公}}$ words.

The last solution for path length calculation was chosen in this study. Path length can serve as an indicator of similarity between two words. This similarity will be interpreted primarily as semantic but could be situational or of other nature. The shorter the length of the path between two words, the closer is their similarity relationship.

Table 1. Nearest neighbors for the monomorphemic 公 words

Vertex	Nearest Neighbor	Weight
公 (a)	雄性 (na)	0.1832
公 (na)	公社 (nc)	0.3227
公 (nb)	公債 (na)	0.8319
公 (nc)	廉政公署 (nc)	0.4776
公 (vh)	大公國 (na)	0.2364

3.3. Rationale for Applying Graph Theory to the Dissimilarity Matrix

MSTs are thought to be the smallest data structures which connect all the words of a semantic space through paths and which embed core relationships between words. Comparisons of weight values for the different edges joining a given vertex allow to determine all kinds of nearest neighbor relationships (NN, RNN) that have been precedently showed as necessary for meaning inference.

MSTs are considered more general and balanced in information than shortest-path trees (SPT).

4. Results

4.1. Nearest Neighbors

From the three dissimilarity matrices D_{192} , D_{202} and D_{283} , we computed the nearest neighbors list of each vertex. We present only the NNs list for our biggest semantic space $A_{283,100}$.

4.1.1. The Nearest Neighbors NN_{283} of $A_{283,100}$.

Uniqueness of Nearest Neighbors. Of the 283 words, three (周公 (nb), 關公 (nb) and 雞 (na)) have two nearest neighbors. The remaining words have a single - uniquely defined - nearest neighbor.

Nearest neighbors for the monomorphemic 公 words. Table 1 shows the nearest neighbors for the monomorphemic 公 words and the associate weights. The smaller the weight value is, the closer is the relationship between the 公 word and its NN.

The nearest neighbor of 公 can be interpreted as its primary meaning in the semantic space $A_{283,100}$. At best, the NN is one of the hypothesized meaning dimension word. For example, 公 (a) has 雄性 as a nearest neighbour which is also a meaning dimension word. It can be concluded that 雄性 is the primary meaning of 公 (a). Others monomorphemic 公 words do not have a dimension word as a NN.

Table 2. The monomorphemic 公 words as nearest neighbors

Vertex	公 Nearest Neighbor	Weight
外公 (na)	公 (a)	0.9058
公社 (nc)	公 (na)	0.3227
∅	公 (nb)	∅
廉政公署 (nc)	公 (nc)	0.4776
大公國 (na)	公 (vh)	0.2364

Table 3. Nearest neighbors of the 公 meaning dimension words

Vertex	Nearest Neighbor	Weight
公正 (vh)	包公 (nb)	0.0606
公平 (vh)	公交法 (na)	0.0327
公開 (vhc)	公益金 (na)	0.0173
公共 (a)	公用 (a)	0.5621
無私 (vh)	公害 (na)	0.2964
貴族 (na)	公爵 (na)	0.0768
爵位 (na)	地位 (na)	0.0424
雄性 (na)	雌性 (na)	0.0672
父 (na)	子 (nb)	0.3519
岳父 (na)	公秉 (nf)	0.2454
機構 (na)	公衛 (na)	0.1886
機關 (na)	公司法 (na)	0.6194

The monomorphemic 公 words as nearest neighbors. Monomorphemic 公 words attracted 公 words and even became their NNs. No word has 公 (nb) as a NN.

From Tab. 1 and Tab. 2, we noticed the existence of *reciprocal nearest neighbors* relationships (RNN). The NN of 公 (na) is 公社 (nc) and vice-versa. {公 (nc), 廉政公署 (nc)} are RNN and {公 (vh), 大公國 (na)} too.

Nearest neighbors of the 12 meaning dimension words. Table 3 presents the nearest neighbors of the 12 meaning dimension words. Except for 雄性, 爵位 and 父, the dimension words have 公 words as NN.

Table 4. Frequencies of the 公 meaning dimension words as nearest neighbors

Dimension Word	Frequencies as a NN	Source Nodes	Weight
公正 (vh)	1	包公 (nb)	0.0606
公平 (vh)	2	公交法 (na), 公平性 (na)	0.0327, 0.2287
公開 (vhc)	1	公益金 (na)	0.0173
公共 (a)	0		
無私 (vh)	0		
貴族 (na)	2	公爵 (na), 小雞 (na)	0.0768, 0.2430
爵位 (na)	2	公子 (na), 地位 (na)	0.0907, 0.0424
父 (na)	0		
岳父 (na)	1	女兒 (na)	0.6103
雄性 (na)	1	公 (a)	0.1832
機構 (na)	3	公信力 (na), 公衛 (na) 提起公訴 (vb)	0.6165, 0.1886 0.2226
機關 (na)	0		

Frequencies of the meaning dimension words as nearest neighbors. Table 4 shows how many times the dimension words have served as a nearest neighbor. From Tab. 3 and Tab. 4, we observed that the meaning dimension word 公正 and the 公-word 包公 are RNN. Therefore in $A_{283,100}$, the meaning of 公 in 包公 is 公正 (just, fair). Actually, 包公 is the name of an Ancient China high official who symbolizes justice. Chinese speakers will associate differently the meaning of 公 in 包公 to the noun judge or lord which is conceptually close to the meaning of just, fair.

Table 5 lists some vertices with their NNs. Such a list captures genuine semantic similarity.

Table 5. Nearest neighbors capturing genuine semantic similarity

Vertex	Nearest Neighbor
公雞 (na)	雞 (na)
雞 (na)	雞子 (na)
雞母 (na)	雞子 (na)
雌性 (na)	雌 (a)
阿公 (na)	伯公 (na)

4.2. Uniqueness of the Three MSTs

For each of the three adjacency matrices D_{192} , D_{202} and D_{283} , some edges have a same weight. Therefore, we concluded than none of our three minimum spanning trees MST_{192} , MST_{202} and MST_{283} may be unique (Eppstein, 1995).

4.3. Counting the Number of MST Meaningful to our Study

The total number of MSTs can be calculated as described in § 3.2.5 . However, we aim at determining the number of MSTs embedding the maximum of information for meaning inference. The MSTs of interest are the MSTs for which the number of patterns defined in §3.2.6 would be maximum. We are going to prove that in our case any MST is the best one: all MSTs contain the same amount of information for meaning inference.

4.4. Any MST_{283} is optimal for meaning inference.

Let's assume $G = (V, E)$ to be the complete connected weighted graph of the semantic space $A_{283,100}$. We preably identified in §4.1.1 the three words that have more than one NNs. These three words and their NNs are neither dimension words neither $\hat{\text{公}}$ monomorphemic words and they account for the non-uniqueness of MST_{283} . Two of the three words share the same two nearest neighbors, so that these three words and their NNs form two subparts of the MST and not three ones. These two subparts - sub1MST and sub2MST - will be distinctly connected through any smallest shortest path to the remaining subpart of the MST - sub3MST -. Sub3MST connects all the vertices not belonging to sub1MST and sub2MST.

Lemma 2. ⁹*If every vertex of a graph G has only one nearest neighbor, the MST of G is unique.*

Sub3MST's vertices have each only one NN such that by application of Lemma 2, we deduced the uniqueness of sub3MST. Therefore, sub3MST embeds in a unique configuration all of the twelve meaning dimension words and all of the five monomorphemic $\hat{\text{公}}$ words and their associated NNs or RNNs. The closest words to these seventeen words are still in sub3MST. By consequence, the patterns of meaning inference are to be found in sub3MST and will not change in the different MSTs.

4.5. Meaning Inference for a 283 Vertices MST: MST_{283}

MST_{283} can be used to extract genuinely the meaning of a $\hat{\text{公}}$ in a $\hat{\text{公}}$ word.

⁹This lemma can be proved by using Prim's first necessary condition (NC1) for a MST in a simple proof by contradiction as in (Prim, 1957).

4.5.1. Inferring the Meaning of 公 in 公鹿 (Male Deer).

Figure 4 shows the path from one of the monomorphemic 公 word 公 (A) to 公鹿 (male deer). 公 (A) forms an edge with the meaning dimension word 雄性 (male or maleness). The pattern presented by Fig. 4 follows the pattern of Fig. 2 in §3.2.6. 雄性 is the NN of 公 (A), it represents its primary meaning. In Fig. 4, the augmented structure of the vertices of the non-公 words, 雌性 (female or femaleness) 雌 (female) is shown. These two words are both antonyms to 雄性. Following §3.2.7, path length is measured as 1: only one concept (femaleness) separates the concept of 公鹿 and 雄性.

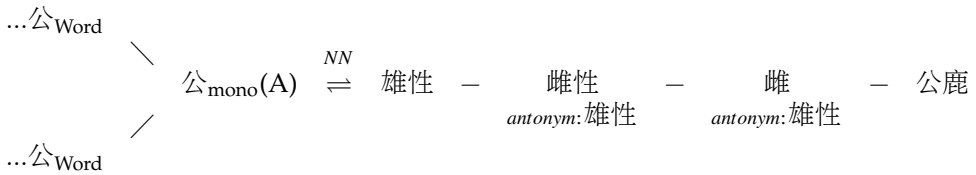


Figure 4. Augmented pattern for inferring the Meaning of 公 in 公鹿

As for Fig.2 in §3.2.6, it can be asserted that in $A_{283,100}$, the closest meaning of 公 in 公鹿 (male deer) is 雄性 (male, maleness). Every Chinese speaker will agree on the meaningfulness and correctness of such a conclusion.

4.5.2. Inferring the meaning of 公 in 大公國 (Grand Duchy).

From Tab. 1 and Tab. 2, we observed that 公 (vh) and 大公國 are RNN. 大公國 represents an unexpected dimension of meaning of 公. Hence, in $A_{283,100}$, the meaning of 公 (vh) would be the meaning of 大公國. The pattern for meaning inference is presented in Fig. 5. By an examination of edge weights, it is found that 公 (vh) and 國家 (na) are secondary reciprocal nearest neighbors (SRNN).

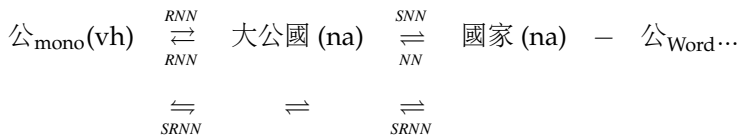


Figure 5. {公, 大公國, 國家} subcomponent of MST_{283}

國家 (country, nation) as mentioned in §2.2, is a possible meaning dimension word for some Taiwanese subjects. If here 國家 is considered as a meaning dimension word, the conclusion is that the meaning of 公 in both 公 (vh) and 大公國 is 國家 (country, nation).

4.5.3. Inferring the meaning of 公 in 廉政公署 (Independent Commission Against Corruption).

In Fig. 6, 公 (nc) and 廉政公署 (nc) are RNN. 廉政公署 (nc) represents an unexpected dimension of 公. The second nearest neighbor of 廉政公署 is 政府 (na). In $A_{283,100}$, the closest non-公 word to 公 (nc) and 廉政公署 is 政府 (government). Therefore the meaning of 公 in 廉政公署 and 公 (nc) would be 政府. Such an inference is congruent with 公署 in 廉政公署 meaning government office.

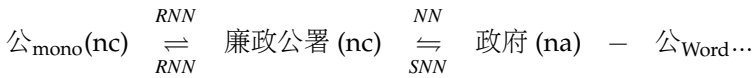


Figure 6. The meaning of 公 in 廉政公署 is 政府 (na).

5. Conclusion

This work brought major changes and improvements to the computational approach introduced by (Galmar and Chen, 2010) for inferring the meaning of a polysemous Chinese morpheme embedded in polymorphemic words. Our modified computational approach uses the similarity matrix of reduced Singular Value Decomposition to build nearest neighbors lists and minimum spanning trees whose vertices structure can be augmented to compute a conceptual distance measure. A constructive proof of optimality of the results have been given.

Our computational approach could serve as a first step to add a new feature to current Chinese Wordnets: the listing of all the Chinese words embedding a same polysemous morpheme with a fixed identified meaning. Such a listing could also assist cognitive scientists in preparing materials for experiments aiming at investigating the effects of repetitive exposure to Chinese polysemous morphemes embedded in compound words.

6. Future Directions

Firstly, Galmar and Chen (2010) mentioned that the Academia Sinica Balanced Corpus could not reflect adequately the representation of human knowledge. They advanced the Chinese Wikipedia - with its content organization following categorization meaningful to human - as a sound corpus for a replication of their study.

Secondly, our computational approach can be abstracted and viewed as a computational chain for processing Chinese words semantics. The chain can be unfolded into the following elements:

1. Dimensionality reduction.
2. Completeness reduction. Here the completeness of the graph - whose adjacency matrix is the similarity matrix - is reduced to a minimum spanning tree.
3. Building of an augmented structure with additional semantic relationship information.
4. Search of semantic patterns.
5. Logical Inference.

The first element in the chain, dimensionality reduction was done through Latent Semantic Analysis. Other possibilities would include:

1. Fiedlar retrieval: Hendrickson (2007) proposed that by considering the term-document matrix as a bipartite graph between the set of words and the set of documents and computing a set of the smallest eigenvalues of the Laplacian matrix of the bipartite graph, one can perform an enhanced kind of LSA analysis where unlikely to traditional LSA, documents and terms are considered equivalent and cohabiting in a same space. This approach belongs to spectral graph theory (Mohar, 1997; Chung, 1997; de Abreu, 2007).
2. Probabilistic models of semantic analysis: Latent Dirichlet Allocation (LDA) or Probabilistic LSA. They are probabilistic successors of LSA which have been found to outperform LSA (Blei et al., 2003, 2004; Blei and Lafferty, 2007).

For the second element in the chain, we could:

1. Replace the minimum spanning tree by other kinds of trees such as:
 - (a) Light Approximate Shortest-path Tree which is a hybrid tree between the shortest path tree and the minimum spanning tree (Khuller et al., 1995).
 - (b) The multi-criteria Minimum Spanning Tree (mc-MST) which takes into account constraints (Zhou and Gen, 1999).
2. Instead of working with the similarity matrix viewed as the adjacency matrix of a complete graph, we could consider to view the similarity matrix as the matrix representation of an hypergraph - a generalization of graphs where edges can join more than 2 vertices (Berge, 1976). Then, we would have to generate a minimum spanning tree for the hypergraph.

A last point, by iterating our approach with the change of the meaning dimensions words and the non- $\hat{\Delta}$ words, we could capture the meaning of $\hat{\Delta}$ in more multimorphic $\hat{\Delta}$ words than in a single iteration.

7. Acknowledgments

We thank Iris Huang and Train Min Chen for fruitful discussions and suggestions concerning the present work.

Bibliography

- Berge, C. *Graphs and hypergraphs*. North-Holland Pub. Co., 1976.
- Blei, D.M. and J.D. Lafferty. A correlated topic model of science. *Annals of Applied Statistics*, 1(1):17–35, 2007.
- Blei, D.M., A.Y. Ng, and M.I. Jordan. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.
- Blei, D., T.L. Griffiths, M.I. Jordan, and J.B. Tenenbaum. Hierarchical topic models and the nested Chinese restaurant process. *Advances in neural information processing systems*, 16:106, 2004.
- Broder, A.Z. and E.W. Mayr. Counting minimum weight spanning trees. *Journal of Algorithms*, 24(1):171–176, 1997.
- Chen, Jenn-Yeu, Bruno Galmar, and Hsiao-Jen Su. Semantic Satiation of Chinese Characters in a Continuous Lexical Decision Task. In *The 21st Annual Convention of the Association For Psychological Science*, 2009.
- Chung, F.R.K. *Spectral graph theory*. Amer Mathematical Society, 1997.
- de Abreu, N.M.M. Old and new results on algebraic connectivity of graphs. *Linear Algebra and its Applications*, 423(1):53–73, 2007.
- Eppstein, D. Finding the k smallest spanning trees. *BIT Numerical Mathematics*, 32(2):237–248, 1992.
- Eppstein, D. Representing all minimum spanning trees with applications to counting and generation. Technical report, Citeseer, 1995.
- Foulds, L.R. *Graph theory applications*. Springer, 1995.
- Galmar, B. and J.Y. Chen. Identifying Different Meanings of a Chinese Morpheme through Latent Semantic Analysis and Minimum Spanning Tree Analysis. *International Journal of Computational Linguistics and Applications*, 1(1-2):153–168, 2010.
- Galmar, Bruno and Jenn-Yeu Chen. Can neural adaptation occur at the semantic level? a study of semantic satiation. In *The 12th annual meeting of the Association for the Scientific Study of Consciousness*, 2007.
- Graham, R.L. and P. Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7(1):43–57, 1985.
- Gross, J.L. and J. Yellen. *Graph theory and its applications*. CRC press, 2006.
- Hendrickson, B. Latent semantic analysis and Fiedler retrieval. *Linear Algebra and its Applications*, 421(2-3):345–355, 2007.

- Khuller, S., B. Raghavachari, and N. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305–321, 1995.
- Landauer, T.K. and S.T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211–240, 1997.
- Landauer, T.K., D.S. McNamara, S. Dennis, and W. Kintsch. *Handbook of latent semantic analysis*. Lawrence Erlbaum, 2007.
- Mohar, B. Some applications of Laplace eigenvalues of graphs. *Graph Symmetry: Algebraic Methods and Applications*, 497:227–275, 1997.
- Packard, J.L. *The morphology of Chinese: A linguistic and cognitive approach*. Cambridge Univ Pr, 2000.
- Prim, R.C. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, 1957.
- Wright, P. Counting and constructing minimal spanning trees. *Bulletin of the Institute of Combinatorics and its Applications*, 21:65–76, 1997.
- Wright, P. On Minimum Spanning Trees and Determinants. *Mathematics Magazine*, 73(1):21–28, 2000.
- Wu, B.Y. and K.M. Chao. *Spanning trees and optimization problems*. Chapman & Hall, 2004.
- Zhou, G. and M. Gen. Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research*, 114(1):141–152, 1999.

Address for correspondence:

Bruno Galmar
hsuyueshan@gmail.com
National Cheng Kung University, Institute of Education
No.1, University Road, Tainan City 701, Taiwan (R.O.C.)
FAX: 886-6-2766493



Pragmatically-Motivated Utterance Fine-Tuning in Human-Computer Dialogue

Vladimir Popescu^a, Jean Caelen^b, Corneliu Burileanu^c

^a Laboratoire Informatique d'Avignon, University of Avignon, France

^b Laboratoire d'Informatique de Grenoble, CNRS, France

^c Politehnica University of Bucharest, Romania

Abstract

The naturalness and user-friendliness of the utterances generated by the computer when engaging in dialogue with humans is a key point for the success of spoken language interaction-based computer applications. This article addresses the issue by proposing a mechanism for controlling the strength of the illocutionary force conveyed by the utterances produced by the machine. The degree of strength for a speech act roughly quantifies the pressure that this act puts on its recipient. Few research efforts are reported for controlling this pressure in utterance generation. This article provides the means to adjust this force, relying on the discourse structure of the dialogue and on the public commitments that the speakers make during the dialogue. After a concise statement of the formal framework, the proposed approach is presented in a detailed manner and qualitatively assessed via relevant examples.

1. Introduction

In this article we address a particular issue in answer generation for human-computer dialogues. We are concerned with fine-tuning the degree of strength for the utterances generated by the computer in dialogues, so that a natural dialogue is achieved. Tuning this degree of strength is important not only for social reasons, but also for psychological reasons, related to the level of motivation that an utterance determines in its addressee. Hence, this degree of strength – named, in line with (Vanderveken, 1990-1991), *illocutionary force degree*, reflects, from a pragmatical viewpoint, the illocutionary goal set by a speaker when she produces an utterance¹. This illocutionary goal has, certainly, perlocutionary effects on the hearer, in that it

¹We will henceforth designate the speaker by *she*, and the hearer by *he*.

may determine his further dialogue turns. Although the importance of this aspect in linguistic interaction has been pointed out in several studies (Vanderveken, 1990-1991; Faller, 2006), the actual way the illocutionary force degree might be controlled was, to our knowledge, not specifically addressed. Therefore, the goal of this article is to specify a formal framework allowing us to compute the illocutionary force degree for an utterance that the machine is due to produce.

For computing the illocutionary force degree, we rely on two key ideas: (i) using (Maudet et al., 2004, 2006)'s work for computing public commitments starting from SDRT ("Segmented Discourse Representation Theory") rhetorical relations; (ii) using (Vanderveken, 1990-1991)'s ideas for defining and formalizing the notion of illocutionary force degree, adapting it to human-computer dialogue. Moreover, by the "agnostic" attitude that we adopt with respect to the beliefs and intentions of the agents, we follow a research agenda along the lines of (Kibble, 2006b,a). Indeed, we distance ourselves from the mentalistic approaches derived from the BDI ("Belief-Desire-Intention") framework (Cohen and Levesque, 1990b,a; Cohen and Perrault, 1979). We chose to do so, since computing intentions of the agents is considered to be rather unreliable, with respect to the social commitments, that are readily available (Kibble, 2006b, 2007).

The main contribution of this article consists in combining points (i) and (ii) above. In order to do this, we start from a set of assumptions. First, we assume that each dialogue partner (more specifically, the computer and the human speaker) has a *commitment store* that contains the semantic forms for the utterances each speaker overtly commits to during conversation (Kibble and Piwek, 2007), along with the rhetorical relations between them; the latter are accounted for in a framework inspired from SDRT (Asher and Lascarides, 2003). We posit that, from the viewpoint of the aspects tackled in this article, the goal of the machine is to achieve *dialogue* success, as opposed to *task* success, which is the responsibility of the dialogue and task managers (Caelen and Xuereb, 2007). Concerning speakers' commitments, dialogue success is achieved if the machine's and user's commitments are not logically inconsistent. A sufficient (albeit restrictive) and, in our opinion, easier to check, condition that guarantees the consistency of the commitment stores is their equivalence. Informally stated, the more the commitment stores are logically equivalent, the more the dialogue is successful. Although this hypothesis does not hold for the general case of human dialogues, it is supported, for service-oriented dialogues by empirical evidence brought by experiments performed using an artificial dialogue agent, in the context of a room reservation dialogue system (Nguyen, 2005). Unlike Maudet *et alia*, who consider the viewpoint of an *external* observer of the dialogue (Maudet et al., 2006), we adopt the viewpoint of the machine. This implies that the user's and machine's commitment stores are different if and only if there exists a semantic form that is either in the commitment store of the machine and not in the user's commitment store, or vice versa. In this situation, the machine has to adjust the illocutionary force degree²

²In line with (Caelen and Xuereb, 2007) and (Vanderveken, 1990-1991), we consider the notions of *speech act type* and *illocutionary force* as synonymous.

for this utterance, so that either the user commits to it, or the machine concedes to retract its commitment to this semantic form.

From the assumptions shown above, a general adjustment policy for the degree of illocutionary force can be derived, by taking into account the way the machine (as a speaker) wants to act on the commitments of its interlocutor: first, if the machine wants the interlocutor to integrate something in her/his commitment store, then it produces an utterance having a strong degree of force (that is, stronger than a “neutral” degree of force, usable for pure assertions, irrespective of the status of the commitments); second, if the machine wants its interlocutor to retract something from her/his commitment store, then it produces an utterance with an even stronger degree of force; this utterance is supposed to determine the interlocutor to give up one of her/his commitments. On the other hand, if the machine wants to signal its interlocutor that it integrated the contents of one of her/his utterances in its commitment store, it produces an utterance with a weak degree of force. Moreover, if the machine wants to signal that it concedes to retract one of its commitments (thus endorsing one of its interlocutor’s commitments), then it produces an utterance with an even weaker degree of force.

Concerning previous work related to the research described in this article, we can mention (Traum *et al.*, 2003)’s dialogue model, implemented with conversational agents that interact in a multi-party dialogue context, with virtual and human speakers, in several applications, ranging from military troops training, to interactive story-telling. This approach addresses two different aspects of the interaction: (i) the *discursive* character of the agents’ linguistic contributions – Traum *et al.* do not explicitly model rhetorical constraints imposed by dialogue history on the speech turns, this is why their approach only *simulates* rhetorical capabilities for the agents; (ii) the *intentionality* of the dialogue contributions – Traum *et alia*’s approach indeed succeeds in altering the surface form of communicative intentions, but they do this by relying on speakers’ *emotional attitudes*, which requires access to their intentional state. This can only be done reliably in very limited contexts (concerning a specified task), the approach does not seem readily transposable to different tasks, in that task-specific knowledge is not separate enough from task-independent knowledge.

Another relevant work is (Gupta *et al.*, 2007)’s POLLY system: the authors propose a system architecture for controlling the surface form of the utterances in dual-party human-system conversations. For this, the authors use (Brown and Levinson, 1987)’ Politeness Theory. Gupta *et alia* rely on pragmatic elements for choosing an adequate surface form for an utterance that is specified in a deep form (as a plan, i.e. a set of preconditions and actions to realize if the preconditions are satisfied). The authors use constraints imposed by a speaker’s *menaces*, which basically boils to taking into account three contextual factors: (i) the power that the hearer has over the speaker, (ii) the social distance between the speaker and the hearer, and (iii) an imposing level of the speech act realized via an utterance. Starting from Brown and Levinson’s Politeness Theory, Gupta *et alia* propose a fine-grained control of the surface form of the utterances, according to the type of the menace that the speaker undergoes. The passage from these politeness specifications (called “strategies” in Politeness Theory) to linguistic forms is realized in a rather *ad-hoc* manner, since predefined lexical items are “collated” to the prototypical, neutral, linguistic form of an utterance. Unlike in POLLY, we

propose to take into account other facets of the dialogue context, namely discourse structure and public commitments, in order to handle basically the same phenomenon – choosing a contextually-adequate linguistic form, between several alternatives for expressing the same semantic content.

In view of these previous research efforts however, we do not claim that the usage of the users' commitments as a decision criterion in computing the degree of illocutionary force yields better performances than politeness elements, or than emotion-related elements. We only try to show that the rhetorical structure, a resource that is already computed and used for other purposes (computing speech act types (Xuereb and Caelen, 2005), generating discourse connectors in multi-sentential utterances (Popescu and Caelen, 2009), or elliptic constructions (Popescu et al., 2008)), can also be used for acting on the degree of illocutionary force. The issue of *reusing* already available (computed) resources is of major importance in dialogue systems, where timely response to users' demands is essential from an ergonomical standpoint. However, the issue whether users' commitments are, in general, an appropriate resource to be taken into account when computing degrees of illocutionary force falls out of the scope of this article. What we do show though, is that there is a certain empirical motivation behind our approach.

This article is structured as follows: in Section 2 we introduce the setting where our contributions are made, i.e. task-oriented dialogue systems; in Section 3 we describe in detail the mechanisms used for adjusting the illocutionary force degree in human-computer dialogue; these mechanisms are also validated on several typical dialogue situations concerning a book reservation service. Lastly, a conclusive discussion ends up the article.

2. Research Context: Pragmatic Control of Utterance Generation

The work described in this article is developed in the context of a formal framework for controlling certain aspects of utterance production in human-computer dialogue. These aspects are either pragmatic in nature (such as the degree of strength that is assigned to a speech act type that is performed by an utterance), or semantic (such as the semantic ellipsis control in utterances). Both these sides are controlled by leaning on a formal account of the rhetorical structure of dialogues: the interaction has to take place so that the *global coherence* of the dialogue is maximized (Asher and Lascarides, 2003). On the other hand, an account of *local relevance* of the utterances is used in order to foster the global coherence; namely, the duality between utterance-level speech act types and rhetorical relations that connect pairs or larger sets of utterances is taken into account (Caelen and Xuereb, 2007), in order to improve the rhetorical structuring of the dialogue.

Thus, in this article we address the issue of how the commitment stores of the dialogue partners (Asher and Lascarides, 2008) trigger decisions regarding the adjustment of the degree of strength for the illocutionary force performed by the utterance. More specifically, we give an account on how illocutionary force degrees are computed from the interplay between the commitments of the speakers in dialogue.

Before continuing with the presentation of our formal framework for controlling the illocutionary force degree, we introduce a set of notational conventions: (i) U and M denote the dialogue partners, a user and the machine, respectively; (ii) L denotes a generic speaker; in our framework, this means either M , or a human speaker, U ; (iii) CS_L denotes the commitment store of speaker L ; (iv) π_i are utterance labels, for a positive integer i ; (v) $K(\pi)$ represents the semantic form for utterance π (expressed in a first-order logic); (vi) Σ_ρ represents the semantics for the rhetorical relation ρ , expressed in the same first-order logic as the utterances; (vii) $\text{emitter}(\pi)$ is a discourse function that returns the identity of the speaker that had produced utterance π ; (viii) $\text{equals}(\alpha, \beta)$ is a binary discourse predicate that is true if variable α is bound to the value β ; (ix) the operation \leftarrow_π states that the left-hand part is updated with the right-hand part, via utterance π .

3. Pragmatics-based Illocutionary Force Tuning

3.1. Formalizing Vanderveken’s Illocutionary Force Degree

In formalizing the notion of degree of illocutionary force, we rely somehow on previous work of (Vanderveken, 1990-1991). Thus, in an utterance generation context, we assume that the input to the generator is a logical expression of the form $F(p)$, where

- F is an *illocutionary force* in Vanderveken’s terminology, or a speech act type, in Caelen’s terminology: F^F (“make-do”, a directive act), F^S (“make-know”, an act of informing the hearer on something), F^{FS} (“make-do-know”, a request for information from the hearer), F^P (“make-can”, an act of providing the hearer with several choices), et F^D (“make-must”, an act of imposing an obligation on the hearer) (Caelen and Xuereb, 2007);
- p is a *propositional content* in Vanderveken and Caelen’s terminologies; in the framework proposed here, this propositional content is identified with the logical formula that expresses the semantics of the utterance that realizes the act F ; such a propositional content is denoted by $K(\pi)$, as above, if the label of utterance $F(p)$ is π .

Like in the work of Vanderveken and Caelen, we assume that each speech act type F can be realized (via an utterance) with a certain degree of strength; we call this degree *illocutionary force degree* and we denote it by $\partial\phi$. Thus, the relation between an utterance, its propositional content, its illocutionary force and the degree of this force can be seen as a series of functional compositions:

$$\pi \rightarrow^K K(\pi) \rightarrow^F F(K(\pi)) \rightarrow^{\partial\phi} \partial\phi(F(K(\pi)) \times K(\pi)),$$

which can also be written as $(\partial\phi \circ (F \circ K) \times K)(\pi)$. We explicitly present below each of these functions.

The K function assigns meanings (expressed in a first-order logic) to utterances and to rhetorical relations; thus, it maps labels to logical formulas:

$$K : \Pi \cup P \rightarrow \Lambda,$$

where $\Pi = \{\pi_1, \dots\}$ and $P = \{\rho_1, \dots\}$ are two sets of labels – for utterances and for rhetorical relations, $\Lambda = \{\lambda_1, \dots\}$ is a set of logical formulas.

Thus, the K function is computed, when utterance generation is concerned, by the dialogue controller and, when user utterance analysis is concerned, by the semantic analyzer. However, we assume that the representations produced by the semantic analyzer and by the dialogue controller are coherent, i.e. the same set of predicates (in the task ontology) is used. When the K function takes the label of a rhetorical relation as argument, its operation is equivalent to a simple search of the semantics of the rhetorical relation under discussion.

Function F assigns speech act types to (the semantics of) utterances; this function is undefined on the semantics of the rhetorical relations:

$$F : \Lambda \setminus \{\Sigma_\rho : \rho \in P\} \cup \rightarrow \{F^F, F^S, F^{FS}, F^P, F^D\};$$

here, $\{\Sigma_\rho : \rho \in P\}$ represents the set of semantics for all known rhetorical relations; if function F is applied on a formula $K(\pi)$ such that it is the machine that produced utterance π (hence, $\text{equals}(M, \text{emitter}(\pi))$), then it is the dialogue controller that computes the F function.

Function $\partial\phi$ assigns degrees of strength to speech act types, that are in turn applied to the logical forms of utterances; thus, $\partial\phi$ maps speech act types to integers.

In order to endow the utterances with more variability, we stipulate that function $\partial\phi$ can have, for the same speech act type, effects that depend on the propositional content of the utterance, $K(\pi)$. Thus, we define function $\partial\phi$ on the Cartesian product between the set of speech act types and the set of propositional contents:

$$\partial\phi : \{F\} \times (\Lambda \setminus \{\Sigma_\rho : \rho \in P\}) \rightarrow \{-2, -1, 0, 1, 2\},$$

where $\{F\}$ and Λ denote, respectively, the set of speech act types, and the set (theoretically infinite, but practically limited by the particular task considered) of logical formulas that can be generated from a specified (e.g. in the task ontology) set of predicates. We assign five possible levels to the illocutionary force degree, classifying them in **very strong**, **strong**, **neutral**, **weak** and **very weak**; -2 and -1 represent very weak and weak forces respectively, 0 represents the neutral force, and 1 and 2 represent strong and very strong forces, respectively.

The goal of this article is to define “analytically” the $\partial\phi$ function. Before doing this, we are first going to motivate the illocutionary force degree via an example. We will thus show that there is a relevant relationship between the illocutionary force degree that is assigned to an utterance, and its linguistic form. Consider for instance a dialogue between a user U and a machine M , where the user says:

U : Sorry, can you tell me, please where I can find book ‘B’?

The dialogue manager in M produces, as a response to U ’s question, a communicative intention of informing the user that the book can be found on the first floor, to the left; this communicative intention is expressed, in logical form, and offered to the generator (π denotes the label of this utterance):

$$F^S(\exists X, Y, Z, T, U, V : \text{agent}(X) \wedge \text{equals}(X, \neg \text{emitter}(\pi)) \wedge \text{object}(Y) \wedge \text{equals}(Y, \text{'book'}) \wedge \text{feature}(Y, Z) \wedge \text{equals}(Z, \text{'title'}) \wedge \text{equals}(Z, \text{'B'}) \wedge \text{feature}(Y, T) \wedge \text{equals}(T, \text{'location'}) \wedge \text{feature}(T, U) \wedge \text{equals}(U, \text{'level'}) \wedge \text{equals}(U, \text{'1'}) \wedge \text{feature}(T, V) \wedge \text{equals}(V, \text{'direction'}) \wedge \text{equals}(V, \text{'left'})) .$$

Some possible linguistic forms, for this communicative intention and different degrees for the illocutionary force F^S , are (linguistic markers associated to illocutionary force degrees are written in boldface):

- $\partial\phi(F^S(K(\pi)), K(\pi)) = -2$:
M: On the first floor, to the left, **I think**.
- $\partial\phi(F^S(K(\pi)), K(\pi)) = -1$:
M: **Wait a minute please... here it is**: book ‘B’ is on the first floor, to the left.
- $\partial\phi(F^S(K(\pi)), K(\pi)) = 1$:
M: You can **certainly** find book ‘B’ on the first floor, **just** to the left.
- $\partial\phi(F^S(K(\pi)), K(\pi)) = 2$:
M: **Listen** you can **certainly** find book ‘B’ on the first floor, **just** to the left, **do you understand?**

Given that the speech act type considered is an F^S (provision of information), all these forms stem from a neutral illocutionary force degree:

$$\partial\phi(F^S(K(\pi)), K(\pi)) = 0:$$

M: You can find book ‘B’ on the first floor, to the left.

After having seen this example, we observe that, from an answer generation viewpoint, there are two issues to deal with, concerning the illocutionary force degree: (i) computing the appropriate illocutionary force degree, for a given utterance, specified in logical form, and (ii) mapping a specified illocutionary force degree to an appropriate linguistic form. In this article, only the first issue is explicitly addressed. The second problem, of mapping specified illocutionary force degrees to surface forms, is tackled by manual annotation of canned utterances.

Before introducing the illocutionary force handling mechanisms, we discuss on the adequacy of the five-level scale for the degree of illocutionary force. In our view, this issue has two facets. First, why a discrete and not a continuous scale? This has been argued for at length in several articles by Searle and Vanderveken *inter alia*, e.g. (Searle and Vanderveken, 2006), or (Motsch, 1980). Essentially, a discrete scale seems to be rather a theoretical choice, partly supported by the idea that language is a discrete phenomenon, than a necessary constraint. Second, why a five-level scale? An answer is that the choice of five levels is arbitrary (in theory, one can have a denumerable set of levels if we wish, but for a practical system one has to make a choice for a finite number of degrees). However, we chose five levels that correspond, roughly, to the informal (and intuitive) distinction between very weak, weak, neutral, strong and very strong degrees. Moreover, given that in the following subsection we discuss and illustrate the coupling between the relations between interlocutors’ commitments and the very weak/weak/strong/very strong degrees of force, the -2 to $+2$ scale seems a methodologically convenient choice, i.e., canned phrases can easily be annotated in this way, so that an answer generation module can choose a suitable utterance (i.e., parameterized by the required degree of force).

3.2. Illocutionary Force Handling Mechanism

The main idea that guides the computation of the illocutionary force degree stems from the connection between public commitments and speech acts. As we emphasized before, we rely on (Maudet et al., 2006)'s framework for computing commitment stores from discourse structure.

For each user U in a dialogue, there exists a commitment store CS_U that contains the semantics of the utterances that U has produced, along with the semantics of the machine's utterances, that U has agreed with (this is indicated by rhetorical relations between these utterances and utterances of U), and finally, along with the *negated* semantics of the machine's utterances that U did *not* agree with, along with the rhetorical relations that emphasize this fact, e.g. *P-Corr* (Plan Correction) or *Contrast* (Asher and Lascarides, 2003). For example, consider the following dialogue, between a human user U and the machine M , which simulates a librarian:

M : You can still borrow three books, sir!

U : So, I can take this one as well?

M : Yes, you can take it, sir.

This interaction contains a question of U , that is in an $Elab_q$ relation to the first utterance of M ; the subsequent answer of M is in an *Elaboration* relation to the first utterance, since, indeed the two turns of M achieve the same effect (from the point of view of the task that the dialogue tries to help resolving) as a unique turn of M :

M : You can still borrow three books, so, for instance, you can take book 'X' that you want.

where book 'X' and 'this one' in the user's question, refer to the same object in the physical world.

The way that commitment stores are defined and used is inspired from (Maudet et al., 2006); for instance, in the previous dialogue example, the commitment store of U , after she had asked the question, is a set:

$$CS_U = \{K(\pi_1), K(\pi_2), \Sigma_{Elab_q(\pi_1, \pi_2)}\},$$

where π_1 and π_2 denote the first utterance of M and the first utterance of U (the question) respectively, the function $K(\pi)$ denotes the semantic content of utterance π , and $\Sigma_{Elab_q(\pi_1, \pi_2)}$ denotes a Prolog-style semantics of the rhetorical relation $Elab_q(\pi_1, \pi_2)$, which specifies that utterance π_2 is a question such that any relevant answer elaborates on utterance π_1 (Asher and Lascarides, 2003).

The notion of commitment store and the way such a structure is computed for each speaker in dialogue could have been given a better account, by using the notion of *common ground*. More specifically, we could have relied on the idea that common ground in dialogue can be seen as the joint entailments of all the speakers' public commitments (Lascarides and Asher, 2009). Thus, commitments could have been seen as SDRSs, one such discourse structure for each speaker. Moreover, in an adequate framework, the commitments should be computed by taking into account the *preferences* of the speakers, which are learned from and affected by conversational moves. The dynamic interplay between preferences and conversational actions

could be seen as a game-theoretical problem and modeled as such (Asher and Lascarides, 2008). However, we believe that in the context of this article, where the emphasis is put on the way these commitments are *used* for computing illocutionary force degrees, the approximation provided above for the commitment stores is sufficient as a departure point.

Then, for computing a scalar value for the illocutionary force degree, we rely on the intuition that the closer the speakers' public commitments are, the closer to zero (a *neutral* value) the illocutionary force degree is. In other words, in an ideal situation where U and M share exactly the same commitments throughout the dialogue, the machine can express its utterances in a neutral manner, from an illocutionary standpoint. That is, M assigns a zero degree to the illocutionary force.

For most of the speech turns in dialogue we have that $CS_U \neq CS_M$, i.e., U and M 's commitment stores are not equivalent³ after each speech turn. This non-equivalence results in several possible cases, concerning the relationship between the interlocutors' commitments:

1. $CS_M \subset CS_U$, that is, M 's commitment store is strictly included in U 's commitment store; this is typical for dialogues where the user teaches the machine new tasks;
2. $CS_M \supset CS_U$, that is, conversely, M 's commitments strictly include U 's commitments; this situation is typical for tutoring dialogues;
3. $CS_M \cap CS_U \equiv CS_{\cap} \neq \emptyset$, that is, there is a certain overlapping between U and M 's commitments; this is typical for service-oriented dialogues, where the machine interacts with previously unknown users;
4. $CS_M \cap CS_U \equiv \emptyset$, that is, M and U have no commitment in common; this is typical for dialogues where speakers have fundamentally different cultural backgrounds, or where the user teases the system; this type of dialogue represents only unsuccessful interactions.

Before furthering the discussion, we have to explain what we mean by intersecting two commitment stores: this notion can be considered either at a purely syntactic level (i.e. the logical forms contained *ad litteram* in the two commitment stores), or at a semantic level (i.e. the logical forms present in the commitment stores and satisfiable in the same set of models). Similarly to the commitments equivalence, we consider the intersection of commitment stores at a semantic level (a syntactic definition would have been too constrained). Thus, in order to automatically determine the intersection of two commitment stores, or to evaluate their equivalence, we go through several steps, stated below:

for two commitment stores CS_i and CS_j :

(a) **for** each logical formula $\lambda_k^{(i)} \in CS_i$ and $\lambda_l^{(j)} \in CS_j$:

i. compute the models where these formulas are satisfiable:

$$\mu_k^{(i)}[m] : \mu_k^{(i)}[m] \models \lambda_k^{(i)}; \mu_l^{(j)}[n] : \mu_l^{(j)}[n] \models \lambda_l^{(j)};$$

ii. compute the conjunctions of these models:

$$M_k^{(i)} = \bigwedge_{m; \mu_k^{(i)}[m] \neq \emptyset} (\mu_k^{(i)})[m]; M_l^{(j)} = \bigwedge_{n; \mu_l^{(j)}[n] \neq \emptyset} (\mu_l^{(j)})[n];$$

³This equivalence is considered at a semantic level, stemming from the identity of the models where the logical forms in the commitment stores are satisfiable.

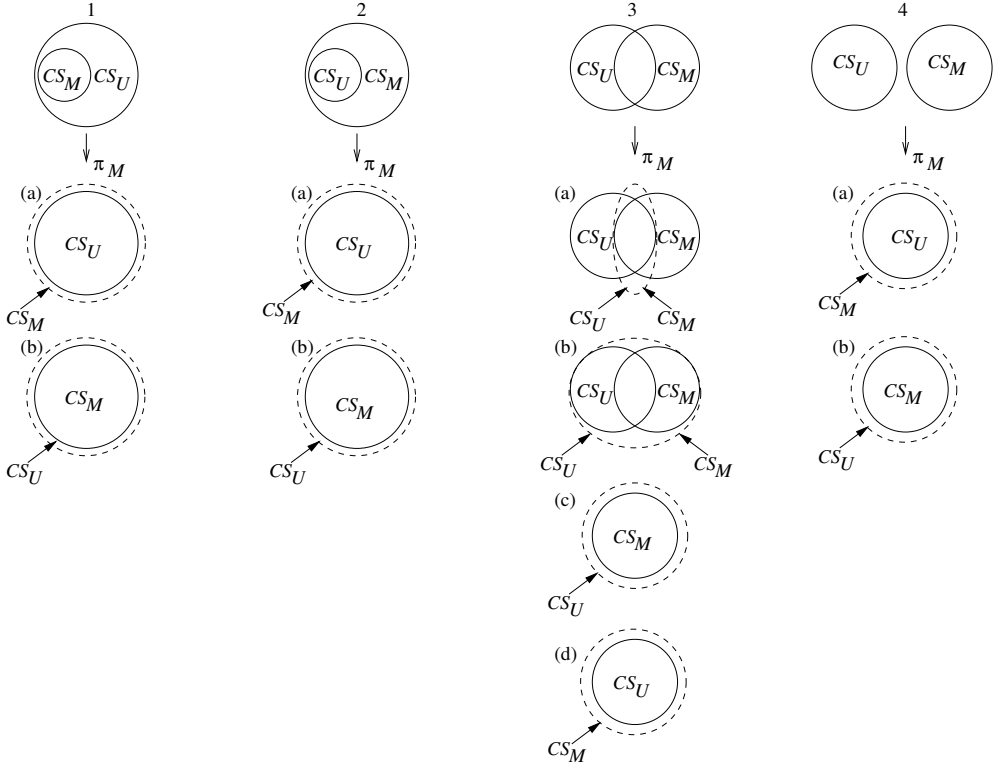


Figure 1. Commitment stores for U and M

(b) compute the unions of the conjunctions of the models:

$$M_i = \bigcup_{k: M_k^{(i)} \neq \emptyset} (M_k^{(i)}); M_j = \bigcup_{l: M_l^{(j)} \neq \emptyset} (M_l^{(j)}).$$

Then, we compare these sets M_i and M_j for the two commitment stores CS_i and CS_j , in order to evaluate their equivalence or their intersection. The crucial issue with this approach lies in determining the set of models that satisfy the logical formulas. Essentially, this is accomplished by considering the whole task ontology as a model, from which we first eliminate the predicates that make the logical formula unsatisfiable in the model. Then, we can still generate models where the logical formula is satisfiable, by further eliminating predicates in the task ontology. However, we thus obtain a set of models strictly ordered with respect to inclusion. We therefore observe that the sets $M_k^{(i)}$, for each logical formula $\lambda_k^{(i)}$ in the commitment store CS_i , are subsets of predicates in the task ontology, such that adding $\lambda_k^{(i)}$ to this subset does not lead to a contradiction.

The strict inclusion relation (\subset) between commitment stores is defined in semantic terms as well, hence of models where the formulas contained in the commitment stores are satisfiable. Thus, an inclusion relation between two commitment stores CS_i and CS_j is equivalent to the *inverse* inclusion relation between the corresponding sets of models M_i and M_j :

$$CS_i \subset CS_j \Leftrightarrow M_j \subset M_i.$$

This is explained by the fact that the more formulas are contained in a commitment store (modulo the logical equivalence), the less there are models that satisfy *all* these formulas. The difference operation between two commitment stores is defined in model-theoretic terms as well: the result of a difference between two commitment stores represents the set of models that do not satisfy any formula in the second commitment store, but satisfy all the formulas in the first commitment store that are not equivalent to the formulas in the second commitment store:

$$CS_i \setminus CS_j = \{\mu : \forall \lambda_j \in CS_j \mu \not\models \lambda_j \wedge \forall \lambda_i \in CS_i : \lambda_i \not\equiv \lambda_j \mu \models \lambda_i\}.$$

Given the fact that throughout the article we adopt the machine's viewpoint, in order to predict its behavior in producing a speech turn, we assume that, in each of the four cases presented above, the next dialogue contribution to be produced belongs to the machine; we thus denote by π_M the *next* speech turn waited from M . In this context, we analyze in a detailed manner below each of the four situations indicated above (concerning the relations between commitment stores); these situations are illustrated in Figure 1:

Case 1: $CS_M \subset CS_U$

M 's goal in dialogue is to make the commitment stores of the speakers equivalent, that is, $CS_M \equiv CS_U$. We assume that, when the machine produces turn π_M , it tries to modify as scarcely as possible its own commitment store. Moreover, for simplicity we assume that π_M consists of a single utterance; in fact, this does not reduce the generality of the problem, because we can assume that the machine produces each utterance sequentially, even in a speech turn that contains several utterances. Thus, two possible situations arise:

(a) $CS_M \leftarrow_{\pi_M} CS_M \cup (CS_U \setminus CS_M)$:

In this case, M produces an utterance π_M through which it marks the acceptance of all the preceding utterances in U 's commitment store, but absent from his own commitments, i.e., the machine commits to $K(\pi_{U-}) = CS_U \setminus CS_M$, where π_{U-} denotes a (set of) utterance(s) previously produced by U . Thus, the machine chooses a *weak* degree of illocutionary force for π_M ⁴: $\partial\phi(F(K(\pi_M))) = \partial\phi(\pi_M) = -1$.

(b) $CS_U \leftarrow_{\pi_M} CS_U \setminus (CS_U \setminus CS_M)$:

In this case, M produces an utterance π_M through which it tries to convince U to withdraw her commitment to $K(\pi_{U-})$ (i.e., to facts the machine is not committed to). Thus, the machine chooses a *very strong* degree of illocutionary force: $\partial\phi(\pi_M) = 2$.

⁴We will henceforth simplify the notation and write $\partial\phi(\pi)$ instead of $\partial\phi(F(K(\pi)), K(\pi))$ for the illocutionary force degree of an utterance labeled π .

The machine will first try the second possibility and, only if this fails (i.e. the user does not cooperate with the machine in adjusting its commitments to M 's), M will choose the first possibility.

Case 2: $CS_M \supset CS_U$

Assuming that the machine has the same goals as in the preceding case (i.e., to make its commitments equivalent to the user's commitments) two possibilities arise as well:

(a) $CS_M \leftarrow_{\pi_M} CS_M \setminus (CS_M \setminus CS_U)$:

In this case, M produces utterance π_M through which it concedes to withdraw previous commitments not shared by U (i.e., $K(\pi_{M-}) = CS_M \setminus CS_U$, where π_{M-} denotes an utterance previously produced by M). Thus, the machine marks this withdrawal by a *very weak* degree of illocutionary force for its utterance: $\partial\phi(\pi_M) = -2$.

(b) $CS_U \leftarrow_{\pi_M} CS_U \cup (CS_M \setminus CS_U)$:

In this case, M produces utterance π_M through which it tries to convince U to commit to $K(\pi_{M-}) = CS_M \setminus CS_U$. Thus, the machine chooses a *strong* degree of illocutionary force for π_M : $\partial\phi(\pi_M) = 1$.

As in the previous case, the machine first tries the second possibility and, only if this fails, M will go for the first possibility.

Case 3: $CS_M \cap CS_U = CS_{\cap}$

For the subsequent presentation, we introduce the following notations: (i) $K(\pi_{M-}) = CS_M \setminus CS_U$, (ii) $K(\pi_{U-}) = CS_U \setminus CS_M$, (iii) $K(\pi_{UM-}) = CS_U \cap CS_M$, (iv) $K(\pi_{M-U-}) = CS_M \setminus CS_{\cap}$, (v) $K(\pi_{U-M-}) = CS_U \setminus CS_{\cap}$. We thus have that $CS_U \cup CS_M = \{K(\pi_{M-}), K(\pi_{U-}), K(\pi_{UM-})\}$.

Under the same assumptions concerning M 's goal in dialogue and the heuristics on the preferences for updating the commitment stores, four possibilities arise in this case:

(a) $(CS_M \leftarrow_{\pi_M} CS_{\cap}) \wedge (CS_U \leftarrow_{\pi_M} CS_{\cap})$:

In this situation, M produces utterance π_M through which it concedes to withdraw its commitment to $K(\pi_{M-U-})$ and it tries to convince U to withdraw its commitment to $K(\pi_{U-M-})$. Thus, the machine first aggregates π_M in two utterances, π'_M and π''_M , such that $K(\pi'_M) \wedge K(\pi''_M) = K(\pi_M)$. Secondly, M assigns a *very weak* degree of illocutionary force to π'_M : $\partial\phi(\pi'_M) = -2$, and a *very strong* degree of illocutionary force to π''_M : $\partial\phi(\pi''_M) = 2$.

(b) $(CS_M \leftarrow_{\pi_M} CS_M \cup CS_U) \wedge (CS_U \leftarrow_{\pi_M} CS_M \cup CS_U)$:

In this case, M produces utterance π_M through which it commits to $K(\pi_{U-})$ and it tries to convince U to commit to $K(\pi_{M-})$. Thus, the machine first aggregates π_M in two utterances, π'_M and π''_M , such that $K(\pi'_M) \wedge K(\pi''_M) = K(\pi_M)$. Secondly, M assigns a *weak* degree of illocutionary force to π'_M : $\partial\phi(\pi'_M) = -1$, and a *strong* degree of illocutionary force to π''_M : $\partial\phi(\pi''_M) = 1$.

(c) $CS_U \leftarrow_{\pi_M} CS_M \equiv CS_U \setminus (CS_U \setminus CS_M) \cup (CS_M \setminus CS_U)$:

In this case, M produces utterance π_M through which it tries to convince U to withdraw its commitment to $K(\pi_{U-})$ and to commit to $K(\pi_{M-})$. Thus, the machine first aggregates π_M in two utterances, π'_M and π''_M such that $K(\pi'_M) \wedge K(\pi''_M) = K(\pi_M)$. Secondly, M chooses a *very strong* degree of illocutionary force for the first utterance, and a *strong* degree for the second utterance: $\partial\phi(\pi'_M) = 2$ and, respectively, $\partial\phi(\pi''_M) = 1$.

- (d) $CS_M \leftarrow_{\pi_M} CS_U \equiv CS_M \setminus (CS_M \setminus CS_U) \cup (CS_U \setminus CS_M)$:

In this case, M produces utterance π_M through which it withdraws its commitment to $K(\pi_{M-U-})$ and commits to $K(\pi_{U-M-})$. Thus, the machine first aggregates π_M in two utterances, π'_M and π''_M such that $K(\pi'_M) \wedge K(\pi''_M) = K(\pi_M)$. Secondly, M assigns a *very weak* degree of illocutionary force to π'_M and a *weak* degree to π''_M : $\partial\phi(\pi'_M) = -2$, and $\partial\phi(\pi''_M) = -1$, respectively.

We assume that the machine adopts the heuristic of trying to modify as scarcely as possible its commitments, preferring to add utterances, rather than removing them. Thus, the order of preference for the possibilities listed above is: (c) \rightarrow (b) \rightarrow (a) \rightarrow (d).

Case 4: $CS_M \cap CS_U = \emptyset$

Adopting the constraint of not having empty commitment stores following an updating operation, and keeping the same assumption on M 's goal in dialogue, as well as on the heuristic concerning the preferences in commitment stores updating, we have two possibilities:

- (a) $CS_M \leftarrow_{\pi_M} CS_U$:

In this situation, via π_M M withdraws all its commitments, accepting all the user's commitments instead. This represents an extreme version of the case 3.(d), but the formal treatment is identical to that case.

- (b) $CS_U \leftarrow_{\pi_M} CS_M$:

In this case, M produces utterance π_M through which it tries to convince U to give up all his previous commitments, while accepting all of M 's commitments. This situation represents an extreme version of the case 3.(c), but the formal treatment remains unchanged.

Given the heuristic whereby M modifies as scarcely as possible its commitment store, preferring additions to it instead of removals, the order of preferences for the possibilities presented above for this case is: (b) \rightarrow (a).

Concerning the assignment of illocutionary force degrees, several comments arise, in order to defend the choice of a five-level scalar representation, and to clarify the distinction between the five possible values for this degree of force. Thus, given a speaker L and an utterance π produced by L , we can depict several situations:

- when via π , L concedes to remove something from its commitment store CS_L , like $CS_L \leftarrow_{\pi} CS_L \setminus \{\pi_{L-}\}$ for some previous utterances π_{L-} of L , we have that $\partial\phi(\pi) = -2$, hence a **very weak** degree of illocutionary force;

- when via π , L adds something to its commitment store CS_L , like $CS_L \leftarrow_{\pi} CS_L \cup \{\pi_{\bar{L}-}\}$ for some previous utterances of another speaker, \bar{L} , we have that $\partial\phi(\pi) = -1$, hence a **weak** degree of the illocutionary force;
- when via π , L tries to convince another speaker \bar{L} to add something to its commitment store $CS_{\bar{L}}$ (i.e., $CS_{\bar{L}} \leftarrow_{\pi} CS_{\bar{L}} \cup \{\pi_{L-}\}$), we have that $\partial\phi(\pi) = 1$, hence a **strong** degree of illocutionary force;
- when via π , L tries to convince another speaker \bar{L} to remove something from its commitment store $CS_{\bar{L}}$ (i.e., $CS_{\bar{L}} \leftarrow_{\pi} CS_{\bar{L}} \setminus \{\pi_{\bar{L}-}\}$ for a previous utterance $\pi_{\bar{L}-}$ of \bar{L}), we have that $\partial\phi(\pi) = 2$, hence a **very strong** degree of illocutionary force.

For the **neutral** degree of illocutionary force (i.e., of value 0), we stipulate that an utterance can be produced with a neutral degree only if it asserts un-contestable facts in the world (which are encoded in the task ontology), for instance the fact that ‘Normally, a book cannot be read in the total absence of light’, it realizes a speech act of the type F^S and is uttered for the first time in dialogue (i.e. it is not re-uttered in an attempt to make the interlocutor accept the utterance, after having previously rejected it). By consequence, any other type of utterance, realizing any other speech act type, cannot have a neutral degree of illocutionary force.

We thus observe how the set of possible relations between the commitment stores for the speakers in dialogue is mapped into a set of values for the degree of illocutionary force. This provides a first justification *a posteriori* for the five-level scale for this degree of force. Yet, the choice of a discrete scale for the degrees of illocutionary force can still seem methodological in nature: indeed, it makes it possible to annotate easier corpora of pre-defined utterances that thus map degrees of force into linguistic realizations. However, this five-level scale accounts well for the (set-theoretic) relations between the commitment stores of the speakers. For instance, if we had chosen a continuous scale of values in an interval (e.g., $[-1; +1]$), we would have had difficulties in mapping these values to relationships between commitment stores.

A possible criticism to this approach concerns the dialogue success criterion. Indeed, we could consider the following dialogue example, between the machine M (a virtual librarian) and a user U :

U_1 : Hello, I would like to have some books on the French revolution!

M_1 : Hello, Sir, I have two books: an introductory one, ‘X’, and a more advanced one, ‘Y’.

U_2 : I would like to have book ‘Y’, it seems more interesting.

M_2 : OK, give me your card, please.

In this dialogue, the utterance ‘there is a book ‘Y’ on the French revolution in the library’ should belong to CS_M and to CS_U . Nevertheless, whereas the utterance ‘there is a book ‘X’ on the French revolution in the library’ belongs to CS_M ; U has not committed to its content, either explicitly or implicitly; U is neutral about the contents of this utterance; hence, the utterance does not belong to CS_U . Thus, after these speech turns, we have that $CS_U \subset CS_M$; yet, the dialogue is successful, since the user manages to make its reservation. The dialogue would not have been more successful if, for instance, U had produced ‘OK, I agree’ as well, in

the beginning of speech turn U_2 , thus committing to the entire content of the previous speech turn of M (that is, to the contents of M_1):

U'_2 : OK, I agree. I would like to have book ‘Y’, it seems more interesting.

On the one hand, the equivalence between the commitment stores of U and M in dialogue represents a *sufficient* condition to dialogue success; even though this condition is not always *necessary*, its fulfillment ensures that the dialogue is successful. On the other hand, speech turn U_2 may represent an *indirect implicit* confirmation (essentially, if U does not contradict, explicitly or implicitly, any part of the preceding speech turn of M , then we stipulate that *by default* U commits to the entire speech turn of M) to the *whole* speech turn M_1 .

We further argue why we considered commitments’ equivalence instead of commitments’ mutual consistency as a dialogue success criterion. The argument pertains to computational considerations. We consider that it is easier to perform a mere “set-theoretic” commitments equivalence check, than a consistency check, because a consistency check would require us to verify for revisions in the commitment stores, that is, to check the *entire* commitment stores for *each* commitment store updating process. However, if equivalence is checked, only what is being *currently* added to the commitment stores needs to be checked, since their equivalence is aimed at during the entire dialogue. Even though this equivalence is never attained, it is “asymptotically” aimed at, in that the machine (not necessarily like the human interlocutor, and this is important, since, in our view, the machine need neither imitate, nor even approximate human behavior, but only be a useful and “friendly” assistant to the human) tries to adjust its commitments (and, if necessarily, its interlocutors’ commitments) so that they are the closest possible, for each dialogue turn pair. Hence, although restrictive, the condition on the equivalence of the commitment stores is, we believe, operationally more feasible than the check on their consistency.

Lastly, another relevant criticism stems from the fact that the degree of illocutionary force is not determined only from the discourse structure and the public commitments of the speakers: social roles are involved as well, in limiting the set of values that the degree of illocutionary force might take. According to our framework as it was presented until now, in a situation where the user U states α and the machine M states $\neg\alpha$, and the machine tries to convince U to give up his commitment to α , then it should utter $\neg\alpha$ with an illocutionary force degree of $+2$. This would lead the machine to produce an utterance like: ‘But Sir, α is certainly not true, do you understand?’. In a service scenario, where U is the customer and M the “server”, this utterance is not appropriate, because it would lead U to complain about M ’s impoliteness. This is a case where M has much less power than U in conversation. M simply does not have the right to produce an utterance with an illocutionary force degree of $+2$. All the machine can do in this case is to produce an utterance with a degree of force of $+1$: ‘Sir, I am convinced that α is not true’. We therefore observe how a hierarchical social status with respect to the interlocutor induces a tighter upper bound for the degree of illocutionary force. In the same context, we can ask ourselves whether, conversely, there are relational configurations that induce a tighter lower bound for the degree of illocutionary force as well. We believe that here such a lower bound can only be imposed by the personality of the speaker: for example,

if the latter has a proud nature, then she will tend not to produce utterances with a degree of force of -2 (for thus giving up its previous commitments), preferring to produce utterances with a degree of force of -1 instead. Thus, the speaker only concedes to accept the interlocutor's utterances (that could in turn *entail* the withdrawal of certain commitments). Consider for instance the following dialogue, between a user and a "proud" machine, in the sense stated above:

U_1 : But book 'X' is not on the first floor, as you told me! Actually, this book is on the second floor!

M_1 : Hm... I think you are right; I will send one of my colleagues on the second floor bring you the book.

In M_1 (which has a degree of illocutionary force of -1), the machine does not explicitly withdraw its previous commitment to the fact that book 'X' were on the first floor, preferring to confirm vaguely user's turn U_1 . Had the machine been less "proud", it would have explicitly given up its commitment to the fact that book 'X' were on the first floor, as in a speech turn $M_1^{(r)}$: 'Yes, indeed, I was wrong, I am sorry; hence, I will send one of my colleagues on the second floor bring you the book'.

To conclude on the social roles, we propose, in a first approximation, that the domain of values for the degree of illocutionary force be limited by an upper bound that stems from a lower position on the social hierarchy, with respect to the interlocutor. In the "virtual librarian" example, the latter could only produce utterances with illocutionary force degrees in the set $\{-2; -1; 0; +1\}$. On the contrary, the human user could produce utterances with any degree of illocutionary force, between -2 and $+2$. The issue of a finer analysis of the effects that social roles induce on the illocutionary force in producing utterances in conversation remains open to further study.

3.3. Assessment of the Framework

In this section we assess the proposed framework via two typical dialogues, illustrating the mechanism for adjusting the degree of illocutionary force. The mapping between degrees of force and linguistic form is realized by using canned phrases, annotated with illocutionary force degrees, based on linguistic intuitions. The two dialogues considered differ only in certain speech turns, the task context being the same: a user U tries to find a book 'B' in a library, whose (virtual) librarian is the machine M .

Dialogue 1:

U_1 : Sorry, can you tell me, please, where I can find book 'B'? — $F^{FS}(\pi_1)$

M_1 : Hello, well, you can find the book just at the end of the corridor, to the left. — $F^S(\pi_2)$; $QAP(\pi_1, \pi_2)$

U_2 : But I've just looked there, and I couldn't find the book! — $F^S(\pi_3)$; $P - Corr(\pi_2, \pi_3)$

M_2 : Oh, I was wrong, I am sorry; indeed, you can find book 'B' on the first floor, to the right. — $F^S(\pi_4)$; $Contrast(\pi_2, \pi_4)$; $P - Elab(\pi_3, \pi_4)$; $QAP(\pi_1, \pi_4)$

Dialogue 2:

U_1 : Sorry, can you tell me, please, where I can find book ‘B’? — $F^{FS}(\pi_1)$

M'_1 : Hello, wait a minute please... here it is: you can find this book on the first floor, to the left. — $F^S(\pi'_2); QAP(\pi_1, \pi'_2)$

U_2 : But I’ve just looked there, and I couldn’t find the book! — $F^S(\pi_3); P - Corr(\pi'_2, \pi_3)$

M'_2 : Listen, you can certainly find book ‘B’ on the first floor, just to the left, I have just checked in my data base! — $F^S(\pi'_4); P - Corr(\pi_3, \pi'_4); QAP(\pi_1, \pi'_4)$

Next to each speech turn in these dialogues we have marked the speech act type, the label of the utterance and the rhetorical relations that connect this speech turn to previous turns⁵ Moreover, for M ’s speech turns, we have marked the illocutionary force degree as well. The rhetorical relations are computed in the framework of (Asher and Lascarides, 2003) SDRT, whereas the speech act types are assigned based on linguistic markers (for user utterances), using (Colineau, 1997)’s connectionist approach, or provided directly by the dialogue controller (for machine utterances) (Caelen and Xuereb, 2007). By consequence, we limit our description to the commitment stores updating, and to the manner whereby the commitments allow us to compute the illocutionary force degrees. These processes are presented in detail below, for each of the two dialogues:

Dialogue 1:

1. when U produces π_1 , its commitment store remains unchanged, and so does M ’s commitment store: $CS_U \leftarrow_{\pi_1} CS_U \wedge CS_M \leftarrow_{\pi_1} CS_M$;
2. when M produces π_2 , its commitment store is updated, whereas U ’s commitment store remains, for the moment, unchanged: $CS_M \leftarrow_{\pi_2} CS_M \cup \{K(\pi_2), \Sigma_{QAP(\pi_1, \pi_2)}\} \wedge CS_U \leftarrow_{\pi_2} CS_U$; M ’s goal is that U update its commitment store with $K(\pi_2)$ as well: $CS_U \leftarrow_{\pi_2} CS_U \cup \{K(\pi_2)\}$, hence M assigns a value of +1 to the illocutionary force degree of π_2 : $\partial\phi(\pi_2) = +1$;
3. when U produces π_3 , its commitment store is updated by adding $\neg K(\pi_2)$, whereas M ’s commitments do not change for the moment: $CS_U \leftarrow_{\pi_3} CS_U \cup \{\neg K(\pi_2)\}$, $\Sigma_{P-Corr(\pi_2, \pi_3)} \wedge CS_M \leftarrow_{\pi_3} CS_M$;
4. when M produces π_4 , its commitment store is updated by removing $K(\pi_2)$ from it, and by adding $K(\pi_4)$ to it, whereas U ’s commitments do not change for the moment: $CS_M \leftarrow_{\pi_4} CS_M \setminus \{K(\pi_2)\} \cup \{K(\pi_4), \Sigma_{Contrast(\pi_2, \pi_4)}, \Sigma_{P-Elab(\pi_3, \pi_4)}, \Sigma_{QAP(\pi_1, \pi_4)}\}$; M ’s goal is that U ’s commitment store be updated by adding $K(\pi_4)$: $CS_U \leftarrow_{\pi_4} CS_U \cup \{K(\pi_4)\}$ such that $K(\pi_4) \Rightarrow \neg K(\pi_2)$, hence M assigns a value of -2 to the illocutionary force degree of π_4 : $\partial\phi(\pi_4) = -2$.

Dialogue 2:

1. when U produces π_1 , its commitment store remains unchanged, and so does M ’s commitment store: $CS_U \leftarrow_{\pi_1} CS_U \wedge CS_M \leftarrow_{\pi_1} CS_M$;

⁵For the simplicity of the presentation, we consider that each speech turn contains only one utterance.

2. when M produces π'_2 , its commitment store is updated, whereas U 's commitment store remains unchanged for the moment: $CS_M \leftarrow_{\pi'_2} CS_M \cup \{K(\pi'_2), \Sigma_{QAP}(\pi_1, \pi'_2)\} \wedge CS_U \leftarrow_{\pi_2} CS_U$; M 's goal is that U 's commitment store be updated with $K(\pi'_2)$ as well: $CS_U \leftarrow_{\pi_2} CS_U \cup \{K(\pi'_2)\}$; but first, M has to update its own commitment store with $K(\pi'_2)$, hence M assigns a value of -1 to the illocutionary force degree of π_2 : $\partial\phi(\pi_2) = -1$ (unlike in *Dialogue 1*, M does not have $K(\pi'_2)$ in its commitment store, hence it has to ask the task manager for this information);
3. when U produces π_3 , its commitment store is updated by adding $\neg K(\pi'_2)$, whereas M 's commitments do not change for the moment: $CS_U \leftarrow_{\pi_3} CS_U \cup \{\neg K(\pi'_2), \Sigma_{P-Corr}(\pi_2, \pi_3)\} \wedge CS_M \leftarrow_{\pi_3} CS_M$;
4. when M produces π'_4 , its commitment store still contains $K(\pi'_2)$, whereas U 's commitments do not change for the moment: $CS_M \leftarrow_{\pi'_4} CS_M \cup \{\Sigma_{P-Corr}(\pi_3, \pi'_4), \Sigma_{QAP}(\pi_1, \pi'_4)\}$; M 's goal is that U 's commitment store be updated by removing $\neg K(\pi'_2)$, and by adding $K(\pi'_2)$: $CS_U \leftarrow_{\pi'_4} CS_U \setminus \{\neg K(\pi'_2)\} \cup \{K(\pi'_2)\}$, hence M assigns a value of $+2$ to the illocutionary force degree of π'_4 : $\partial\phi(\pi'_4) = 2$.

For these two dialogues, we have been in two different cases, concerning the (set theoretic) relation between U and M 's commitment stores:

- In dialogue 1 we are in the case $CS_U \cap CS_M = CS_U \neq \emptyset$ after speech turn M_2 were produced;
- In dialogue 2 we are in the case $CS_U \cap CS_M = \emptyset$ after speech turn M'_2 were produced.

If the degree of illocutionary force had not been adjusted, then we would have obtained in all cases a single speech turn, $M^{(0)}$, instead of M_2 , or M'_1 and M'_2 ; speech turn $M^{(0)}$ would have contained an utterance like:

$M^{(0)}$: You can find book 'B' on the first floor, to the left.

This would have led the machine to produce more annoying and less natural turns.

However, the approach has several limits. First, the machine is sometimes "rude" for a public service, especially when choosing very strong degrees of force for determining its interlocutor to give up some of her/his commitments. However, such as behavior is not totally implausible, if we think at more informal interactions, especially those that are less subject to politeness conventions: consider for example a dialogue situation between a bartender and a (drunk) customer in a bar next to a highway at night, where very strong illocutionary degrees and behaviors such as those shown in the article seem casual. Yet, there is another problem, that stems from the imprecisions in determining the interlocutors' public commitments: indeed, if, for some reason (i.e. sudden attack) the interlocutor stops in the middle of its utterance, or the interlocutor tries to tease the system by feeding incoherent turns into it, the machine might perform wrong calculations, and hence, for instance, inadequately produce utterances with a $+1$ (strong) degree of illocutionary force, in trying to determine the interlocutor to commit to something to which the latter has already committed, or which is irrelevant to the dialogue at hand; these limitations are discussed more thoroughly in (Popescu et al., 2009).

4. Discussion and Conclusions

In this article we have shown that fine-tuning the degree of illocutionary force is a crucial aspect in generating (machine) utterances in dialogue. We have proposed a computational framework for computing the illocutionary force degree. The mechanism is based on the public commitments of the speakers (the machine, which emulates a public service, and a human customer of this service). The commitment stores are computed from the discourse structure in a manner borrowed from (Maudet *et al.*, 2006). Furthermore, we have given a precise formalization of Vanderveken’s notion of illocutionary force degree, coupling it with the public commitment, in a condition on dialogue success. Although limited and arguable, this condition (of aiming at identical commitments for the interlocutors) is relevant in service-oriented dialogues, that are restrained to a very specific task, priorly specified.

However, the approach described in this article has certain other limits and arguable aspects. First, out of the six components of the illocutionary force, as defined by (Vanderveken, 1990-1991), we take into account explicitly only three: propositional content, degree, and illocutionary point (i.e., speech act type). Another arguable point, although in agreement with (Vanderveken, 1990-1991): p. 120, is the domain of values for the illocutionary force degree: $\{-2, -1, 0, 1, 2\}$; Vanderveken does not limit the number of possible illocutionary force degree, but we do it, on the one hand because the five-level scale emerges from the relationships between the public commitments of the speakers, and on the other hand, for methodological reasons, related to the possibility to conveniently annotate a set of canned phrases that can be used in surface generation.

Another possible criticism concerns the manner whereby we define the negation of an utterance, especially when this utterance is a question. One could object that an interrogative utterance is of the wrong semantic type for the negation operation. This is why we need to clarify this point. Let us consider for example a question as ‘Is this book OK for you’, labeled π . At a semantic level, this utterance is logically represented via the $K/1$ function. Since it is a question, the utterance contains a predicate which takes a non-initialized variable as argument:

$$\exists Y, Z : \text{object}(\text{'book'}) \wedge \text{feature}(\text{'book'}, Y) \wedge \text{title}(Y) \wedge \text{equals}(Y, \langle \text{book_title} \rangle) \wedge \text{want}(\neg\text{emitter}(\pi), \text{'book'}, Z) \wedge \text{equals}(Z, \text{'?'}).$$

Here, the non-initialized variable is the boolean Z that contains the truth value of the predicate $\text{want}/3$, which is true if the entity designated by its first argument (in our case, the recipient of π) wants the entity designated by the second argument (in our case, the book ‘book’, whose title is specified by the value of variable Y). The negation of such a question does not boil down to the classical negation of each predicate in the conjunction, followed by the substitution of the conjunctions with disjunctions, but to assigning the value 0 to the boolean Z ; hence, in our case, $\neg K(\pi)$ has the same form as $K(\pi)$, excepting the last predicate, which has the form $\text{equals}(Z, 0)$.

Another aspect left untackled in this article concerns the importance of the conversational genre in determining relevant illocutionary force degrees for the utterances. Thus, for instance, in absurd theatrical plays very strong illocutionary force degrees can be appropriate

even in social configurations where the speaker has a lower position on the social hierarchy, with respect to the hearer, simply because ethical adequacy is sometimes distorted. Moreover, in such plays (or in informal dialogues between close friends, for instance), a very strong illocutionary force degree might as well be used for withdrawing a commitment, in an utterance like ‘Listen, I was certainly wrong, I am really sorry, do you understand me?’. However, in the service oriented customer-institution dialogues concerned by this study, such illocutionary configurations seem rather inadequate, at least in the context of dialogue corpora for task-oriented dialogues, such as the PVE (“Portail Vocal pour l’Entreprise”) system (Nguyen, 2005). In our view, genre acts in the form of supplementary constraints in human-system dialogues. These constraints can perform a post-filtering on the set of authorized illocutionary forces, in a manner akin to that discussed for social roles, in the end of Section 3.2. Nevertheless, the issue of formalizing the intricate interaction between social and genre constraints in producing dialogue contributions that are adequate from an illocutionary standpoint remains, in our view, open to further study.

In spite of several limitations of the current framework, it represents a first version of a principled way of computing the adequate illocutionary force degree of machine utterances. A first prospect for this research would be to evaluate in quantitative manner the framework described, by comparing automatically generated utterances to human utterances generated via Wizard-of-Oz techniques, or present in real dialogue corpora.

Acknowledgement

The authors wish to thank Prof. Andrew Kehler from the University of California, San Diego, for his valuable comments and thorough remarks concerning the contents of this article.

Bibliography

- Asher, Nicholas and Alex Lascarides. *Logics of Conversation*. Cambridge University Press, United Kingdom, 2003.
- Asher, Nicholas and Alex Lascarides. Making the right commitments in dialogue. In *Fall 2008 Workshop in Philosophy and Linguistics*. University of Michigan, 2008.
- Brown, Penelope and Stephen Levinson. *Politeness: Some Universals in Language Use*. Cambridge University Press, UK, 1987.
- Caelen, Jean and Anne Xuereb. *Interaction et pragmatique – jeux de dialogue et de langage*. Editions Hermès - Lavoisier, Paris, France, 2007.
- Cohen, Philip and Hector Levesque. Persistence, intention and commitment. In Cohen, Philip, Jerry Morgan, and Martha Pollack, editors, *Intention in Communication*, pages 33–69. MIT Press, 1990a.
- Cohen, Philip and Hector Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42 (2-3):213–261, 1990b.
- Cohen, Philip and Raymond Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.

- Colineau, Nathalie. *Etude des marqueurs discursifs dans le dialogue finalisé*. Ph D Thesis, Joseph Fourier University, Grenoble, <http://tel.archives-ouvertes.fr/tel-00004928/en/>, 1997.
- Faller, Martina. Evidentiality and epistemic modality at the semantics / pragmatics interface. In *Fall 2006 Workshop in Philosophy and Linguistics*. University of Michigan, 2006.
- Gupta, Swati, Marilyn Walker, and Daniela Romano. Generating politeness in task based interaction: An evaluation of the effect of linguistic form and culture. In *Proceedings of the 11th European Workshop on Natural Language Generation ENLG'07*, Schloss Dagstuhl, 2007.
- Kibble, Rodger. Speech acts, commitment and multi-agent communication. *Computational and Mathematical Organization Theory*, 12:127–145, 2006a.
- Kibble, Rodger. Reasoning about propositional commitments in dialogue. *Research on Language and Computation*, 4:179–202, 2006b.
- Kibble, Rodger. Generating coherence relations via internal argumentation. *Journal of Logic, Language and Information*, 16:387–402, 2007.
- Kibble, Rodger and Paul Piwek. Introducing dialogue games. In *Lecture Notes, ESSLLI 2007*. FoLLI, Dublin, Ireland, 2007.
- Lascarides, Alex and Nicholas Asher. Agreement, disputes and commitments in dialogue. *Journal of Semantics*, 26(2):109–158, 2009.
- Maudet, Nicolas, Philippe Muller, and Laurent Prévot. Tableaux conversationnels en SDRT. In *Proceedings of the SDRT Workshop, Traitement Automatique des Langues Naturelles*, Fès, Maroc, 2004.
- Maudet, Nicolas, Philippe Muller, and Laurent Prévot. Social constraints on rhetorical relations in dialogue. In *Workshop Constraints in Discourse*, Maynooth, Ireland, 2006.
- Motzsch, Wolfgang. Situational context and illocutionary force. In Searle, John, Ferenc Kiefer, and Manfred Bierwisch, editors, *Speech Act Theory and Pragmatics*, pages 155–168. D. Reidel Publishing Company, 1980.
- Nguyen, Hoá. *Dialogue homme-machine : modélisation de multisession*. Ph D Thesis, Joseph Fourier University, Grenoble, <http://tel.archives-ouvertes.fr/tel-00008789/en/>, 2005.
- Popescu, Vladimir and Jean Caelen. Argumentative ordering of utterances for language generation in multi-party human-computer dialogue. *Argumentation*, 23(2):205–237, 2009.
- Popescu, Vladimir, Jean Caelen, and Corneliu Burileanu. Contrôle rhétorique de l'ellipse sémantique en génération du langage pour le dialogue homme-machine à plusieurs locuteurs. *Traitement Automatique des Langues*, 49(1):115–139, 2008.
- Popescu, Vladimir, Jean Caelen, and Corneliu Burileanu. A constraint satisfaction approach to context-sensitive utterance generation in multi-party dialogue systems. *International Journal of Speech Technology*, 12(2-3):95–112, 2009.
- Searle, John and Daniel Vanderveken. Speech acts and illocutionary logic. In Vanderveken, Daniel, editor, *Logic, Thought and Action*, volume 2 of *Logic, Epistemology and the Unity of Science*, pages 109–132. Springer, 2006.
- Traum, David, Michael Fleischman, and Eduard Hovy. NL generation for virtual humans in a complex social environment. In *Proceedings of the AAAI 2003 Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*, Palo Alto, 2003.

Vanderveken, Daniel. *Meaning and Speech Acts*. Cambridge University Press, United Kingdom, 1990-1991.

Xuereb, Anne and Jean Caelen. Actes de langage et relations rhétoriques en dialogue homme-machine. *Revue de l'Université de Moncton*, 36(2):5-51, 2005.

Address for correspondence:

Vladimir Popescu
vladimir.popescu@imag.fr
339, chemin des Meinajaries
Agroparc BP 91228
84911 Avignon Cedex 9
France



The Prague Bulletin of Mathematical Linguistics
NUMBER 94 SEPTEMBER 2010 57-66

MT Server Land: An Open-Source MT Architecture

Christian Federmann, Andreas Eisele

DFKI, German Research Center for Artificial Intelligence

Abstract

We describe the implementation of *MT Server Land*, an open-source architecture for machine translation that is developed by the MT group at DFKI. A *broker* server collects and distributes translation requests to several *worker* servers that create the actual translations. Users can access the system via a fast and easy-to-use web interface or use an XML-RPC-based API interface to integrate it into their applications. The source code is published under a BSD-style license and is freely available from GitHub¹.

1. Introduction

Easy-to-use machine translation (MT) services that are available via the internet are an important means to increase visibility of MT research and to help shaping the multi-lingual web. Applications such as *Google Translate* allow lay users to quickly and effortlessly create translations of texts or even complete web pages; the continued success of such services shows the potential that lies in *usable* machine translation, something both developers and researchers should strive for.

Despite impressive progress in recent times, MT can by far not be regarded as a solved problem, and the ongoing research on many levels requires careful analysis of existing systems that may vary along many dimensions or that may be hybrid solutions composed from building blocks taken from different paradigms. A significant number of existing systems from ongoing research projects should be made available to researchers from the field for a couple of reasons.

¹You can download a copy of the code at <http://github.com/cfedermann/mt-serverland>.

For one, the ease of comparative evaluation would advance the understanding of merits and weaknesses and hence facilitate progress towards higher quality in MT. But the easy availability of systems would also allow researchers and developers from related areas to use MT functionality as building blocks in a larger context. Areas that would benefit most from this include efforts towards computer-aided translation (CAT) platforms, cross-lingual search and question answering, easy deployment of multilingual websites, knowledge acquisition from multilingual document repositories, and many more.

Beyond such groups, also decision makers from language industry and large organisation that are potential users of MT functionality should be given easy access to the existing functionality in order to allow them to judge the potential of such systems for specific applications.

Last not least, the general public, who often takes the offerings of service providers like Google or Microsoft to be representative of the current state of the art in MT, should be given a chance to compare these services against the functionality provided by ongoing research. In the context of ongoing MT research projects at DFKI's language technology lab, such as EuroMatrixPlus, ACCURAT or TaraXÜ, we have decided to design and implement such a translation application. We have published the source code as open-source and hope that it becomes a useful tool for the MT community.

2. Scope and Requirements

Considering some intended usages of the toolkit, we have collected a set of requirements our software should meet. We are planning for a staged delivery, where subsequent releases of the software will meet an increasing number of the requirements and where the priorities concerning the next round will be determined based on experience collected with active usage of the system as it was already delivered, in a set of realistic applications. The requirements can be grouped into core functionalities, important extensions, and features that would be useful in advanced applications.

Core Functionalities: A central requirement for the toolkit is to provide a single entry point to multiple MT engines for multiple users. The system should also support multiple language pairs and multiple MT engines per language pair, including different types of engines (SMT, RBMT, hybrid MT) and multi-engine setups, as well as variants of systems optimized for multiple application domains, text types, and styles. The system should provide access both via user-friendly, web-based interaction, as well as programmatically via a simple, yet powerful API such as a Web service.

Important Extensions: The system should allow to assign appropriate roles to each user (e.g. not every user should have access to every system, some user may have priority over others, etc.). The system should support many concurrent translation requests and multiple installations of the engines on different computers. It

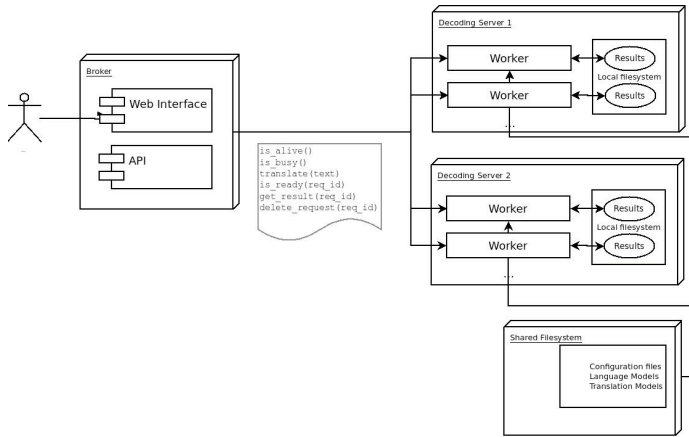


Figure 1. Overview of the System Architecture

should make sure that work is distributed over available resources via queuing and load balancing. The system should be able to recognize and handle exceptional circumstances caused by failure of engines and communication. The system should minimize the required administrative effort, even under heavy load.

Advanced Features: The system should be able to pass not only translation in- and output between users and MT engines, but also additional data generated by the engines, such as alignments, results of intermediate processing steps, as far as the engines are able to generate these. It should allow users to pass in additional information to the engines that will allow the engines to adapt to the needs of the each user (personalization, incremental training). Furthermore, it should provide auxiliary functionality, such as splitting of longer documents into paragraphs and sentences, tokenisation, case normalisation.

3. System Architecture

In this section, we will give an overview on the system’s general architecture and the several components it is composed of. Figure 1 shows a bird’s-eye view on the MT Server Land application. A similar application has been described in (Victor M. Sanchez-Cartagena, 2010).

3.1. Overview

The system consists of two different layers: first, we have the *broker* server that handles all direct requests from end users or API calls alike. Second, we have a layer of so-called *worker* servers, each implementing some sort of machine translation func-

```

package serverland;

message TranslationRequestMessage {
  required string request_id = 1;           // Random UUID-4 32-digit hex number
  required string source_language = 2;     // ISO 639-2 language codes
  required string target_language = 3;
  required string source_text = 4;        // UTF-8 encoded texts
  optional string target_text = 5;

  message KeyValuePair {
    required string key = 1;
    required string value = 2;
  }

  repeated KeyValuePair packet_data = 6; // Contains additional request data
}

```

Figure 2. *TranslationRequestMessage* .proto definition

tionality. All communication between users and workers is channeled through the broker server which acts as a central “proxy” server. For users, both broker and workers “constitute” the MT Server Land application.

Human users connect to the system using any modern web browser, API access can be implemented using XML-RPC calls. It would be relatively easy to extend the API interface to support other protocols such as SOAP or REST. By design, all internal method calls that connect to the worker layer have to be implemented with XML-RPC. In order to prevent encoding problems with the input text, we send and receive all data encoded as Base64 Strings between broker and workers; the broker server takes care of the necessary conversion steps.

3.2. Broker Server

The broker server has been implemented using the *django web framework* which takes care of low-level tasks and allows for rapid development and clean design of components. We have used the framework for other project work before and think it is well suited to the task. More information on django can be found on the project website which is available at <http://www.djangoproject.com/>, the framework itself is available under an open-source BSD-license.

3.2.1. Translation Request Messages

Each translation request is defined by a unique *request id*, a *source* and *target* language as well as a *source text*. After the translation has been produced, the request will also contain the *target translation* and, for some worker implementations, *additional data* such as log files, alignment information or even parse trees that have been returned from the translation engine.

In order to allow flexible serialization of translation requests, we have implemented them using Google Protocol Buffers (Google, 2010b). Our `.proto` definition is shown in Figure 2, it can be compiled into Python code using the following command:

```
$ protoc --python_out=workers/ TranslationRequestMessage.proto
```

This will create a new Python file named `TranslationRequestMessage_pb2` inside the `workers/` folder of our MT Server Land application. Using protocol buffers allows to easily serialize Python instances to a binary representation and vice versa, something that has proven to be very useful during the development of the system.

3.2.2. Object Models

The broker server implements two main object *django models* which we describe below. Please note that we have also developed additional object models, e.g. for quota management or API access authentication. See the MT Server Land source code for more information.

A `WorkerServer` instance stores all information related to a remote worker server. This includes the respective hostname and port address as well as a name and a short description. In fact, this is just a shallow wrapper around the XML-RPC interface.

The `TranslationRequest` model represents an external translation job and related information such as the chosen worker server, the assigned request id and additional information about the creation date or the owner. We also prepare some fields for caching of translation request state. Please note that neither *source* nor *target* texts are stored within the `django` instance; instead they are kept in form of a serialized `TranslationRequestMessage` file which is named by the request id and stored in a configurable location on the broker server's hard disk.

3.2.3. User Interface

We developed a browser-based web interface to access and use the MT Server Land application. End users first have to authenticate before they can access their *dashboard* which lists all known translation requests for the current user and also allows to create new requests. Once a translation request has been completed by the chosen worker server, the result is transferred to the broker server's data storage, deleting the request data from the worker server. The user can view the result within the dashboard or download the file to a local hard disk. It is also possible to delete "pending" translation requests at any time, effectively terminating the corresponding thread within the connected worker server.

3.2.4. API Interface

In parallel to the browser interface, we have designed and started to implement an API that allows to connect applications to the MT functionality provided by our service using XML-RPC. Again, we first require authentication before any machine translation can be used. We plan to use so-called *auth tokens*, i.e. randomly generated 32-digit hexadecimal numbers which are bound to a certain user account, for this. We provide methods to list all requests for the current “user” (i.e. the application account) and to create, download, or delete translation requests. Extension to REST or SOAP protocols is possible. Again, serialized `TranslationRequestMessage` objects are used to exchange requests between the user’s application and the MT Server Land.

3.2.5. Starting the Broker Server

Like any other django project, the broker server can be started in *debug mode* using the python `manage.py runserver` command. For internal deployment of the system, we have used the *lighttpd web server* which is a lightweight, fast and open-source web server that can be easily combined with a django application. More information can be found on the project website which is available at <http://www.lighttpd.net/>. We have configured the web server to serve all django media files and send all other requests to the django FCGI server that runs in a background process. A sample server configuration file `lighttpd-django.conf` and `startup/stop` scripts for django’s FCGI mode are contained in the source code release package.

3.3. Worker Servers

Actual machine translation functionality is implemented by a layer of so-called worker servers that are connected to the central broker server. We have created a Python-based `AbstractWorkerServer` class which is the foundation for all worker implementations. The basic worker interface is described next.

Attributes: `finished`: Boolean that controls the main server loop. Defaults to `False`. `server`: The actual `SimpleXMLRPCServer` instance is bound here. `jobs`: Dictionary memorizing all translation requests the worker has accepted. Maps request ids as keys to `Process` objects that represent the actual worker threads. Request ids are random 32-digit hexadecimal UUID numbers.

General Methods: `__init__`: Constructor, takes care of setting up the logging and creates the actual XML-RPC server instance. `start_worker`: Starts the main server loop that handles requests. `stop_worker`: Sets `finished` to `True` and terminates all running translation processes. Intermediate results are lost, the file storage of the worker server should be cleaned afterwards to avoid keeping invalid requests.

Status Methods: `list_requests`: Returns a list of all registered translation request ids. `is_alive`: Returns `True` to signal that the worker server is up and running. `is_busy`: Checks whether the worker server is currently processing requests.

`is_ready`: Checks whether the request with the given request id is finished. `is_valid`: Checks whether the request id is valid, i.e. contained within jobs.

Translation Methods: `language_pairs`: Returns a read-only tuple containing tuples that encode the available language pairs which are supported by this translation engine. All languages are identified by ISO 639-2 codes². `language_code` Converts the given ISO 639-2 code into the internal representation of language codes used by the worker's translation engine. `start_translation`: Takes the given serialized `TranslationRequestMessage` object, creates a local copy inside the worker server's `/tmp/` folder and then starts a `Process` that calls the `handle_translation` handler. `fetch_translation`: Retrieves the translation result for the given request id if already available. Otherwise returns an empty `String`. `delete_translation`: Deletes the translation request with the given request id from the jobs dictionary, terminating the connected process if still running. `handle_translation`: Implements the actual translation functionality of a worker implementation. Custom worker servers need to overwrite this method.

3.3.1. Example: Implementing a Google Translate Worker

Worker servers can be implemented by subclassing `AbstractWorkerServer` and creating a custom `handle_translation` method. The listing in Figure 3 shows the actual code for a "Google worker" server that sends its input text to Google Translate and extracts the translation from the resulting website.

3.3.2. Worker Server Implementations

We have implemented worker servers for several MT systems:

- **Lucy RBMT**: our Lucy (Alonso and Thurmair, 2003) worker is implemented using an internal Lucy Server mode wrapper. Due to the system's architecture, this has to be run on a Windows machine. The actual worker code can be started on any platform.
- **Moses SMT**: a Moses (Koehn et al., 2007) worker is configured to serve exactly one language pair. We use the Moses Server mode to keep translation and language model in memory which helps to speed up the translation process.
- **Joshua SMT**: similar to the Moses worker, we have created a Joshua (Li et al., 2009) worker that works by creating a new Joshua instance for each translation request.

We have also created worker servers for popular online translation engines such as Google Translate, Microsoft Translator and Yahoo! Babel Fish which already makes available a huge number of language pairs for use in MT research contexts.

²See <http://www.loc.gov/standards/iso639-2/> for more information.

```

import re, sys, urllib, urllib2
from worker import AbstractWorkerServer
from TranslationRequestMessage_pb2 import TranslationRequestMessage

class GoogleWorker(AbstractWorkerServer):
    """ Implementation of a worker server that connects to Google Translate. """
    __name__ = 'GoogleWorker'

    def language_pairs(self):
        """Returns a tuple of all supported language pairs for this worker."""
        languages = ('af', 'alb', 'ara', ..., 'vie', 'wel', 'yid')
        return tuple([(a,b) for a in languages for b in languages if a != b])

    def language_code(self, iso639_2_code):
        """Converts a given ISO-639-2 code into the worker representation."""
        mapping = { 'af': 'af', 'alb': 'sq', ... 'wel': 'cy', 'yid': 'yi' }
        return mapping.get(iso639_2_code)

    def handle_translation(self, request_id):
        """Translation handler that connects to Google Translate."""
        handle = open('/tmp/{0}.message'.format(request_id), 'r+b')
        message = TranslationRequestMessage()
        message.ParseFromString(handle.read())

        source = self.language_code(message.source_language)
        target = self.language_code(message.target_language)
        the_url = 'http://translate.google.com/translate_t'
        the_data = urllib.urlencode({'js': 'n', 'sl': source, 'tl': target,
            'text': message.source_text.encode('utf-8')})
        the_header = {'User-agent': 'Mozilla/5.0'}

        opener = urllib2.build_opener(urllib2.HTTPHandler)
        http_request = urllib2.Request(the_url, the_data, the_header)
        http_handle = opener.open(http_request)
        content = http_handle.read()
        http_handle.close()

        result_exp = re.compile('<textarea name=utrans wrap=50FT ' \
            'dir="ltr" id=suggestion.*?(.*?)</textarea>', re.I|re.U)
        result = result_exp.search(content)

        if result:
            message.target_text = unicode(result.group(1), 'utf-8')
            handle.seek(0)
            handle.write(message.SerializeToString())

        handle.close()

```

Figure 3. Source code for the Google Translate worker

4. Basic Usage

The MT Server Land code can be obtained from GitHub and extracted to a local folder named `serverland/` using the following command:

```
$ git clone git://github.com/cfedermand/mt-serverland.git serverland
```

After downloading the source code, we need to create a database for the project. This can be done using the `manage.py syncdb` command, as shown below:

```
$ python manage.py syncdb
```

It is mandatory to create a superuser account during the `syncdb` step. We also provide a sample `development.db` file with a sample user `admin:admin` at the GitHub repository³. It is now possible to startup `django` in development using `manage.py runserver`, as we have already mentioned. However, before any translation work can be done, at least a single worker server instance has to be started and registered inside the `django` database.

The available worker server implementations can be found inside `workers/`. We also provide scripts to start and stop worker server instances. To startup the Google Translate worker server, we have to start it using the following command:

```
$ ./start_worker.py GoogleWorker localhost 1234
```

This will create a new `GoogleWorker` instance serving from `http://localhost:1234/`. In order to make this worker instance accessible from the MT Server Land system, we have to register it inside the broker server's database. For this, we access the `django` administration backend (which is available at `http://127.0.0.1:8000/admin/`) and create a `WorkerServer` object pointing to the correct host and port address. After the worker server has been created, authenticated users can create new translation requests which are then processed by the respective worker server.

5. Conclusion and Future Work

We have presented an open-source architecture for machine translation. The system can flexibly be extended and allows lay users to make use of MT technology within a web browser or by using XML-RPC method calls from custom applications. A central broker server receives requests from clients and dispatches them to a layer of worker servers that take care of the translation duties. We have used open-source software to build the system and have released the source code under a BSD-style license.

5.1. Open-Source Development

We hope that the MT Server Land software will benefit from and grow by being maintained as an open-source project. We have opted for hosting at the GitHub platform as this guarantees transparent development and ensures open access to the

³At <http://github.com/downloads/cfedermann/mt-serverland/mt-serverland-development.db>

source code. We continue to extend the MT Server Land code and available worker servers, possibly starting at the Machine Translation Marathon in Le Mans for which we are currently preparing project ideas related to the MT Server Land platform.

Acknowledgments

We would like to thank all members of the MT Group at DFKI for testing the MT Server Land prototype and for all their helpful feedback during the development of this software. This work was supported by the EuroMatrixPlus project (IST-231720) which is funded by the European Community under the Seventh Framework Programme for Research and Technological Development.

Bibliography

- Alonso, Juan A. and Gregor Thurmair. The Compendium Translator system. In *Proceedings of the Ninth Machine Translation Summit*, New Orleans, USA, 2003.
- Google. Google Translate, 2010a. URL <http://translate.google.com/>.
- Google. Google Protocol Buffers, 2010b. URL <http://protobuf.googlecode.com/>.
- Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P07-2045>.
- Li, Zhifei, Chris Callison-Burch, Chris Dyer, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece, March 2009. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W/W09/W09-0x24>.
- Microsoft. bing Translator, 2010. URL <http://www.microsofttranslator.com/>.
- Victor M. Sanchez-Cartagena, Juan Antonio Perez-Ortiz. ScaleMT: a Free/Open-Source Framework for Building Scalable Machine Translation Web Services. In *Open Source Tools for Machine Translation, MT Marathon 2010*, Dublin, Ireland, 2010.
- Yahoo! Yahoo! Babel Fish, 2010. URL <http://babelfish.yahoo.com/>.

Address for correspondence:

Christian Federmann
cfedermann@dfki.de
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, GERMANY



The Prague Bulletin of Mathematical Linguistics
NUMBER 94 SEPTEMBER 2010 67-76

**CorporAI: a Method and Tool
for Handling Overlapping Parallel Corpora**

Mark Fishel, Heiki-Jaan Kaalep

Institute of Computer Science, University of Tartu

Abstract

This work introduces a method and tool for handling overlapping parallel corpora – i.e. corpora that are based on the same source material. The method is insensitive to minor changes in the text, different segmentation levels of the corpora and omitted material from either corpora. The aim is to detect matching sentence pairs and either produce combinations of the overlapping corpora or compare them and assess their quality in comparison to each other. The introduced tool enables the user to define the desired behavior when combining corpora pairs, resulting in pure comparison, maximum-size or maximum-quality versions of the combinations. We test the tool on two cases of overlapping parallel corpora and five language pairs. We also evaluate the impact of using the method on two translation systems – a phrase-based and a parsing-based one.

1. Introduction

The target of this research is parallel corpora that are based on partially or fully overlapping sources of the same language pair – overlapping parallel corpora. Such corpora can exist, for instance, when the same source documents are independently used to create corpora at different times or different institutions.

Processing such corpora can be quite problematic. Simply concatenating them is not a valid solution, since the data distribution of the combined corpus will be skewed. At the same time using the standard `diff` utility is not guaranteed to elegantly solve the problem of detecting the repeated and unique samples. Typically the texts have differences in representation, or some typing or aligning errors fixed or introduced in one of the corpora. In addition sentence pairs might be segmented differently in the two corpora or be omitted from one of them.

On the other hand, if those difficulties could be overcome, the overlap could be exploited to many advantages. By comparing the two corpora the level of segmentation of both can be increased, the potential alignment error spots can be found and the size of both can be increased on the account of omitted sentence pairs from one or the other corpus. Finally, if it can be assumed that one of the corpora is much more accurate, the other corpus can be proofed against it to evaluate or improve its quality.

Here we present a method that can be used to do all of the tasks mentioned above, together with its implementation. We apply the method to two cases of overlapping parallel corpora and evaluate its influence on the scores of statistical translation systems, trained on the resulting corpora.

2. Overlapping Parallel Corpora

Let us first look at some examples of overlapping parallel corpora.

(Kaalep and Veskis, 2007) compare the JRC-Acquis corpus (version 2.2) (Steinberger et al., 2006) and the corpus of the University of Tartu¹. The latter also includes Estonian laws with their English translations, in addition to the EU legislation. To our knowledge (Kaalep and Veskis, 2007) is the only work addressing the issue of overlapping parallel corpora.

Another example is the JRC-Acquis corpus itself, since it provides two alternative alignments for every language pair it includes – done with Vanilla² and HunAlign (Varga et al., 2005). This means that, although the text might be exactly the same, the level of segmentation can be different in the two versions. In addition, it is common practice for aligners to exclude sentence pairs in which they are not confident enough.

In the experimental part of this work we focus on the two presented cases; however there are other examples as well. The Hunglish corpus (Varga et al., 2005) includes EU legislation, obtained from the same sources as the JRC-Acquis. One part of the CzEng corpus (Bojar and Žabokrtský, 2009) also consists of EU legislation, whereas the source documents were taken directly from JRC-Acquis, but the text processing and alignment was done all over. Also a whole domain of corpora is a potential source for multiple versions of the same text – movie subtitles.

3. Method Description

Let us start with an example of two parallel corpora containing an overlap (figure 1). The third sentence pair of corpus B is omitted from corpus A and the third sentence pair of corpus A is segmented into two sentence pairs in corpus B. Also there are slight differences in punctuation between the two corpora.

¹<http://www.cl.ut.ee/korpused/paralleel/?lang=en>

²<http://nl.ijs.si/telri/Vanilla/>

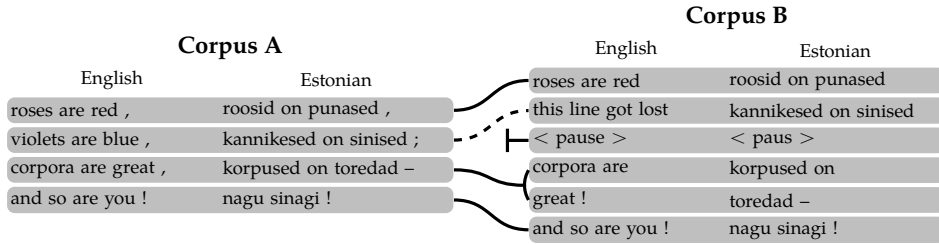


Figure 1. An example of overlapping parallel corpora with the correspondence of the two corpora shown. Second sentence pair of corpus B is an erroneous alignment.

Knowing both English and Estonian, it is easy to see that the English sentence from second sentence pair in corpus B got distorted, which makes the pair an erroneous alignment. Without knowing either of the languages, it can still be detected that one of the second sentence pairs in both corpora is probably erroneous – since the Estonian parts are practically the same, while the English parts are nothing like each other. Very simply put, this language-wise comparison is the basis of the method that we are about to introduce.

The method involves two steps. The first step consists of aligning the corresponding language parts to each other; see figure 2 (a) for an illustration. In the second step the resulting language alignments are themselves aligned to each other. Here the aim is to find the matching and mismatching alignment chunks. This way whenever in one language two sentences match while in the other language the corresponding sentences do not, this will be detected as an alignment error. See figure 2 (b) for an illustration of the second step; notice the resemblance between the resulting alignment and the correspondence of the parallel corpora in the example on figure 1.

In the following subsections we will describe in detail the two steps of the algorithm, as well as sentence approximate matching.

3.1. Aligning the Corresponding Language Parts

The first step is in essence very similar to the original task of bilingual sentence alignment itself. However, whereas the latter means comparing different languages and therefore requires, for instance, probabilistic solutions, in this case the task is much simpler, since both parts are in the same language and it suffices to compare the sentences using simple text processing. The only problem is that instead of strict comparison of the sentences, here approximate comparison is required due to possible slight differences in different corpora.

The aligning task is therefore analogical to the longest common subsequence problem, where corpora units (i.e. sentences or paragraphs) are matched to each other.

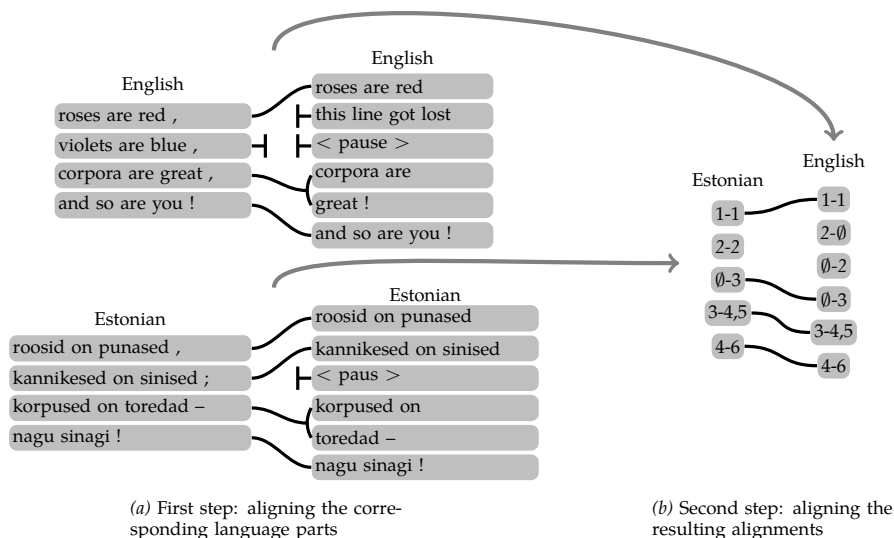


Figure 2. The two steps of processing the overlapping parallel corpora from the example on figure 1.1. \emptyset stands for an empty counterpart (in zero-to-one alignments).

Here the alignment of the two texts is computed using generalized edit distance. The cost of substituting a unit for another equals the similarity between them (obtained with approximate sentence matching, explained in the next subsection). In addition all N-to-M pairs are also considered (up to a predefined limit). This enables matching aligned units even if the segmentation level is very different in the two corpora.

3.2. Approximate Sentence Matching

(Kaalep and Veskis, 2007) use Levenshtein distance with 1% of the average of the two sentence's length as a threshold. Other string similarity metrics applied to written text include several from the edit distance family (the Needleman-Wunsch metric, the Smith-Waterman metric, etc), the Jaro metric and others.

Here we use the method of (Kaalep and Veskis, 2007), extended to generalized edit distance. For instance the weight of replacing/inserting digits is extremely high, so that e.g. sentences "article 3" and "article 5" will not be considered to match with no matter what edit distance percentage threshold. On the other hand operations on empty symbols (spaces, tabs) and punctuation have low weights. This allows to set the percentage threshold higher without adding obvious matching errors.

3.3. Aligning the Alignments

As soon as the language part alignments are obtained, their correspondence to each other is to be determined. Although different language parts are to be compared here, only the alignments between unit numbers are compared, which again enables using direct comparison. In this case it is accomplished with the Levenshtein distance of the alignment cells.

It is important to note that a mismatch between two alignments does not indicate, which of the corpora has an erroneous alignment; instead, it shows a potential spot, where at least one of the corpora has an error. If one of the corpora is known to be accurately aligned, the errors of the other corpus can be corrected automatically this way. Otherwise the spots can be manually post-processed and the errors in the appropriate corpus – corrected.

On the other hand a match between alignments also merely indicates that the two corpora have matching alignments. This can occur both in case of correct alignments and coinciding erroneous alignments, though the latter is less likely (depending on the used alignment method).

3.4. Implementation – the CorporAl Tool

The CorporAl open-source project is available from Sourceforge³. The implementation is done as a PERL script and thus can be run on any platform with a PERL interpreter; the interface of the tool is command-line-based.

The tool name is meant to reflect the core idea of the method – “aligning” the corpora to each other. Using the alignment between the corpora the tool generates a new combined corpus. The exact behaviour can be controlled with input parameters: whether to include or exclude sentences from the unique and the matching parts, whether to skip mismatched sentence pairs or define one of the corpora as the more trustworthy one and include sentences from it. If sentence pairs match, the side with a higher level of segmentation is automatically included. Also it is possible to just output the alignment of the corpora to be used for further processing.

The main direction of further development of CorporAl is extending it to support monolingual corpora with annotation, in addition to parallel corpora. If the two overlapping corpora are augmented with the same annotation then both the text and the annotation can be compared, just like the two language parts of parallel corpora.

Alternatively, if the annotations differ, only the text can be matched and not the annotation. As a result the tool would allow to produce a text corpus with both annotations, regardless of differences of the texts. Also it could be applied to parallel corpora where all languages are aligned to one, like Europarl (Koehn, 2005), to produce a corpus of any two languages without re-applying the aligners. This just requires making the alignment of the annotation or second part of a parallel corpus optional.

³<http://corporal.sf.net>

4. Experiments

Our final aim was to test the presented method in practice. We focused the experiments on two cases of overlapping parallel corpora, described in section 2: first, the corpus of the University of Tartu (UT) and the English-Estonian (en-et) part of JRC-Acquis version 2.2 (JRC2) and second, the HunAlign and Vanilla versions of the English-Estonian (en-et), English-Latvian (en-lv), Estonian-Latvian (et-lv) and German-English (de-en) parts of JRC-Acquis version 3 (JRC3). First we present the results of processing the corpora and then go on to testing the effect of our method on statistical translation systems.

4.1. Processing Overlapping Parallel Corpora

We first grouped the documents in all corpora by their CELEX codes, which resulted in three groups: documents unique to one of the corpora and the ones present in both corpora in a pair. Then the common parts of the corpora were processed with the CorporAl tool. We generated two different versions of the combination: one (called max-size) prioritized the resulting corpus size and the other one (called max-accuracy) prioritized the resulting accuracy – the latter thus included only the matching sentence pairs, present in both corpora.

The sizes of the documents and the resulting corpora parts are presented in Table 1 and the frequencies of the types of sentence pair matches – in Table 2.

Looking at the match type frequencies it can be seen that the many-to-one matches constitute just a small percent of all the matches (below 1% on both sides). Thus, contrary to our initial assumption, the levels of segmentation of the UT and JRC2 corpora overlapping parts are practically the same. The same goes for the JRC3 pairs, where the total percentage of many-to-many alignments is even lower.

An interesting observation about the JRC3 pairs is the difference between the documents included only in the Vanilla or HunAlign versions. It can be seen in Table 1 that while the HunAlign versions of all the four pairs include only three to five documents that are not included in the Vanilla versions, the total numbers of words and sentence pairs in these documents are much higher than their counterparts in the Vanilla versions. In addition the total sizes of the common parts of the HunAlign versions are bigger than the same document sets of Vanilla versions. These two facts might indicate that in the HunAlign version documents and sentences were more confidently included into the corpus than in the Vanilla versions.

As a result of similarity of the JRC3 pairs the max-size combinations are practically of the same size as the bigger HunAlign common parts (with only 100-150 extra sentence pairs). The max-size combination of UT and JRC2 is visibly bigger than both corpora. On the other hand the max-accuracy combinations are slightly smaller than the source corpora in all five cases, which is caused by the portion of mismatching and omitted sentence pairs.

UT+JRC2		#docs UT/JRC	#snt pairs UT/JRC ($\cdot 10^3$)	#lang-1 words UT/JRC ($\cdot 10^6$)	#lang-2 words UT/JRC ($\cdot 10^6$)
en-et	Unique	2048/5807	134.7/205.0	3.12/4.86	2.17/3.25
	Common	2009	93.2/68.2	1.9/1.7	1.3/1.1
	Max-size	2009	98946	2.03	1.36
	Max-acc	2009	56234	1.35	0.88
JRC3		#docs Hun/Van	#snt pairs Hun/Van ($\cdot 10^3$)	#lang-1 words Hun/Van ($\cdot 10^6$)	#lang-2 words Hun/Van ($\cdot 10^6$)
en-et	Unique	5/173	63.5/8.4	0.80/0.28	0.73/0.22
	Common	23181	1247.3/1183.9	31.26/31.12	22.49/22.29
	Max-size	23181	1247.4	31.26	22.49
	Max-acc	22512	1084.5	18.27	20.00
en-lv	Unique	4/183	63.5/9.1	0.80/0.26	0.75/0.30
	Common	22560	1235.2/1175.8	30.84/30.77	25.34/25.10
	Max-size	22560	1235.3	30.84	25.34
	Max-acc	21975	1080.1	28.22	22.43
et-lv	Unique	3/54	63.5/3.4	0.73/0.06	0.75/0.14
	Common	22681	1293.7/1272.0	22.31/22.29	25.51/25.41
	Max-size	22681	1293.7	22.31	25.51
	Max-acc	22588	1242.3	21.67	24.44
de-en	Unique	4/83	66.1/3.7	0.84/0.11	0.80/0.08
	Common	23331	1272.7/1236.0	29.54/29.44	32.00/31.97
	Max-size	23331	1272.8	29.54	32.00
	Max-acc	22805	1189.9	27.98	30.70

Table 1. Results of processing the corpora: number and sizes of the documents in the common parts, documents present in just one corpus and the resulting max-size and max-accuracy combinations

	UT+JRC2	JRC3 en-et	JRC3 en-lv	JRC3 et-lv	JRC3 de-en
	UT/JRC	Hun/Van	Hun/Van	Hun/Van	Hun/Van
\emptyset	7.1%/9.8%	12.2%/8.4%	11.7%/8.1%	3.9%/2.4%	6.1%/3.9%
0-1	0.0%/8.2%	0.0%/0.0%	0.0%/0.0%	0.0%/0.0%	0.0%/0.0%
1-0	32.5%/0.0%	0.7%/0.0%	0.7%/0.0%	0.0%/0.0%	0.3%/0.0%
1-1	59.3%/81.0%	86.8%/91.4%	87.3%/91.7%	95.8%/97.4%	93.0%/95.8%
N-M	1.0%/0.9%	0.1%/0.1%	0.1%/0.1%	0.2%/0.1%	0.4%/0.2%

Table 2. Frequency of the match types between sentence pairs of the corpora pairs; given as proportion of sentences per match type and corpus.

4.2. Influence on Machine Translation

Whenever it is known that two corpora overlap, concatenating them is an erroneous solution. As a result of straightforward concatenation the sentence pairs present in both parts of the overlap will be overrepresented since their relative frequency will increase in comparison to the sentence pairs outside the overlap or the ones that are present in only one corpus. The correct baseline method of combining overlapping corpora is taking the non-overlapping parts of both corpora and the overlapping part from just one of them. In our case instead of giving preference to either part of UT+JRC2 or JRC3 pairs we used both versions of the baseline, comparing them to the max-size and max-accuracy combinations of CorporAL.

Development and test sets were separated from the rest of the material, prior to processing the common parts. The size of both the dev and test sets was 2500 sentence pairs for all translation directions.

We evaluated the influence of the different corpora versions on two statistical translation systems: the first one is a phrase-based system, implemented in the Moses toolkit (Koehn et al., 2007) and the second one – hierarchical phrase-based, implemented in the Joshua toolkit (Li et al., 2009). Word alignment and language modeling for both systems were done with GIZA++ (Och and Ney, 2003) and SRILM (Stolcke, 2002). We used the BLEU (Papineni et al., 2001) and NIST (NIST, 2002) scores to compare the translation hypotheses.

The resulting scores of all the translation systems are presented in Table 3. In case of the UT+JRC2 pairs a clear pattern is visible: although in some cases the JRC-based results are better than the UT-based results, in general the max-accuracy, UT-based and JRC-based results are very similar and the max-size results noticeably exceed all three. The JRC3 pairs on the other hand do not exhibit any clear pattern. The scales of the differences suggest that there is no significant difference between all four systems in most cases.

Both of these opposite conclusions for UT+JRC2 and JRC3 experiments can be explained by the UT and JRC2 corpora being much more heterogeneous than all the JRC3 pairs, as showed by the results of processing them, as well as by the UT+JRC2 max-size combinations being considerably bigger than the other parts and the JRC3 combinations being of the same size.

At the same time the max-accuracy results are roughly the same as the baselines in the UT+JRC2 case. Similarly, although (Kaalep and Veski, 2007) showed Vanilla alignments to be of worse quality than HunAlign ones, there is no significant difference between the baselines of all the JRC3 pairs. This can be attributed to the frequency-based re-estimation of parameters in statistical machine translation, which results in automatic discarding of noise in the data (such as errors in sentence or word alignments) and thus also in lower sensitivity to alignment quality.

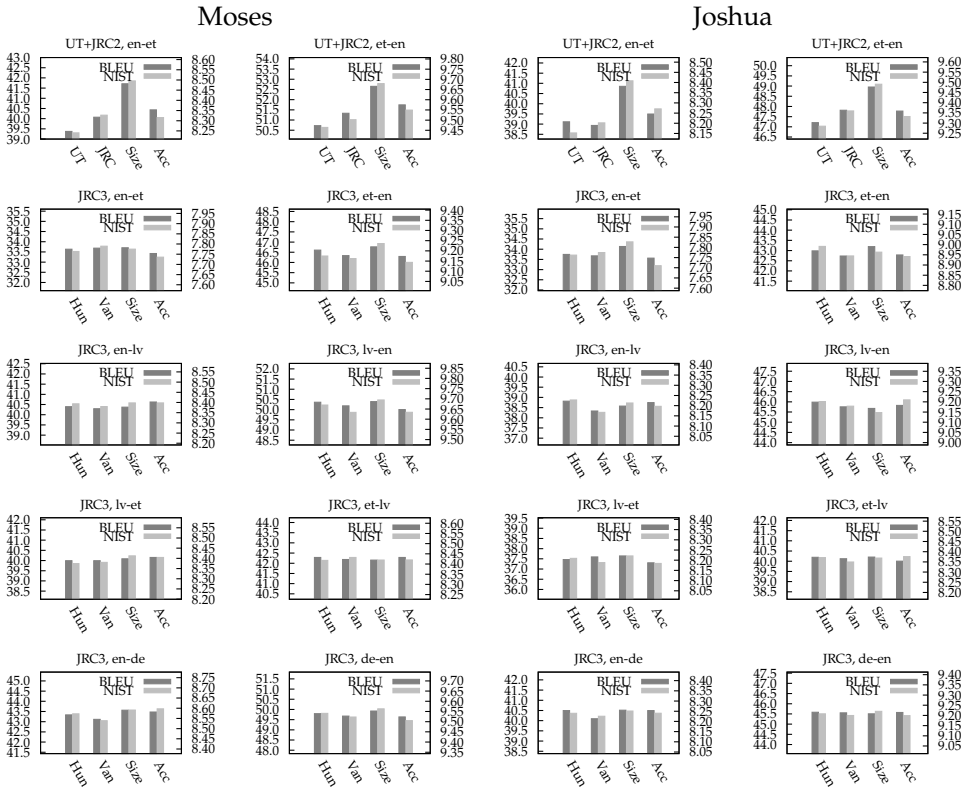


Table 3. Results of the machine translation experiments. The BLEU scale is on the left, and the NIST scale - on the right.

5. Conclusions

In this paper we have introduced a method for handling parallel corpora that are based on the same source material – i.e. overlapping parallel corpora. The method can detect matching and mismatching sentence pairs and omitted sentences. It can cope with minor differences in the text, such as typing errors and different notations. Also it can detect matches between several sentence pairs.

We described the CorporAI tool, which supports flexible combination of overlapping corpora and analysis of their similarities and differences.

The method was tested on two pairs of overlapping parallel corpora: the JRC-Acquis (version 2.2) with the corpus of the University of Tartu and the Vanilla and HunAlign-based versions of the JRC-Acquis (version 3.0); in the second case we in-

cluded four language pairs. Processing the first pair resulted a bigger joint corpus while in case of the other four language pairs the size practically did not increase. Machine translation results showed dependence on the size and heterogeneity of the initial corpora and low sensitivity to alignment quality.

Bibliography

- Bojar, Ondřej and Zdeněk Žabokrtský. CzEng0.9: Large Parallel Treebank with Rich Annotation. *Prague Bulletin of Mathematical Linguistics*, 92, 2009.
- Kaalep, Heiki-Jaan and Kaarel Veskis. Comparing parallel corpora and evaluating their quality. In *Proceedings of MT Summit XI*, pages 275–279, Copenhagen, Denmark, 2007.
- Koehn, Philipp. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit X*, pages 79–86, Phuket, Thailand, 2005.
- Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL'07*, pages 177–180, Prague, Czech Republic, 2007.
- Li, Zhifei, Chris Callison-Burch, Chris Dyer, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece, 2009.
- NIST. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. Technical report, NIST, 2002.
- Och, Franz J. and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.
- Papinen, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL'01*, pages 311–318, Philadelphia, PA, USA, 2001.
- Steinberger, Ralf, Bruno Poulighen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, Dan Tufiş, and Dániel Varga. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of LREC'06*, pages 2142–2147, Genoa, Italy, 2006.
- Stolcke, Andreas. SRILM – an extensible language modeling toolkit. In *Proceedings of ICSLP'02*, volume 2, pages 901–904, Denver, Colorado, USA, 2002.
- Varga, Daniel, László Németh, Péter Halácsy, András Kornai, Viktor Trón, and Viktor Nagy. Parallel corpora for medium density languages. In *Proceedings of RANLP'05*, pages 590–596, Borovets, Bulgaria, 2005.

Address for correspondence:

Mark Fishel
fishel@ut.ee
University of Tartu, J. Liivi 2, 50409 Tartu, Estonia



The Prague Bulletin of Mathematical Linguistics
NUMBER 94 SEPTEMBER 2010 77-86

Asiya: An Open Toolkit for Automatic Machine Translation (Meta-)Evaluation

Jesús Giménez, Lluís Màrquez

TALP Research Center, Universitat Politècnica de Catalunya

Abstract

This article describes the ASIYA Toolkit for Automatic Machine Translation Evaluation and Meta-evaluation, an open framework offering system and metric developers a text interface to a rich repository of metrics and meta-metrics.

1. Introduction

Evaluation methods are a key ingredient in the development cycle of Machine Translation (MT) systems (see Figure 1). They are used to identify the system weak points (error analysis), to adjust the internal system parameters (system refinement) and to measure the system performance, as compared to other systems or to different versions of the same system (evaluation). Evaluation methods are not a static component. On the contrary, far from being perfect, they evolve in the same manner that MT systems do. Their development cycle is similar: their weak points are analyzed, they are refined, and they are compared to other metrics or to different versions of the same metric so as to measure their effectiveness. For that purpose they rely on additional meta-evaluation methods.

In this article, we present ASIYA, an open toolkit aimed at covering the evaluation needs of system and metric developers along the development cycle¹. In short, ASIYA

¹Asiya was the Israelite wife of the Pharaoh who adopted Moses after her maids found him floating in the Nile river (see <http://en.wikipedia.org/wiki/Asiya>). The ASIYA toolkit is the natural evolution/extension of its predecessor, the IQ_{MT} Framework (Giménez and Amigó, 2006). ASIYA is publicly available at <http://www.lsi.upc.edu/~nlp/Asiya>.

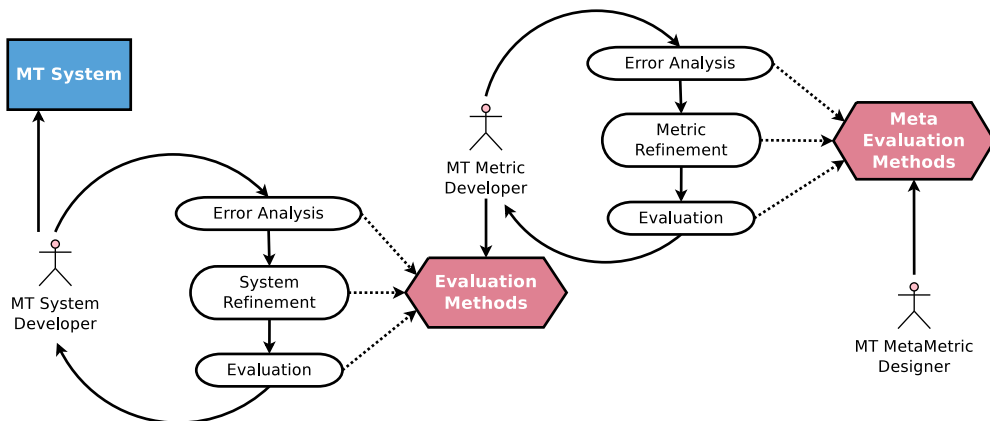


Figure 1. System development cycle in Machine Translation

is a common interface to a compiled collection of evaluation and meta-evaluation methods (i.e., hexagonal boxes in Figure 1). The metric repository incorporates the latest versions of most popular metrics, operating at different linguistic dimensions (lexical, syntactic, and semantic) and based on different similarity assumptions (precision, recall, overlap, edit rate, etc.). ASiYA also incorporates schemes for metric combination, i.e., for integrating the scores conferred by different metrics into a single measure of quality. The meta-metric repository includes both measures based on human acceptability (e.g., correlation with human assessments), and human likeness, such as ORANGE (Lin and Och, 2004a) and KING (Amigó et al., 2005).

2. Tool Description

ASiYA operates over predefined test suites, i.e., over fixed sets of translation test cases (King and Falkedal, 1990). A test case consists of a source segment, a set of candidate translations and a set of manually-produced reference translations. The utility of a test suite is intimately related to its representativity, which depends on a number of variables (e.g., language pair, translation domain, number and type of references, system typology, etc.). These variables determine the space in which MT systems and evaluation metrics will be allowed to express their capabilities, and, therefore, condition the results of any evaluation and meta-evaluation process conducted upon them.

ASiYA requires the user to provide the test suite definition through a configuration file. Different test suites must be placed in different folders with their correspond-

ing configuration files. Preferred input format is the NIST XML as specified in the Metrics MaTr Evaluation Plan (Callison-Burch et al., 2010)². For instance, the sample configuration file in Table 1 defines source material (source.xml), candidate translations (candidates.xml), and reference translations (references.xml). If the source file is not provided, the first reference will be used as source for those metrics which take it into consideration. Candidate and reference files are required.

```
# lines starting with '#' are ignored

src=source.xml
sys=candidates.xml
ref=references.xml

some_metrics=-TERp METEOR-pa CP-STM-6 DP-Or(*) SR-Or(*) DR-Or(*) DR-STM-6
some_systems=system01 system05 system07
some_refs=reference02 reference04
```

Table 1. Sample configuration file ('sample.config')

ASIYA may be then called by typing the following on the command line: `Asiya.pl sample.config`. When called without any additional option further than the name of the configuration file, ASIYA will read the file and check its validity (i.e., whether the defined files exist and are well-formed). No output will be delivered to the user other than status and error messages. However, several files will be generated. Input XML files are processed and texts are extracted and saved as plain '.txt' files in the original data folder. There will be one source file, and as many candidate and reference files as systems and reference sets are specified in the XML file. The correspondence between text files and document and segment identifiers is kept through simple index files ('.idx').

2.1. Evaluation Options

Evaluation reports are generated using the '-eval' option followed by a comma-separated list of evaluation schemes to apply. Three schemes are currently available:

- **Single** metric scores
- **Ulc** normalized arithmetic mean of metric scores
- **Queen** scores as defined by Amigó et al. (2005)

²<http://www.nist.gov/itl/iad/mig/metricsmatr10.cfm>

Several output formats are available through the ‘-o’ option. Default format is ‘-o mmatrix’ (one system, doc or segment per line, each metric in a different column). By default metrics are sorted according to the order as typed by the user. It is also possible to sort them alphabetically using the ‘-sorted name’ option. Other output formats are ‘-o smatrix’ (one metric per line, each system in a different column) and ‘o nist’ which saves metric scores into files complying with the NIST output format as specified in the Metrics MaTr Evaluation Plan.

As an additional option, evaluation scores for the reference translations may be also retrieved through the ‘-include_refs’ option. References will be evaluated against all other references in the test suite.

Besides evaluation reports, ASYA generates, for convenience, several intermediate files:

- **Metric scores:** Results of metric executions are stored in the ‘./scores/’ folder in the working directory, so as to avoid having to re-evaluate already evaluated translations. It is possible, however, to force metric recomputation by setting the ‘-remake’ flag. Moreover, because each metric generates its reports in its own format, we have designed a specific XML representation format which allows us to access metric scores in a unified manner.
- **Linguistic annotations:** Metrics based on syntactic and semantic similarity may perform automatic linguistic processing of the source, candidate and reference material. When necessary, these will be stored in the original data folder so as to avoid having to repeat the parsing of previously parsed texts.

2.2. Meta-Evaluation Options

Meta-evaluation reports are generated using the ‘-metaeval’ option followed by a comma-separated list of metric combination schemes and a comma-separated list of meta-evaluation criteria to apply. Five criteria are currently available:

- **Pearson** correlation coefficients
- **Spearman** correlation coefficients
- **Kendall** correlation coefficients
- **King** scores (Amigó et al., 2005)
- **Orange** scores (Lin and Och, 2004a)

In order to compute correlation coefficients, human assessments must be provided using the ‘-assessments’ option followed by the name of the file containing them. The assessments file must comply with the NIST CSV format (i.e., comma-separated fields, one assessment per line).

By default, correlation coefficients are accompanied by 95% confidence intervals computed using the Fisher’s z-distribution. It is also possible to compute correlation coefficients and confidence intervals applying bootstrap resampling (Koehn, 2004). If the number of samples is reasonably small, as it may be the case when computing correlation with system-level assessments, exhaustive resampling is feasible (‘-ci

xbootstrap’). Otherwise, the number of resamplings may be selected using the ‘-ci bootstrap’ and ‘-n_resamplings’ options (1,000 resamplings by default). Also, the degree of statistical may be adjusted using the ‘-alfa’ option. ASIYA implements also paired metric bootstrap resampling. All metrics are compared pairwise. The proportion of times each metric outperforms the other, in terms of the selected criterion, is retrieved.

Finally, ASIYA provides a mechanism to determine optimal metric sets. These may be found using the ‘-optimize’ option followed by a specific evaluation scheme and meta-evaluation criterion (see Section 2.2).

2.3. General Options

Input Format Candidate and reference translations may be represented in a single file or in separate files. Apart from the NIST XML format, previous NIST SGML and plain text formats are also accepted. Input format is specified using the ‘-i’ option followed by any of the formats available (‘nist’ or ‘raw’).

Language Pair By default, ASIYA assumes the test suite to correspond to an into-English translation task. This behavior may be changed using the ‘-srclang’ (source language) and ‘-trglang’ (target language) options. Metrics based on linguistic analysis, or using dictionaries or paraphrases, require a proper setting of these values. It is also possible to tell ASIYA whether text case matters or not. By default, ASIYA will assume the text to be case-sensitive. This behavior may be changed using the ‘-srcase’ (source case) ‘-trgcase’ (target case) options.

Pre-defined Sets The set of metrics to be used may be specified using the ‘-metric_set’ and/or the ‘-m’ options. The ‘-metric_set’ option must be followed by the name of the set as specified in the config file (see Table 1). The ‘-m’ option must be followed by a comma-separated list of metric names. The effect of these options is cumulative. Analogously, you may tell ASIYA to focus on specific system sets (‘-system_set’ and ‘-s’) and reference sets (‘-reference_set’ and ‘-r’). The full list of metric system and reference names defined in the test suite may be listed using the ‘-metric_names’, ‘-system_names’ and ‘-reference_names’ options, respectively³.

Other Options Another important parameter is the granularity of the results. Setting the granularity allows developers to perform separate analyses of system-level, document-level and segment-level results, both over evaluation and meta-evaluation reports. This parameter may be set using the ‘-g’ option. Default granularity is at the system level. The length and precision of floating point numbers may be adjusted using the ‘-float_length’ (10 by default) and ‘-float_precision’ options (8 by default). Finally, the ‘-tex’ flag produces, when applicable, (meta-)evaluation reports directly in L^AT_EX format.

³The set of available metrics depends on language pair settings.

3. Metric Set

Today, *ASIYA* includes repository of more than 600 metrics. In the following, we provide a brief description. We have grouped metrics according to the linguistic level at which they operate.

- **Lexical Similarity**

BLEU Eight variants for different n -gram lengths, cumulative and non-cumulative, and smoothed or not, have been considered (Papineni et al., 2001).

NIST Ten variants for different n -gram lengths, cumulative and non-cumulative, and smoothed or not, have been considered (Doddington, 2002).

GTM . We included three variants taking different values of the e parameter ($e \in \{1, 2, 3\}$) weighting the importance of the length of matching n -grams (Melamed et al., 2003).

METEOR Four variants, progressively adding ‘exact’, ‘stem’, ‘synonym’ and ‘paraphrase’ modules have been considered (Denkowski and Lavie, 2010).

ROUGE Eight variants, for different n -gram lengths, allowing for skip bigrams or not, weighted or not, have been considered (Lin and Och, 2004b).

TERp Four variants, with and without paraphrasing support, have been included (Snover et al., 2009).

O₁ Lexical overlap (Giménez and Màrquez, 2010).

- **Syntactic Similarity**

Shallow Parsing (SP) Average lexical overlap over parts of speech, and base phrase chunk types, and NIST score over sequences of lemmas, parts of speech, and chunks (Giménez and Màrquez, 2010).

Dependency Parsing (DP) Head-word chain matching (Liu and Gildea, 2005) over word forms, grammatical categories and relations, and average lexical overlap between tree nodes according to their tree level, category or relation (Giménez and Màrquez, 2010).

Constituency Parsing (CP) Average lexical overlap over parts of speech and syntactic constituents (Giménez and Màrquez, 2010), and syntactic tree matching (Liu and Gildea, 2005).

- **Semantic Similarity**

Named Entities (NE) Average lexical overlap between NEs according to their type (Giménez and Màrquez, 2010).

Semantic Roles (SR) Average lexical overlap between SRs according to their type, and average role overlap, i.e., overlap between semantic roles independently from their lexical realization (Giménez and Màrquez, 2010).

Discourse Representations (DR) Average lexical and morphosyntactic overlap between DRs according to their type (Giménez and Màrquez, 2010).

metric	bbn-		dcu-		lium-		
	combo	dcu	combo	google	jhu	systran	rbmt3
BLEU _s	0.31	0.27	0.31	0.31	0.27	0.27	0.20
NIST	7.95	7.36	7.91	8.05	7.31	7.33	6.22
GTM ₂	0.28	0.25	0.29	0.29	0.24	0.26	0.22
ROUGE _W	0.35	0.32	0.34	0.34	0.32	0.32	0.29
-TER _p	-0.47	-0.50	-0.48	-0.46	-0.51	-0.51	-0.58
METEOR _{pa}	0.55	0.53	0.55	0.54	0.52	0.52	0.49
SP-NIST _p	6.85	6.40	6.92	7.07	6.24	6.49	5.88
CP-STM ₆	0.40	0.38	0.39	0.41	0.37	0.38	0.35
DP-HWC _w	0.21	0.18	0.20	0.22	0.18	0.18	0.15
DP-HWC _c	0.34	0.32	0.32	0.35	0.32	0.32	0.30
DP-HWC _r	0.30	0.28	0.28	0.31	0.28	0.28	0.26
DP-O _r (★)	0.25	0.23	0.24	0.26	0.22	0.23	0.19
NE-O _e (★)	0.38	0.35	0.40	0.40	0.29	0.38	0.34
SR-O _r (★)	0.24	0.20	0.23	0.24	0.20	0.20	0.18
DR-O _r (★)	0.32	0.29	0.31	0.33	0.28	0.29	0.24
DR-O _{rp} (★)	0.48	0.45	0.47	0.48	0.44	0.45	0.44
DR-STM ₆	0.45	0.42	0.44	0.46	0.41	0.43	0.39

Table 2. ASIYA-generated evaluation report (system level), WMT09 fr-en

4. A Use Case

In this section, we illustrate some of the ASIYA functionalities over a particular test suite. Specifically, we have used the French-English (fr-en) translation task from the 2009 ACL Workshop on Machine Translation, WMT09, (Callison-Burch et al., 2009). There have been three main reasons for selecting this test bed: (i) it is publicly available, (ii) it is reasonably heterogeneous, since it includes system based on different paradigms (statistical vs. rule-based, hybrid, combined), and (iii) it is neutral, since systems are evaluated out-of-domain, i.e., in a domain other than the training domain.

The test suite consists of 111 documents totaling 2525 segments, one reference translation and automatic translations by 21 different systems. Human assessments at the segment level based on different criteria are available for a subset of segments.

First, we use ASIYA to evaluate a subset of the participant systems based on a selected set of metrics operating at different linguistic levels. We use the ‘-tex’ flag to generate directly the table in L^AT_EX format⁴. The output is Table 2.

⁴The command is the following: `Asiya.pl -v -m BLEUs,NIST,GTM-2,ROUGE-W,-TERp,METEOR-pa,SP-NIST,CP-STM-6,DP-HWC_w-4,DP-HWC_c-4,DP-HWC_r-4,DP-Or(*),NE-Oe(*),SR-Or(*),DR-Or(*),DR-Orp(*),DR-STM-6 -s bbn-combo,dcu,dcu-combo,google,jhu,lium-systran,rbmt3 -eval single -o smatrix -float_precision 2 -g sys -tex Asiya.config'.

metric	ρ	confidence
		interval
BLEU _s	0.90	(0.76, 0.97)
NIST	0.89	(0.66, 0.97)
GTM ₂	0.89	(0.72, 0.97)
ROUGE _w	0.93	(0.80, 0.98)
-TER _p	0.86	(0.66, 0.96)
METEOR _{pa}	0.91	(0.78, 0.98)
SP-NIST _p	0.83	(0.58, 0.94)
CP-STM ₆	0.93	(0.79, 0.99)
DP-HWC _w	0.91	(0.75, 0.98)
DP-HWC _c	0.96	(0.88, 0.99)
DP-HWC _r	0.94	(0.83, 0.99)
DP-O _r (★)	0.93	(0.81, 0.98)
NE-O _e (★)	0.67	(0.29, 0.87)
SR-O _r (★)	0.93	(0.80, 0.98)
DR-O _r (★)	0.93	(0.77, 0.98)
DR-O _{rp} (★)	0.92	(0.76, 0.98)
DR-STM ₆	0.93	(0.80, 0.99)

Table 3. ASIYA-generated meta-evaluation report (system level), WMT09 fr-en

Now, let us use ASIYA to evaluate a selected set of metrics. Since we count on human assessments we can compute correlation coefficients. For this example we have used the ‘rank’ assessments. Each assessor was presented with a set of translation outputs to be ranked from best to worst being 1 assigned to the best output, 2 to the second best and so on. The total number of assessments is 2,668. We take the negative rank as a positive measure of quality. With this kind of assessments, segment-level Pearson correlation coefficients would not be very reliable/informative. We can, however, compute Spearman correlation coefficients at the system level. Confidence intervals are computed via bootstrap resampling at a 95% statistical significance⁵.

5. Ongoing and Future Steps

Current development of the toolkit goes in two main directions. First, we are augmenting the metric repository. We are incorporating new metrics and we are porting linguistic metrics to other languages. We also plan to design and implement a mech-

⁵The command is the following: `Asiya.pl -v -m BLEUs,NIST,GTM-2,ROUGE-W,-TERp,METEOR-pa, SP-pNIST,CP-STM-6,DP-HWC_w-4,DP-HWC_c-4,DP-HWC_r-4,DP-Or(*),NE-Oe(*),SR-Or(*),DR-Or(*), DR-Orp(*),DR-STM-6 -metaeval single spearman -assessments data/rank.csv -ci bootstrap -n_resamplings 1000 -float_precision 2 -g sys -tex Asiya.config'.

anism so users can easily incorporate their own metrics. Moreover, we are currently implementing measures for confidence estimation (i.e., when the reference translation is not available). Also, in the future, we plan to consider more sophisticated metric combination schemes and alternative meta-evaluation criteria.

The second direction is on the construction of a visual interface for ASIYA. We are designing a web application for monitoring the whole development cycle. This application will allow system and metric developers to upload their test suites and perform error analysis, automatic and manual evaluation, and meta-evaluation, using their Internet browser.

Acknowledgements

This work has been partially funded by the Spanish Government (OpenMT-2, TIN-2009-14675-C03) and the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement numbers 247762 (FAUST, FP7-ICT-2009-4-247762) and 247914 (MOLTO, FP7-ICT-2009-4-247914).

Bibliography

- Amigó, Enrique, Julio Gonzalo, Anselmo Penas, and Felisa Verdejo. QARLA: a Framework for the Evaluation of Automatic Summarization. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 280–289, 2005.
- Callison-Burch, Chris, Philipp Koehn, Christof Monz, and Josh Schroeder. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, 2009.
- Callison-Burch, Chris, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, 2010.
- Denkowski, Michael and Alon Lavie. Meteor-next and the meteor paraphrase tables: Improved evaluation support for five target languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 339–342, July 2010.
- Doddington, George. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. In *Proceedings of the 2nd International Conference on Human Language Technology*, pages 138–145, 2002.
- Giménez, Jesús and Enrique Amigó. IQMT: A Framework for Automatic Machine Translation Evaluation. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 685–690, 2006.
- Giménez, Jesús and Lluís Màrquez. Linguistic Features for Automatic MT Evaluation. *To Appear in Machine Translation*, 2010.
- King, Margaret and Kirsten Falkedal. Using Test Suites in Evaluation of MT Systems. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, pages 211–216, 1990.

- Koehn, Philipp. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395, 2004.
- Lin, Chin-Yew and Franz Josef Och. ORANGE: a Method for Evaluating Automatic Evaluation Metrics for Machine Translation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 501–507, 2004a.
- Lin, Chin-Yew and Franz Josef Och. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2004b.
- Liu, Ding and Daniel Gildea. Syntactic Features for Evaluation of Machine Translation. In *Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, pages 25–32, 2005.
- Melamed, I. Dan, Ryan Green, and Joseph P. Turian. Precision and Recall of Machine Translation. In *Proceedings of the Joint Conference on Human Language Technology and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2003.
- Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation, RC22176. Technical report, IBM T.J. Watson Research Center, 2001.
- Snover, Matthew, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 259–268, 2009.

Address for correspondence:

Jesús Giménez
jgimenez@lsi.upc.edu
Universitat Politècnica de Catalunya
C/Jordi Girona, 1-3. Campus Nord. Edifici Omega, despatx S-107
Barcelona, 08028. Spain



The Prague Bulletin of Mathematical Linguistics
NUMBER 94 SEPTEMBER 2010 87-96

An Experimental Management System

Philipp Koehn

University of Edinburgh

Abstract

We describe `Experiment.perl`, an experimental management system, that allows the execution of the entire training and testing pipeline of a machine translation experiment with one configuration files. When carrying out multiple experimental runs with changed settings, `Experiment.perl` automatically detects which steps need to be re-run and which can be re-used.

1. Introduction

Running a machine translation experiment involves many steps: preparing training data, building language and translation models, tuning, testing, scoring and analysis of the results. For most of these steps, a different tool needs to be invoked, so this easily becomes very cumbersome. The Experiment Management System (EMS), or `Experiment.perl`, for lack of a better name, makes it much easier to run experiments. It ships with the Moses machine translation toolkit (Koehn et al., 2007).

A typical example is given in Figure 1. The graph was automatically generated by `Experiment.perl`. All that needed to be done was to specify one single configuration file that points to data files and settings for the experiment. In the graph, each step is a small box. For each step, `Experiment.perl` builds a script file that gets either submitted to a compute cluster or executed on the same machine. Note that some steps are quite involved, for instance tuning: On a cluster, the tuning script runs on the head node a submits jobs to the queue itself.

`Experiment.perl` makes it easy for multiple experimental runs with different settings]. It automatically detects which steps do not have to be executed again. `Experiment.perl` plays the same role as `LoonyBin` (Clark et al., 2010), but there are significant

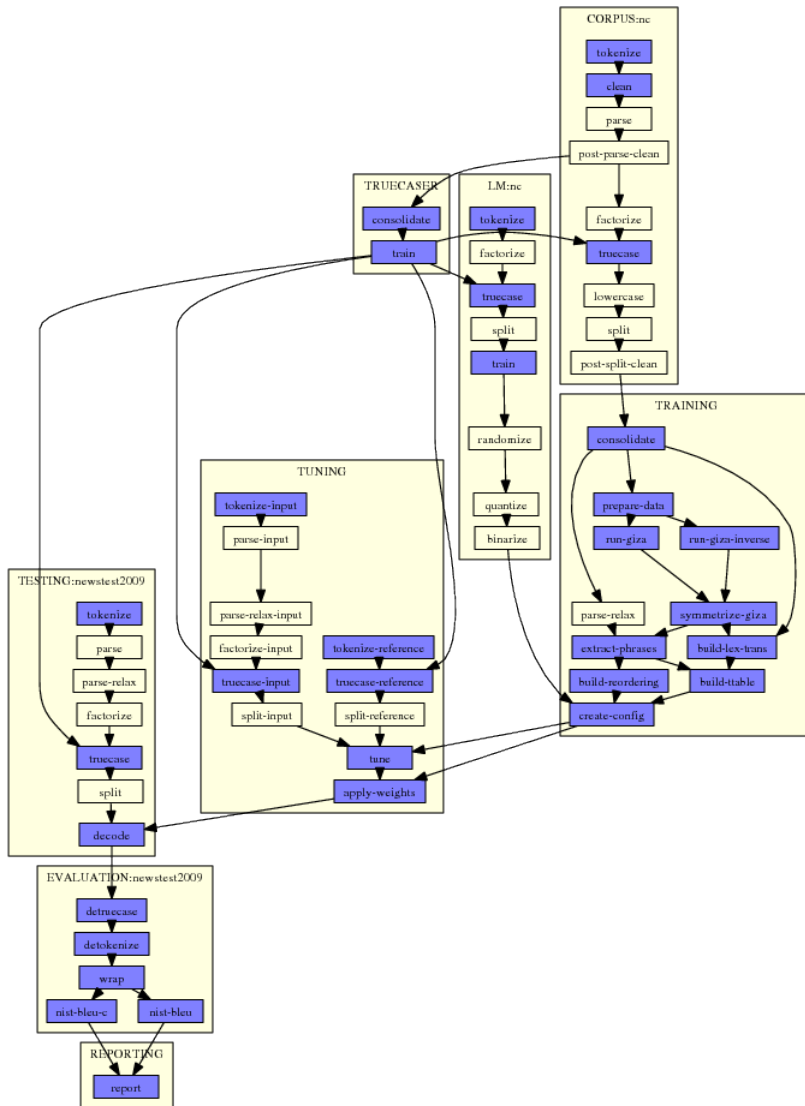


Figure 1. Workflow generated by Experiment.perl: Steps (such as run-giza are grouped into modules (such as TRAINING). Dependencies are indicated by arrows. Skipped steps have a pale background.

differences in its usage. `Experiment.perl` uses a textual configuration file to set up an experiment and a meta configuration file to define all possible workflows, `LoonyBin` offers a graphical user interface that lets the user connect steps.

2. Design

`Experiment.perl` breaks up training, tuning, and evaluating of a statistical machine translation system into a number of steps, which are then scheduled to run in parallel or sequence depending on their inter-dependencies and available resources. The possible steps are defined in the file `experiment.meta`. An experiment is defined by a configuration file.

2.1. Experiment.Meta

The actual steps, their dependencies and other salient information is to be found in the file `experiment.meta`. Think of `experiment.meta` as a "template" file. Steps are grouped into modules, which are:

- CORPUS: preparing a parallel corpus
- INPUT-FACTOR and OUTPUT-FACTOR: commands to create factors
- TRAINING: training a translation model
- LM: training a language model
- INTERPOLATED-LM: interpolate language models
- SPLITTER: training a word splitting model
- RECASING: training a recaser
- TRUCCASING: training a truecaser
- TUNING: running minimum error rate training to set component weights
- TESTING: translating and scoring a test set
- REPORTING: compile all scores in one file

To give an example of a step definition in `experiment.meta`, here the parts of the definition for `LM:get-corporus` and `LM:tokenize`:

```
get-corporus
  in: get-corporus-script
  out: raw-corporus
  [...]

tokenize
  in: raw-corporus
  out: tokenized-corporus
  [...]
```

Each step takes some input (in) and provides some output (out). This also establishes the dependencies between the steps. The step `tokenize` requires the input `raw-corpus`. This is provided by the step `get-corpus`.

The file `experiment.meta` provides a generic template for steps and their interaction. For an actual experiment, a configuration file determines which steps need to be run. This configuration file is specified when invoking `experiment.perl`. It may contain for instance the following:

```
[LM:europarl]

### raw corpus file
#
raw-corpus = $europarl-v3/training/europarl-v3.en
```

Here, the corpus to be used for language modeling is named `europarl` and it is provided in raw text format in the location `$europarl-v3/training/europarl-v3.en` (the variable `$europarl-v3` is defined elsewhere in the config file). The effect of this specification in the config file is that the step `get-corpus` does not need to be run, since its output is given as a file. The workflow starts with the step `tokenize`.

The entire definition of an experiment follows this logic, which is very similar to the principles of a Unix Makefile. The ultimate purpose of an experiment is to generate a result file at the end. If this is not given, then scoring scripts need to be called. Scoring scripts require the output of the decoder on the test sets. The decoder requires a tuned model. Tuning requires a trained model. Trained models require language models and model files, and so on. The workflow, as shown in Figure 1, is generate bottom up, following the input/output dependencies of steps.

2.2. Elements of Step Definitions

Several parameters for step definitions are used in `experiment.meta`:

- `in` and `out`: Established dependencies between steps; input may also be provided by files specified in the configuration.
- `default-name`: Name of the file in which the output of the step will be stored.
- `template`: Template for the command that is placed in the execution script for the step.
- `template-if`: Potential command for the execution script. Only used, if a specified setting exists.
- `error`: `Experiment.perl` detects if a step failed by scanning `STDERR` for key words such as `killed`, `error`, `died`, `not found`, and so on. Additional key words and phrase are provided with this parameter.
- `not-error`: Declares default error key words as not indicating failures.
- `pass-unless`: Only if the given setting is used, this step is executed, otherwise the step is passed (illustrated by a yellow box in the graph).

- `ignore-unless`: If the given setting is used, this step is not executed. This overrides requirements of downstream steps.
- `rerun-on-change`: If similar experiment are runs, the output of steps may be used, if input and settings are the same. This specifies settings whose change disallows a re-use in different run.
- `parallelizable`: When running on a cluster or a multi-core machine, this step may be parallelized (only if `generic-parallelizer` is set in the config file).
- `qsub-script`: If running on a cluster, this step is run on the head node, and not submitted to the queue (because it submits jobs itself).

To complete our example, the full definition of the step `LM:tokenize` is below.

```

tokenize
  in: raw-corpus
  out: tokenized-corpus
  default-name: lm/tok
  pass-unless: output-tokenizer
  template: $output-tokenizer < IN > OUT
  parallelizable: yes

```

The step takes `raw-corpus` and produces `tokenized-corpus`. It is parallelizable with the generic parallelizer. The output is stored in the file according to the definition `corpus/tok`. Note that the actual file name also contains the corpus name, and the run number. In our example the tokenized corpus is stored in a file named `lm/europarl.tok.1`. The step is only executed, if `output-tokenizer` is specified. The template indicate how the command lines in the execution script for the steps are formed.

2.3. Multiple Corpora, One Translation Model

We may use multiple parallel corpora for training a translation model or multiple monolingual corpora for training a language model (or use multiple language models). Each of these have their own instances of the `CORPUS` and `LM` module. There may be also multiple test sets in `TESTING`). However, there is only one translation model and hence only one instance of the `TRAINING` module. The definitions in `experiment.meta` reflects the different nature of these modules. For instance `CORPUS` is flagged as `multiple`, while `TRAINING` is flagged as `single`.

When defining settings for the different modules, the singular module `TRAINING` has only one section, while this one general section and specific `LM` sections for each training corpus. In the specific section, the corpus is named, e.g. `LM:europarl`. When looking up the parameter settings for a step, first the set-specific section (`LM:europarl`) is consulted. If there is no definition, then the module definition (`LM`) and finally the general definition (in section `GENERAL`) is consulted. In other words, local settings override global settings.

2.4. Configuration File

A configuration file for an experimental run consists of a collection of settings, one per line with empty lines and comment lines for better readability, organized in sections for each of the modules.

The start of each section is indicated by the section name in square brackets ([TRAINING] or [CORPUS:europarl]). If the word IGNORE is appended to a section definition, then the entire section is ignored.

The syntax of setting definition is `setting = value` (note: spaces around the equal sign). If the value contains spaces, it must be placed into quotes (`setting = "the value"`), except when a vector of values is implied (only used when defining list of factors: `output-factor = word pos`). Comments are indicated by a hash (#).

Settings can be used as variables to define other settings:

```
working-dir = /home/pkoehn/experiment
wmt10-data = $working-dir/data
```

Variable names may be placed in curly brackets for clearer separation:

```
wmt10-data = ${working-dir}/data
```

Such variable references may also reach other modules:

```
[RECASING]
tokenized = $LM:europarl:tokenized-corpus
```

Finally, reference can be made to settings that are not defined in the configuration file, but are the product of the defined sequence of steps. Say, in the above example, `tokenized-corpus` is not defined in the section `LM:europarl`, but instead `raw-corpus`. Then, the `tokenized corpus` is produced by the normal processing pipeline. Such an intermediate file can be used elsewhere:

```
[RECASING]
tokenized = [LM:europarl:tokenized-corpus]
```

Some error checking is done on the validity of the values in the configuration file before an experimental run is executed. All values that seem to be file paths trigger the existence check for such files. A file with the prefix of the value must exist.

2.5. Step Files

Let us follow our example of the tokenization step in the language model module in more detail. Recall that the `LM:europarl` section has a specification of `raw-corpus` to `$europarl-v3/training/europarl-v3.en`. Since only the raw corpus, but not a tokenized corpus is specified, `Experiment.perl` concludes that it needs to run the tokenization step.

The directory `steps` contains the script that executes each step, its `STDERR` and `STDOUT` output, and meta information:

```
steps/1/LM_europarl_tokenize.1
steps/1/LM_europarl_tokenize.1.DONE
steps/1/LM_europarl_tokenize.1.INFO
steps/1/LM_europarl_tokenize.1.STDERR
steps/1/LM_europarl_tokenize.1.STDERR.digest
steps/1/LM_europarl_tokenize.1.STDOUT
```

The file `steps/1/LM_europarl_tokenize.1` is the shell script that is run to execute the step. The file with the extension `DONE` is created when the step is finished - this communicates to the scheduler that subsequent steps can be executed. The file with the extension `INFO` contains meta information - essential the settings and dependencies of the step. This file is checked to detect if a step can be re-used in subsequent experimental runs.

In case that the step crashed, we expect some indication of a fault in `STDERR` (for instance the words `core dumped` or `killed`). This file is checked to see if the step was executed successfully, so subsequent steps can be scheduled or the step can be re-used in new experiments. Since the `STDERR` file may be very large (some steps create Megabytes of such output), a digested version is created in `STDERR.digest`. If the step was successful, it is empty. Otherwise it contains the error pattern that triggered the failure detection.

2.6. Re-Use of Steps

Let us now take a closer look at re-use. If we run the experiment again but change a settings, say, the order of the language model, then there is no need to re-run the tokenization, but only language model training.

Here is the definition of the language model training step in `experiment.meta`:

```
train
  in: split-corpus
  out: lm
  default-name: lm/lm
  ignore-if: rlm-training
  rerun-on-change: lm-training order settings
  template: $lm-training -order $order $settings -text IN -lm OUT
  error: cannot execute binary file
```

The mention of `order` in the list behind `rerun-on-change` informs `experiment.perl` that this step does need to be re-run, if the order of the language model changes. Since none of the settings in the chain of steps leading up to the training have been changed, those can be re-used.

5689 occurrences in corpus, 590 distinct translations, translation entropy: 3.35224

[#0]

Barack Obama becomes the fourth American president to receive the Nobel Peace Prize
 [0.2521] Barack Obama wird der vierte amerikanische Präsident den Friedensnobelpreis erhalten.
 Barack Obama erhält als vierter US @-@ Präsident den Friedensnobelpreis

Figure 2. Comparing outputs from two experimental runs

If the language model order is changed and the `experiment.perl` is run again in the same working directory, you will see the following files in the directory `lm`:

```
% ls -tr lm/*
lm/europarl.tok.1
lm/europarl.truecased.1
lm/europarl.lm.1
lm/europarl.lm.2
```

Note that a new language model was trained for this second run (`lm/europarl.lm.2`), but no new tokenized and truecased corpus files. These were re-used from run 1.

Steps are re-used from previous runs, unless settings listed under `rerun-on-change` are changed, one of its specified input files (if any) are changed, and if one of its previous steps are re-run. Note that if a filename is not changed, but its time stamp differs, this triggers re-running a step. This ensures that only a minimum number of steps are run to produce the exact same outcome as if all steps are run.

2.7. Web Interface and Analysis

`Experiment.perl` also offers a web interface to the experimental runs for easy access and comparison of experimental results. The web interface gives a listing of experiments and runs for each experiments, with a display of automatic metric scores, and links to configuration files and outputs.

You can include additional analysis for an experimental run in the web interface by specifying the setting `analysis` in its configuration file. This adds reports n-gram precision and recall statistics and color-coded n-gram correctness markup for the output sentences to the web interface. See Figure 2 for an example.

The output is color-highlighted according to n-gram matches with the reference translation. The following colors are used. The darker the color of an output word, the higher n-gram match to the reference translation it is part of.

Additional reports are available when adding the settings `analyze-coverage` and `report-segmentation`. The setting `analyze-coverage` include a coverage analysis: which words and phrases in the input occur in the training data or the translation table? This is reported in color coding and in a yellow report box when moving the

mouse of the word or the phrase. Also, summary statistics for how many words occur how often are given, and a report on unknown or rare words is generated.

The setting `report-segmentation` creates summary statistics about what kind of phrase mappings are used (one word to one word, 1-2, 2-2, etc.), as well as markup of the sentence pair with the phrase segmentation. The phrase segmentation is indicated with black boxes around the words, and the alignment is shown when moving the mouse on the phrases.

3. Usage

3.1. Quick Start

`Experiment.perl` is extremely simple to use:

- Find `experiment.perl` in `scripts/ems`
- Get a sample configuration file from someplace (for instance `scripts/ems/example/config.toy`).
- Set up a working directory for your experiments for this task (`mkdir` does it).
- Edit the following path settings in `config.toy`
 - `working-dir`
 - `data-dir`
 - `moses-script-dir`
 - `moses-src-dir`
 - `srilm-dir`
 - `decoder`
- Run `experiment.perl -config config.toy` from your working directory.
- Marvel at the graphical plan of action.
- Run `experiment.perl -config config.toy -exec`.
- Check the results of your experiment (in `evaluation/report.1`)

3.2. More Examples

The `example` directory contains some additional examples. These require the training and tuning data released for the Shared Translation Task for WMT 2010.

The examples using these corpora are

- a basic phrase based model
- a factored phrase based model
- a hierarchical phrase based model
- a target syntax model

The factored model using all the available corpora is identical to the Edinburgh submission (Koehn et al., 2010) to the WMT 2010 shared task for English-Spanish, Spanish-English, and English-German language pairs. The French language pairs also used the 10^9 corpus, the Czech language pairs did not use the POS language model, and German-English used additional pre-processing steps.

4. Outlook

We have been using Experiment.perl for years and are satisfied with its core functionalities. In future work, we would like support job scheduling on Hadoop clusters and extend the analysis facility.

Acknowledgments

This work was supported in part by the EuroMatrixPlus project funded by the European Commission (7th Framework Programme) and in part under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022.

Bibliography

- Clark, Jonathan H., Jonathan Weese, Byung Gyu Ahn, Andreas Zollmann, Qin Gao, Kenneth Heafield, and Ion Lavie. The machine translation toolpack for loonybin: Automated management of experimental machine translation hyperwork. *The Prague Bulletin of Mathematical Linguistics*, 93:117–126, January 2010.
- Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Christopher J. Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P/P07/P07-2045>.
- Koehn, Philipp, Barry Haddow, Philip Williams, and Hieu Hoang. More linguistic annotation for statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 96–101, Uppsala, Sweden, July 2010. URL <http://www.aclweb.org/anthology/W10-1716>.

Address for correspondence:

Philipp Koehn
pkoehn@inf.ed.ac.uk
School of Informatics, University of Edinburgh
10 Crichton Street
Edinburgh, EH8 9AB, United Kingdom



The Prague Bulletin of Mathematical Linguistics
NUMBER 94 SEPTEMBER 2010 97–106

A Toolkit for Visualizing the Coherence of Tree-based Reordering with Word-Alignments

Gideon Maillette de Buy Wenniger, Maxim Khalilov, Khalil Sima'an

Institute for Logic, Language and Computation, University of Amsterdam

Abstract

Tree-based reordering constitutes an important motivation for the increasing interest in syntax-driven machine translation. It has often been argued that tree-based reordering might provide a more effective approach for bridging the word-order differences between source and target sentences. One major approach (known as Inversion Transduction Grammar) allows permuting the order of the subtrees dominated by the children of any node in the tree. In practice, it has often been observed that the word-alignments usually cohere only to a certain degree with this kind of tree-based reordering, i.e., there are cases of word-alignments that cannot be fully explained with tree-based reordering *when the tree is fixed a priori*.

This paper describes a toolkit for visualizing alignment graphs that consist of a word-alignment together with a source or target tree. More importantly, the toolkit provides a facility for visualizing the coherence of word-alignment with tree-based reordering, highlighting nodes and word-alignments that are incompatible with one another. The tool allows visualizing the tree-based reordered source/target string as well as the reordered tree.

1. Introduction

Word-alignment (the mapping from source language words to target language words) is the starting point for most translation systems in the Statistical Machine Translation (SMT) (e.g. (Och and Ney, 2004; Koehn et al., 2003; Mariño et al., 2006)). Such translation relationships among the words can be *m-to-n* in the most general case, where *m* source words can produce *n* target words.

General tools have been created for the visualization of basic word alignment (Smith and Jahr, 2000; Germann, 2008) as well as for the manual annotation of

sentence pairs. The recent trend in the SMT research community towards including syntactic information into SMT systems makes the simultaneous visualization of alignment and parse trees increasingly more important. This combination of source or target tree with alignment links is also known as an *alignment graph* (Galley et al., 2004). One other toolkit became recently available which focuses on the alignment of parallel treebanks, the *Stockholm Tree Aligner* (STA) (Volk et al., 2007).

Beyond the mere visualization of alignment graphs, there is currently an increasing need for visualizing the coherence of tree-based reordering with word-alignment for the purposes of understanding word-order divergence phenomena between pairs of languages. Our tree and alignment visualization toolkit (TAVT) is aimed exactly at this functionality. TAVT allows visualizing alignment graphs as well as the extent to which word-alignments and the constituency parse tree are compatible with one another. More concretely, TAVT visualizes how well reordering under the ITG constraints (Wu, 1997), while restricting the tree to be the source constituency parse (Yamamoto et al., 2008), succeeds to arrive at monotonic word-alignments, i.e. resolves all crossing word-alignments. Our toolkit allows visualizing the source/target permutation that can be obtained under the assumption of ITG-based tree-reordering as well as the incompatible nodes and word-alignments. This is especially useful for the researchers working on tree-to-string or tree-to-tree translation, as this is exactly the data their systems are built upon.

2. Tree and Alignment Coherence Visualizer

In this section we describe our visualization toolkit TAVT. The first function of our toolkit is the visualization of basic word alignments without trees. However, the main and distinguishing function of TAVT is the simultaneous visualization of trees and alignments (*alignment graphs*). Finally, our toolkit enables the automatic reordering of the source tree to optimally match the target string word order under ITG constraints, and allows visualizing the resulting permutation as well as the nodes and alignments incompatible with one another.

TAVT is implemented using parts of the code extracted from the package *ConstTree-Viewer* – a constituency structures viewer, available on:
<http://staff.science.uva.nl/~fsangati/>.

TAVT itself can be downloaded from:

<http://code.google.com/p/tree-alignment-visualizer/>.

2.1. Basic Alignment Visualization

Visualizing the basic word alignments is done by displaying the two sentences one beneath the other, with the aligned words being connected by colored lines. For ease of reading in case of cluttered and crossing alignment lines, different colors are used

for the alignments of the different words (see Figure 1 for an example of a human made m to n alignment).

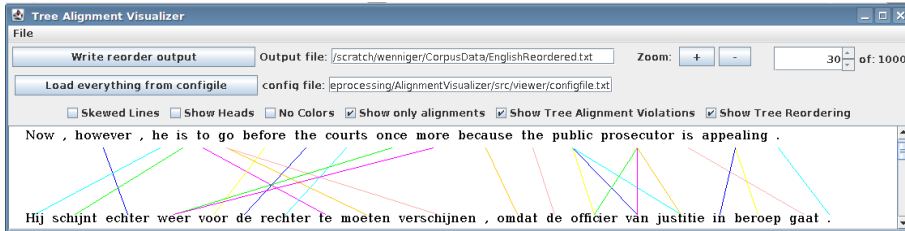


Figure 1: Visualization of the basic word alignments between a source and target sentence.

In our further examples we have restricted ourselves to the *intersection* of the GIZA++ alignments in two directions, which is a common method to improve precision at the price of recall. Furthermore we use IBM model 1 lexical weights to keep only the strongest alignment of a source word, in case it is aligned to multiple target words. These choices are arbitrary and other choices could have been made just as well. The resulting alignments are only 1-to-1 or 1-to-0, and as a result of the intersection they are also incomplete. While different operations such as union allows construction of general m -to- n alignment, the modeling restriction of word-based alignment made by GIZA++ introduces inherent artifacts that direct tree-based alignment could overcome (Pauls et al., 2010).

2.2. Visualizing the Alignment Graph

The visualization of the *alignment graph* is similar to the visualization of the basic word alignments. Rather than displaying the source words, this visualization shows the source tree ending in the leaf nodes containing source words. The source words are again connected by lines to their target-side counterparts.

Basic phrase-based machine translation systems work with phrase pairs that are consistent with the word alignment: the words in a legal phrase pair are contiguous strings consisting of words aligned to each other and not to words outside. As a refinement, syntactic phrases are phrases that are covered by a single subtree in the constituency parse tree (Koehn et al., 2003). From the point of view of syntactic SMT, it is very interesting to see what subtrees of the (source) parse tree are *alignment cohesive* (henceforth *cohesive*), i.e. correspond to the source side of a syntactic phrase pair, and what constituents root a set of children that fail to form a contiguous phrase on the target side. The distinguishing feature of the TAVT is that it gives insight into the *alignment cohesiveness* of subtrees of the parse tree, and *un-cohesiveness* which occurs when alignment spans for subtrees overlap. The overlaps imply that tree-constrained

reordering fails in the sense that ITG-based tree transductions are not sufficient to achieve the ultimate reordering goal: a reordered source tree that has no crossing alignments and whose lexical order matches the order of the target sentence.

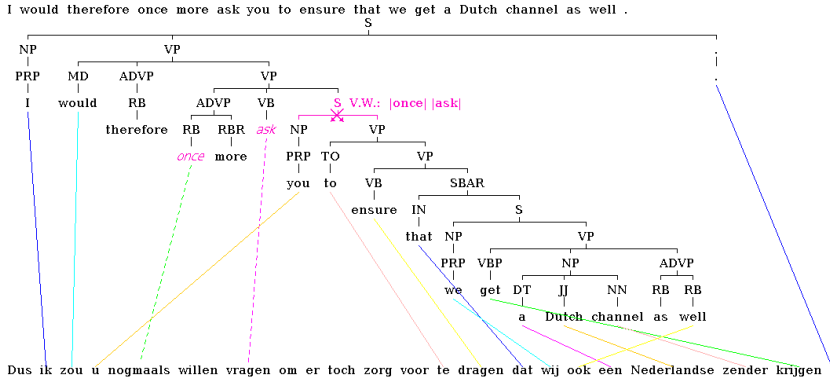


Figure 2: Visualization of the alignment graph.

Tree Alignment Violations and Cohesive Nodes. We define overlapping alignment spans, as follows:

Let $A(n) \rightarrow \{1, \dots, m\}^*$ be the alignment mapping function that maps a source leaf node to a set of zero or more of the m possible target alignment positions.

A certain subtree rooted at node n in the alignment graph *spans* a range $AlignmentSpan(n)$ of target positions defined by the minimum and maximum alignment position over its descending leaf source words:

Definition 2.1 (Alignment Span)

$AlignmentSpan(n) :=$

$$[a_{n_{min}}, a_{n_{max}}] = \left[\min_{x \in LeafNodes(n)} \left(\min_{a_{x'} \in A(x)} a_{x'} \right), \max_{y \in LeafNodes(n)} \left(\max_{a_{y'} \in A(y)} a_{y'} \right) \right]$$

Note that every source leaf node is in principle allowed to map to multiple target word positions, so we have to perform a double minimization/maximization to get the minimum and maximum over this set of sets of alignment positions.

Any source leaf node n' that is not descending from n but aligns to a word in the same range $[a_{n_{min}}, a_{n_{max}}]$ is said to "violate" the *alignment span* of n .

Definition 2.2 (Alignment Violation)

$$\text{violates}(n', n) := \text{terminal}(n') \wedge n' \notin \text{descendants}(n) \wedge \\ (\text{AlignmentSpan}(n) = [a_{n_{\min}}, a_{n_{\max}}]) \wedge (a_{n_{\min}} \leq A(n') \leq a_{n_{\max}})$$

A node n is said to be *cohesive* if it has no alignment violation, in other words that node alone aligns to the contiguous target side of its associated phrase pair.

Alignment violations are indicated by displaying every "violated" subtree in the alignment graph in purple, and showing a list of violating words (V.W.) behind the node label. A symbol consisting of two crossing arrows, just below the root of every subtree that is violated further emphasizes the alignment violations. The violating word itself is accented by an italic purple font, and its alignment is drawn as a striped line to further emphasize its overlap with the alignment range of the other subtree (see Figure 2).

2.3. Optimal tree-constrained source reordering

Displaying how the source tree can be optimally reordered to match the target word order is the goal of the third component of our visualization toolkit. In the tree reordering, we assume that the original parse tree structure must be preserved, the same assumption as made in (Khalilov and Sima'an, 2010). The only allowed operation is the permutation of the child nodes under a parent node (Yamamoto et al., 2008). Given this limited reordering freedom and the alignment spans of different nodes, the tree nodes can be reordered to get a modified tree that better matches the target word order. To do so, every non-terminal node in the tree is visited and every pair of child nodes c_1 and c_2 is compared. c_1 moves before c_2 if and only if the alignment span of c_1 precedes the alignment span of c_2 , denoted as $\text{AlignmentSpan}(c_1) < \text{AlignmentSpan}(c_2)$ and defined as:

Definition 2.3 (Alignment Span Precedence)

$$\text{AlignmentSpan}(c_1) = [a_{1_{\min}}, a_{1_{\max}}] < \text{AlignmentSpan}(c_2) = [a_{2_{\min}}, a_{2_{\max}}] \\ := (a_{1_{\min}} < a_{2_{\min}}) \wedge (a_{1_{\max}} < a_{2_{\min}})$$

Two alignment spans can only be compared if they do not overlap and the one span strictly begins and ends before the other. Note that this is automatically the case if at least one of the two source nodes associated with these alignment spans is *cohesive*. And so the circle closes. *Cohesive* nodes are important since they imply a reordering is possible that will put the source phrases covered by these nodes at the right position, matching the target word position of that phrase (but not necessarily recursively the right order *within* the phrase). *Alignment Violations* are similarly important, since they imply *un-cohesiveness* and thus show where the tree-based reordering scheme fails, be it for alignment errors or simply linguistic complexities.

The example in Figure 3 illustrates the reordering visualization. It shows an alignment graph with multiple crossing alignments due to the fact that the phrase

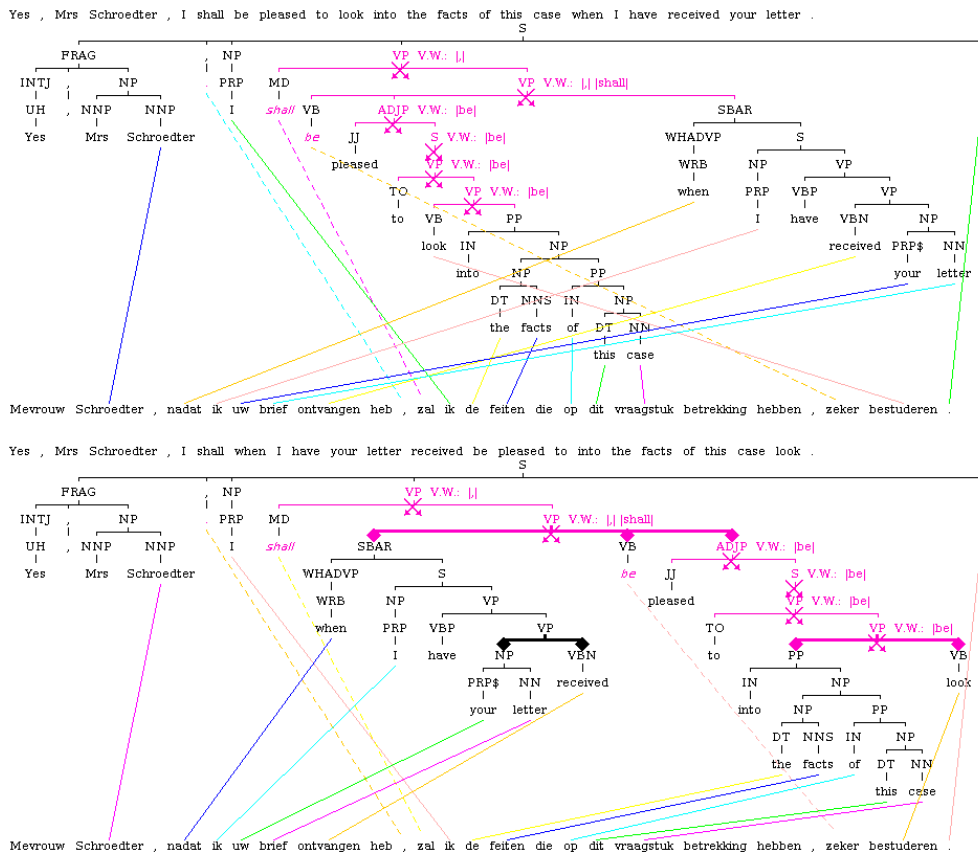


Figure 3: Visualization of the tree-constrained reordering.

“when I have received your letter” → “nadat ik uw brief ontvangen heb” moves from the end of the English source sentence almost completely to the beginning in the Dutch translation. Furthermore there are three words : “,” , “shall” and “be” that cause alignment violations with other subtrees in the alignment graph. In the reordered tree, a thick horizontal line indicates that some of the child nodes under a subtree root are reordered, while those child nodes that really moved to a new position are further emphasized by a diamond just above the node. Unaligned source words like “have” in the example do not directly constrain the word order. However, when the subtrees they belong to move to a new position, they move with them, so their position in the reordered tree is just as well unambiguous. Notice, that in the subtree that roots the phrase “be pleased ... received your letter” not all the children have comparable

alignment spans since the subtree covering “*pleased to look into the facts of this case*” has an alignment span that overlaps with that of its sibling node that covers “*be*”. In contrast, the alignment span of the other sibling covering “*when I have received your letter*” strictly precedes the span of both other siblings, and is thus moved to the front. Therefore, the word order becomes much more like the target word order by performing the reordering procedure, even though overlapping alignment ranges prevent all crossing alignments from being resolvable.

3. Usability

Visualization is generally an effective way for representing and (re-)organizing multi-source information. TAVT is a toolkit that intends to manage word alignment and syntactic information and help users process translation content more efficiently. At the design level, we tried to make our toolkit as intuitive and easy to use as possible. The entire toolkit is written in Java and requires no installation of external libraries. Different aspects of TAVT usability are considered below:

Alignment visualization. At the most basic level TAVT is convenient to browse easily through the different alignments and corresponding parse trees in the data set. This gives a lot of insight in the data in relatively short time and consequently helps in designing effective translation systems.

Alignment and tree visualization. The visualization of trees in addition to the basic alignments helps in different ways. Subtrees are expected to be aligned as word blocks most of the time, if this does not happen it is a clear indication of discrepancy between word alignment and syntactic bilingual segmentation, or alternatively an alignment or parse error.

In this concern, our work has a fair amount of overlap with the STA tool, presented in (Volk et al., 2007). However the focus and functionality of the TAVT toolkit and STA tool differ significantly. While STA visualizes parallel treebanks with alignments, TAVT visualizes alignment graphs which assume only one tree is available. For translation this is often a more realistic assumption, since for many languages no reliable parsers exist. Our goals in developing this application are centered on getting insight into the parallel corpus, dealing with alignment problems, and marking where and how the word reordering takes place and what kind of tree transformations could support it. Therefore we provide functionality for the visualization of tree reordering and order conflicts resulting from subtrees with overlapping alignment spans. Another difference is that STA requires an available parallel treebank, in which case it is very useful, however in practice this implies such a treebank must be (manually) build. In contrast, our toolkit works with automatically extracted

information: GIZA++ alignment and parse trees produced by any constituency parser.

Tree-constrained word reordering visualization. One more important field of TAVT application is the word reordering task at the pre-translation step (Collins et al., 2005; Costa-jussà and Fonollosa, 2006; Xia and McCord, 2004). Here, the word reordering problem is attacked by introducing the pre-processing step into the SMT system, in which the input is rearranged in order to make the source sentence word order resemble that of the target language. Many of these reordering systems exploit syntactic representations of source and target texts and that is where our visualization toolkit can be an asset. The visualization of reordering by means of child node permutations gives a precise idea about how far one can get towards a corpus free of crossing alignments with this transformation to the target word order.

4. Conclusions

TVTA is an open-source visualization toolkit targeted especially to researchers, developers and students working in the field of SMT. Our toolkit goes beyond what other visualization tools offer: by the incorporation of trees in the visualization TVTA gives a lot of meaningful information that other alignment visualization tools do not provide. The extra information is expected to be useful in the research towards better translation systems, and in testing whether certain hypotheses about the translation patterns of a certain language pair actually hold in the data.

5. Future Work

The TVTA visualization framework presents many opportunities for future work. In this section, we describe some of the paths we wish to investigate in the future.

Heuristics in subtree reordering. How should one decide which of two child nodes with overlapping alignment spans should go first in the parse tree? Currently we are preserving their order as it is, since the overlap in alignment spans causes these spans to be *incomparable*. However, this might not always be the best way to deal with it. We have some ideas how we might define order preferences for such incomparable alignment spans. One idea would be to take the alignment weights into account in combination with the target positions of the aligned words in the two spans, and then define a heuristic reordering preference based on these. This is especially important if we want to go beyond mere visualization and learn reordering rules such as it is done in (Khalilov and Sima'an, 2010).

Tree modification. Another idea for an extension would be to incorporate other tree transduction operations, such as insertion and deletion of nodes. Transformation by

a minimum number of such operations would produce a transformed source tree $\tau_{s'}$ that roots the source order in a new order such that the alignments are all non-crossing, while staying as close as possible to the original tree. Visualization of such operations would give a good idea about what combination of operations allows what level of alignment disentanglement. Then with this insight, a more optimal trade-off between level of coverage and computational complexity could be made.

Alignment refinement. Continuing in the same direction, yet another idea is to explore the change of alignments through local re-alignment operations or a whole extra re-alignment phase in the translation process such as described by (Wang et al., 2010). Rather than blindly following the alignments and transforming our trees to match them, we should take into consideration the fact that certain alignments are wrong and the tree can help us to find these. Indeed, recent research in alignment and machine translation builds on the insight that alignments and tree transductions should be optimized simultaneously instead of being factored into two independent steps (Burkett et al., 2010; Pauls et al., 2010). Extensions to the visualization toolbox may indeed be helpful to get insight into the effect of (automatic) re-alignment operations, possibly in combination with simultaneous tree reordering, and how this helps to improve the translation process.

Acknowledgement

This project was supported in part by the Project "Machine Translation When Exact Pattern Match Fails" as part of "free competition for the exact sciences" funded by the Dutch organization for scientific research (NWO) under project number 612066929. The authors wish to thank Federico Sangati who was so kind to make his tree visualization software available on top of which our toolkit has been built.

Bibliography

- Burkett, D., J. Blitzer, , and D. Klein. Joint parsing and alignment with weakly synchronized grammars. In *Proc. of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2010.
- Collins, M., P. Koehn, and I. Kučerová. Clause restructuring for statistical machine translation. In *Proc. of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 531–540, 2005.
- Costa-jussà, M. R. and J. A. R. Fonollosa. Statistical machine reordering. In *Proc. of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 70–76, 2006.
- Galley, M., M. Hopkins, K. Knight, and D. Marcu. What's in a translation rule? In *Proc. of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 273–280, 2004.

- Germann, U. Yawat: yet another word alignment tool. In *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies (ACL-HLT)*, pages 20–23, 2008.
- Khalilov, M. and K. Sima'an. A discriminative syntactic model for source permutation via tree transduction. In *Proc. of The Fourth Workshop on Syntax and Structure in Statistical Translation (SSST-4) at the 23rd International Conference on Computational Linguistics (COLING)*, pages – to appear, 2010.
- Koehn, P., F.J. Och, and D. Marcu. Statistical phrase-based translation. In *Proc. of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL)*, 2003.
- Mariño, J. B., R. E. Banchs, J. M. Crego, A. de Gispert, P. Lambert, J. A. R. Fonollosa, and M. R. Costa-jussà. N-gram based machine translation. *Computational Linguistics*, 32(4):527–549, 2006.
- Och, F. and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449, 2004.
- Pauls, A., D. Klein, D. Chiang, and K. Knight. Unsupervised syntactic alignment with inversion transduction grammars. In *Proc. of 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2010)*, 2010.
- Smith, Noah A. and Michael E. Jahr. Cairo: An alignment visualization tool. In *Proc. of the 2nd Conference on Language Resources and Evaluation (LREC)*, page 549551, 2000.
- Volk, M., J. Lundborg, and M. Mettler. Alignment tools for parallel treebanks. In *In Proc. of The Linguistic Annotation Workshop at the Association for Computational Linguistics (LAW-ACL)*, 2007.
- Wang, W., J. May, K. Knight, and D. Marcu. Re-structuring, re-labeling and re-aligning for syntax-based machine translation. *Computational Linguistics*, 36:247–277, 2010.
- Wu, D. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–404, 1997.
- Xia, F. and M. McCord. Improving a statistical MT system with automatically learned rewrite patterns. In *Proc. of the 20th international conference on Computational Linguistics (COLING)*, pages 508–514, 2004.
- Yamamoto, H., H. Okuma, and E. Sumita. Imposing constraints from the source tree on itg constraints for smt. In *Proc. of the Second Workshop on Syntax and Structure in Statistical Translation (SSST '08)*, page 19, 2008.

Address for correspondence:

Gideon Maillette de Buy Wenniger
gmaillet@science.uva.nl
Institute for Logic, Language and Computation
University of Amsterdam
Science Park 904
1098 XH Amsterdam, Netherlands



The Prague Bulletin of Mathematical Linguistics
NUMBER 94 SEPTEMBER 2010

INSTRUCTIONS FOR AUTHORS

Manuscripts are welcome provided that they have not yet been published elsewhere and that they bring some interesting and new insights contributing to the broad field of computational linguistics in any of its aspects, or of linguistic theory. The submitted articles may be:

- long articles with completed, wide-impact research results both theoretical and practical, and/or new formalisms for linguistic analysis and their implementation and application on linguistic data sets, or
- short or long articles that are abstracts or extracts of Master's and PhD thesis, with the most interesting and/or promising results described. Also
- short or long articles looking forward that base their views on proper and deep analysis of the current situation in various subjects within the field are invited, as well as
- short articles about current advanced research of both theoretical and applied nature, with very specific (and perhaps narrow, but well-defined) target goal in all areas of language and speech processing, to give the opportunity to junior researchers to publish as soon as possible;
- short articles that contain contraversing, polemic or otherwise unusual views, supported by some experimental evidence but not necessarily evaluated in the usual sense are also welcome.

The recommended length of long article is 12–30 pages and of short paper is 6-15 pages.

The copyright of papers accepted for publication remains with the author. The editors reserve the right to make editorial revisions but these revisions and changes have to be approved by the author(s). Book reviews and short book notices are also appreciated.

The manuscripts are reviewed by 2 independent reviewers, at least one of them being a member of the international Editorial Board.

Authors receive two copies of the relevant issue of the PBML together with the original pdf files.

The guidelines for the technical shape of the contributions are found on the web site <http://ufal.mff.cuni.cz/pbml.html>. If there are any technical problems, please contact the editorial staff at pbml@ufal.mff.cuni.cz.

