PBML

The Prague Bulletin of Mathematical Linguistics

NUMBER 99 APRIL 2013

EDITORIAL BOARD

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 99 APRIL 2013

CONTENTS

Articles

High-Precision Sentence Alignment by Bootstrapping from Wood Standard Annotations 5
Éva Mújdricza-Maydt, Huiqin Körkel-Qu, Stefan Riezler, Sebastian Padó

Source-Side Discontinuous Phrases for Machine Translation: A Comparative Study on Phrase Extraction and Search 17
Matthias Huck, Erik Scharwächter, Hermann Ney

The Design of Eman, an Experiment Manager 39
Ondřej Bojar, Aleš Tamchyna

Accounting for Contiguous Multiword Expressions in Shallow Parsing 59
Matthieu Constant, Olivier Blanc, Patrick Watrin

Significance of the Correlation in Turn-taking Dialogues 77
Lourembam Surjit Singh, R.K. Brojen Singh

Utilisation des ressources externes pour la reformulation des requêtes dans un système de recherche d'information 87
Mohammed El Amine Abderrahim

Reviews

Review of the book "Morfologie českého slovesa a tvoření deverbativ jako problém strojové analýzy češtiny" 101
Jaroslava Hlaváčová

Notes

Invitations to Conferences 107

Instructions for Authors 109

The Prague Bulletin of Mathematical Linguistics
NUMBER 99 APRIL 2013 5–16

High-Precision Sentence Alignment by Bootstrapping from Wood Standard Annotations

Éva Mújdricza-Maydt, Huiqin Körkel-Qu, Stefan Riezler, Sebastian Padó

Department of Computational Linguistics, Heidelberg University, Germany

Abstract

We present a semi-supervised, language- and domain-independent approach to high precision sentence alignment. The key idea is to bootstrap a supervised discriminative learner from wood-standard alignments, i.e. alignments that have been automatically generated by state-of-the-art sentence alignment tools. We deploy 3 different unsupervised sentence aligners (Opus, Hunalign, Gargantua) and 2 different datasets (movie subtitles and novels) and show experimentally that bootstrapping consistently improves precision significantly such that, with one exception, we obtain an overall gain in F-score.

1. Introduction

Parallel text is a crucial resource for current approaches to statistical machine translation (Koehn, 2010), statistical models of cross-language information retrieval (Xu et al., 2001; Kraaij et al., 2003; Gao et al., 2006), or other natural language processing tasks that deploy bilingual texts, e.g. monolingual paraphrasing (Bannard and Callison-Burch, 2005).

However, one-to-one sentence parallelism, as found in parliament proceedings, is not the rule but an exception. Most naturally occurring bilingual texts contain roughly corresponding descriptions of the same or overlapping topics. They exhibit parallelism at the level of documents, sentences, or sentence fragments. The challenge to employ such not strictly parallel texts has led to a surge of research on specialized models to extract parallel sentences from sources such as the web (Resnik and Smith, 2003), Wikipedia (Smith et al., 2010), newswire (Munteanu and Marcu, 2005),

Corresponding author: riezler@cl.uni-heidelberg.de

or patents (Utiyama and Isahara, 2007; Lu et al., 2009).

Instead of devising another specialized method for sentence alignment on noisy data, we propose a general two-step model in which a discriminative learner is bootstrapped from data generated by state-of-the-art unsupervised sentence alignment tools. This architecture makes our approach semi-supervised, language- and domain-independent, and applicable to data of various degrees of parallelism. Our bootstrapping methods works as follows: In a first step, we produce large amounts of machine alignments using state-of-the-art sentence aligners. In a second step, we train a discriminative learner on the "wood standard" annotations created in the first step. This combination of arbitrary amounts of machine aligned data and an expressive discriminative learner provides a boost in precision. We evaluate our approach on two datasets: movie subtitles and novels. The efficiency of our approach is ensured by using a moving window of 50 sentence pairs above and below a diagonal of 1-to-1 alignments to break down the huge number of possible alignments (especially in the absence of paragraph breaks, as is the case for movie subtitles). We deploy 3 different unsupervised sentence aligners (Hunalign (Varga et al., 2005), Opus (Tiedemann, 2007), Gargantua (Braune and Fraser, 2010)) for machine labeling. Our experiments show that bootstrapping a discriminative learner significantly improves precision in all cases and, with one exception, also results in an overall gain in F-score.

2. Related work

Most approaches break the sentence alignment problem down into document alignment, e.g. using IR techniques, and a procedure for extracting parallel sentence pairs, e.g., by length-based alignment (Gale and Church, 1993). Typically, sentence pairs are filtered further in a second step on the basis of word alignment scores. These word alignments can be obtained from dictionaries (Utiyama and Isahara, 2007; Lu et al., 2009), external sources (Munteanu and Marcu, 2005; Smith et al., 2010), or from the preliminary sentence pairs obtained in the first step (Braune and Fraser, 2010; Moore, 2002). As an alternative to filtering, word alignments can be integrated as features in a maximum-entropy or CRF model (Munteanu and Marcu, 2005; Smith et al., 2010).

Our approach uses a different kind of two-step approach where a discriminative learner is trained on data that has been machine labeled by state-of-the-art sentence aligners. While our discriminative learner is based on offline computable word-level features, more complex features are hidden in the sentence aligners used in the first step. These may include a word alignment model (e.g., Gargantua (Braune and Fraser, 2010)) or use features that are available only for particular data domains, such as time stamp information (Tiedemann, 2007).

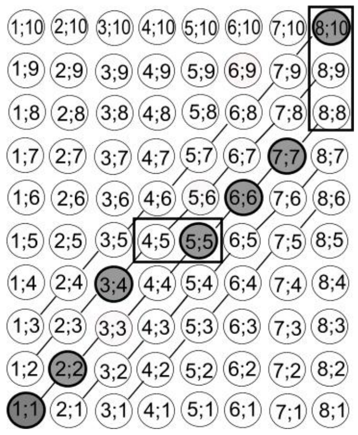

*Figure 1. Example of alignment matrix. Circles represent sentence pairs and are labeled with (source sentence ID; target sentence ID). Bold circles stand for 1:1 gold alignments, bold squares for 1:n and n:1 alignments. Note that target sentence 3 remains unaligned; target sentence 5 participates in a 2:1 alignment (label **S2**); and source sentence 8 takes place in a 1:3 alignment (label **T3**). Filled circles denote model predictions.*

3. Implementation of discriminative sentence alignment

Our toolkit, called *CRFalign*, is implemented in Java and relies heavily on the fast conditional random field sequence classifier *Wapiti* (`wapiti.limsi.fr`) (Lavergne et al., 2010), deploying the expressiveness and flexibility of supervised learning with linear classifiers. Our code is available at `www.cl.uni-heidelberg.de/~mujdricz/software/CRFalign/` and includes the following:

- functions for encoding alignment as sequence labeling problem (see Section 3.1),
- feature functions and interface to *Wapiti* training (see Section 3.2),
- beam search and pruning functions,
- scripts for evaluation and significance testing,
- example corpora and detailed usage instructions.

3.1. Problem encoding

In contrast to simpler sequence labeling problems like part of speech tagging, where we have a sequence of observations to each of which a single label is assigned, the sentence alignment task involves *two* independent sequences, the source and the target corpus. We turn the sentence alignment problem into a set of sequence labeling problems using a *diagonalization* strategy in an alignment matrix, as shown in Figure 1.

This strategy exploits the observation that sentence alignments (like word alignments) tend to follow diagonals in the alignment matrix (Gale and Church, 1993). In other words, the diagonals represent the direction of the most important statistical dependencies: An alignment of sentence pair (n;m) is strong evidence for an alignment at (n+1;m+1). To exploit these dependencies in a linear-chain CRF, we encode each diagonal[1] from an alignment matrix as one sequence labeling problem. For example, one labeling problem would consist of the observation sequence $\langle(1;1),(2;2),\ldots\rangle$ and another one of the observation sequence $\langle(2;1),(3;2),\ldots\rangle$. These sequences capture dependencies between adjacent 1:1 alignments. However, they are unable to directly express the dependencies in 1:n and n:1 alignments, and we express them through labels. We use a set of six labels that express different alignment configurations, defined as follows:

1:1 alignments. If (p;q) is an alignment and no other sentences are aligned with either p or q, the observation (p;q) is labeled with **T**.

2:1 alignments. If (p-1;q) and (p;q) are alignments and no other sentences are aligned with either p or q, then the observation (p;q) is labeled with **S2**.

3:1 alignments. If (p-2;q) (p-1;q) and (p;q) are alignments and no other sentences are aligned with either p or q, then the observation (p;q) is labeled with **S3**.

1:2 alignments. If (p;q-1) and (p;q) are alignments and no other sentences are aligned with either p or q, then the observation (p;q) is labeled with **T2**.

1:3 alignments. If (p;q-2), (p;q-1) and (p;q) are alignments and no other sentences are aligned with either p or q, then the observation (p;q) is labeled with **T3**.

Incomplete alignments and unaligned sentences. All other observations (p;q) are labeled with **F**. This case applies both if p or q are unaligned or if they are part of a larger alignment block.

Therefore, the label sequence in Figure 1 for the observation sequence starting with (1;1) is **T T F F S2 T T F**, and the label sequence for the observation sequence starting with (1;2) is **F F T F F F F F**. This label set is unable to model m:n alignments with $m, n > 1$, or 1:n or n:1 alignments with $n > 3$, but these alignments typically only make up a small fraction of the data, and the cost incurred from introducing more labels exceeds possible benefits.

We apply two optimizations to this process concerning the selection of training and test data to address the predominance of the label **F** in the entirety of alignment matrices. First, we restrict our attention to a subset of all diagonals, exploiting the observation that true alignments usually appear near the first diagonal (the observation sequence starting with (1;1)). Therefore, we only consider diagonals (n;m) where the difference between n and m is smaller than or equal to 50. If the lengths of the texts are very different, we furthermore extend our diagonal set on the axis of the longer

[1] We define a diagonal in a matrix of sentence numbers of two parallel texts as a chain of sentence pairs (*source sentence number*; *target sentence number*) in which the sentence numbers of each next pair are incremented by 1 on both sides.

text. If the source text is x sentences longer than the target text, then we take all diagonals with starting point from $(1;51)$ up to $(51+x;1)$. In the inverse case, we take the diagonals with starting point from $(1;51+x)$ up to $(51;1)$.

The second optimization performs further pruning within this diagonal window: We remove any subsequences of the diagonals that contain more than 15 **F** labels in a row if no other labels follows. This second optimization applies to the training data only, since the test data do not have any access to the labels.

3.2. Features

We use four types of features, all of which are inspired by work on word alignment (Blunsom and Cohn, 2006): (1), length features; (2), position features; (3), similarity features; (4), sequence features. We lift these features to the sentence level. All features, with the exception of the POS agreement feature, are language-independent and can be precomputed from raw text, without the need for linguistic preprocessing.[2] Consequently, our model carries over to other language pairs and to other corpora.

Length ratio. In true alignments, the ratios of source and target sentence lengths can be assumed to be normally distributed (Gale and Church, 1993). The length ratio is an important indicator if a source sentence and a target sentence are a true alignment. We use one feature that captures this intuition. For source and target sentences with m and n words respectively, it is defined as follows:

$$lengthRatio = \min(\frac{m}{n}, \frac{n}{m}).$$

Position ratio. Sentences that are at similar (relative) positions in source and target files are more likely to be aligned than those which are far away from each other. To describe this characteristic of the sentence alignment, we calculate the position ratio. For source sentences and target sentences at positions p and q, and source and target corpora with s and t sentences, the position ratio is defined as

$$positionRatio = |\frac{p}{s} - \frac{q}{t}|.$$

POS similarity. Grammatical agreement between sentences can be evidence that they are aligned. This intuition can be operationalized at the part of speech level. For example, if there are two nouns in the source sentence, there is an increased probability to see two nouns in the target sentence. We obtain POS tags from the TreeTagger (Schmid, 1994) and define a simple cross-lingual mapping. The POS agreement

[2] As we will see in the experimental evaluation, the contribution of the POS similarity feature is marginal, thus vindicating our claim of language independence.

feature is defined as the normalized (by sentence length) overlap between the POS tags of a source and target sentence.

Orthographic similarity. The agreement of named entities and internationally used words in a sentence pair is another strong signal for true alignment. We compute an orthographic similarity-based feature which counts matching words in source and target sentence that contain non-lower-case characters, such as *IBM*, *Oct*, or *2*.

Punctuation similarity. Similarly, shared punctuation between sentences is also evidence for alignment. Here we only compare the last tokens in the current sentence pairs. If they are matching punctuation marks, we assign a weighted similarity value (by inverse corpus frequency) to the sentence pair.

Word edit similarity. Many language pairs share cognates, that is, words with similar spelling. Our *goodEdit* feature captures this observation by counting the relative frequency of similar words:

$$\mathrm{goodEdit}(p, q) = \frac{2 * \mathrm{goodEditCount}}{\mathrm{wordLen}(p) + \mathrm{wordLen}(q)}.$$

where $\mathrm{goodEditCount}$ is defined as the number of word pairs with an edit distance lower than one fifth of their length. This feature can find slightly different spellings of a word in different languages like names such as "Erik" and "Eric", or cognate pairs like "wonder" and "Wunder". This similarity is computed without considering capitalization.

Dice lexical similarity. The Dice coefficient measures the amount of correlated words in two sentences, that is, to what extent the sentences' lexical material co-occurs frequently throughout the corpus. The Dice coefficient for sentences p and q is defined as:

$$\mathrm{Dice}(p, q) = \frac{\sum_i \mathrm{max}_j C(p_i, q_j)}{\mathrm{wordLen}(p) + \mathrm{wordLen}(q)},$$

where the co-occurrence score of two words $C(p_i, q_j)$ is defined as

$$C(p_i, q_j) = \frac{2 * f(p_i, q_j)}{f(p_i) + f(q_j)}.$$

C is highest for word pairs all of whose instances occur in parallel. The Dice coefficient sums the maximum co-occurrence scores for all words in the source sentence that can be obtained by pairing them with the most strongly co-occurring word in the target sentence. The sum is normalized by the lengths of the two sentences. In our computation, we exclude function words from consideration, since they frequently co-occur in unaligned sentences.

Markov sequence feature. Our last feature expresses the first-order Markov dependency between subsequent sentence pairs in a corpus. It is defined to be the label of the preceding pair, i.e. the sequence feature of $(n;m)$ is equal to the label of the preceding pair $(n\text{-}1;m\text{-}1)$. This Markov feature is crucial for our model, since it captures the regularity that subsequent observations are likely to share the same label.

Feature computation. As described in Section 3, our model assigns six labels to observations that represent different sentence alignment configurations. Obviously, decisions among labels like **T** and **S2** (1:1 vs. 2:1 alignment) require access to features not only of the current observation but also of adjacent observations.

For this reason, the feature set for an observation $(p;q)$ consists not only of the feature values for $(p;q)$ itself, but also of the feature values for the four observations that concern the two previous sentences among the source and target sentences and which can have an impact on the label choice: $(p\text{-}1;q)$, $(p\text{-}2;q)$, $(p;q\text{-}1)$ and $(p;q\text{-}2)$. When choosing a label for an observation, the model takes all of these feature "groups" into account. We see that in practice, every feature groups indeed correlates strongly with one label. For example, the model learns that the label **S2** is tightly related with the feature group for $(p\text{-}1;q)$.

4. Experiments

4.1. Data

We evaluate our work on two datasets. The first one is Tiedemann's (2009) OpenSubtitles corpus (`opus.lingfil.uu.se/OpenSubtitles_v2.php`) which consists of parallel subtitles extracted from the on-line subtitle provider `www.opensubtitles.org/`. The parallel data contain over a million translations of movie files in 54 languages. For the language pair German-English, there are 3.4 million sentence pairs comprising 42.8 million tokens. At first glance, the alignment of movie subtitles appears to be a simple problem, since subtitle files contain time stamps that indicate the time of the acoustic appearance of each sentence. (Tiedemann, 2007) has used this information to automatically align sentences for the OpenSubtitles corpus. However, this time information is imperfect, and movie subtitles exhibit other kinds of non-parallelism that make alignment more difficult. Non-parallelism arises from insertions (e.g. of scene descriptions), omissions (e.g., due to compression, language differences, or cultural differences), or other complex mappings (e.g., due to subtitling traditions or special application areas such as subtitles for the hearing impaired that need extra information about sounds).

The second dataset is a parallel corpus of 115 19-th century novels and stories in English and German. The novels are part of the Project Gutenberg (`www.gutenberg.org`) (English) and Projekt Gutenberg-DE (`gutenberg.spiegel.de`) (German) and are available from `www.nlpado.de/~sebastian/data/tv_data.shtml`. As the texts are literary translations, they show a lot of freedom in verbalization. It is also the case that the sentences are on average much longer in novels than in movie subtitles. Sentences in the Gutenberg corpus are on average 25.2 words long. This is more than three times longer than in the OpenSubtitles texts with an average of 7.5 words per sentence. Another source of variability is automatic sentence boundary detection, which leads to typical mistakes in sentence detection which produce $n : m$ alignments ($n, m > 1$). The percentage of such $n : m$ alignments is higher than in the OpenSubtitles corpus.

4.2. Experiment design

For training, *CRFalign* uses *Wapiti* with default meta-parameter settings, except for posterior decoding at labeling time. On the movie subtitle dataset, we use 309 English–German movie pairs from the OpenSubtitles corpus as training set. For evaluation, we manually aligned the German and English subtitles for 6 movies. The annotation guidelines allowed 1:1, 1:n, n:1 and m:n alignments, stating that sentences, or sentence sequences, respectively, should only be aligned if the passages expressed exactly or almost exactly the same content. This guideline was mostly uncontroversial. The train and test data for the Gutenberg dataset consist of 112 novels for training and 3 for testing. The test data set was manually annotated following the OpenSubtitles annotation guidelines.

To evaluate the predictions of our *CRFalign* system for each of the 6 and 3 manually aligned file pairs for OpenSubtitles and Gutenberg data respectively, we compute the standard evaluation measures precision (P), recall (R) and F_1-measure (F). We use the following state-of-the-art sentence alignment tools. On the OpenSubtitles corpus, we deploy the original OpenSubtitles alignments (`opus.lingfil.uu.se/OpenSubtitles_v2.php`) (Tiedemann, 2007) (*OPUS*). This dataset was also sentence-aligned with *Hunalign* (`mokk.bme.hu/resources/hunalign`) (Varga et al., 2005), an unsupervised alignment system that combines length-based alignment with word-alignment filtering. On the Gutenberg corpus, *OPUS* is not available since it relies on time stamp information. Therefore we aligned this data, in addition to *Hunalign*, with *Gargantua* (`sourceforge.net/projects/gargantua/`) (Braune and Fraser, 2010), another state-of-the-art unsupervised sentence alignment tool. Our own system, which we call *CRFalign*, makes its predictions on the basis of these systems for each sentence-pair within the generated diagonals. Recall that for test files we do not prune the diagonals at all, which leads to (a priori) independent predictions for each diagonal. In the case of conflicts (about 5% of predictions), conflicts are resolved by preferring longer alignment chains over shorter ones.

4.3. Experimental results

Table 1 shows a comparative evaluation of state-of-the-art sentence aligners with our discriminative learner trained on machine labeled output of the respective systems.

OpenSubtitles corpus					
OPUS			*Hunalign*		
P	R	F	P	R	F
74.64	73.47	74.05	92.27	91.48	91.87
OPUS + CRFalign			*Hunalign + CRFalign*		
97.59*	85.69*	91.26*	95.51*	87.95*	91.58
Gutenberg corpus					
Gargantua			*Hunalign*		
P	R	F	P	R	F
90.70	89.86	90.28*	74.64	77.76	76.17
Gargantua + CRFalign			*Hunalign + CRFalign*		
91.94	76.64*	83.60*	91.08*	72.22*	80.56*

*Table 1. Precision (P), Recall (R), and F_1-score (F) on OpenSubtitles (top) and Gutenberg (bottom) data for state-of-the-art sentence aligners OPUS, Gargantua, and Hunalign, compared to our CRFalign discriminative sentence aligner trained on the machine labeled output of the respective systems. A statistically significant difference between systems is indicated by * ($p < 0.05$).*

We find that in every single case, precision is significantly improved by bootstrapping the discriminative learner *CRFalign* compared to the original machine-labeled data. Thus, CRFalign consistently acts like a filter that learns to recognize reliable sentence alignment pattern in the output of other aligners. The impact on Recall is more varied: it rises significantly for *OPUS*, drops somewhat for *Hunalign*, and decreases substantially for *Gargantua*. This indicates that the filter is not able to recognize all valid alignment pattern. In the case of Gargantua, F-Score decreases over the initial alignment, however, it increases significantly in most other cases.

We also compared CRFalign against *Hunalign*'s capability to refine its initial alignments in a realignment step (option -realign). *Hunalign*'s realignment results in the following scores (Precision / Recall / F1-measure): 91.14% / 90.73% / 90.93%, and 75.21% / 78.08% / 76.62% on OpenSubtitles and Gutenberg data respectively. Results on OpenSubtitles are slightly lower than the original *Hunalign* alignment scores; the differences on Gutenberg are not statistically significant. These results are lower than the results obtained by realignment via bootstrapping with *CRFalign*.

	OpenSubtitles corpus		Gutenberg corpus	
CRFalign +	*OPUS*	*Hunalign*	*Gargantua*	*Hunalign*
all	92.17	92.22	84.46	79.93
–Markov sequence feature	71.83	60.40	55.07	54.28
–Dice lexical similarity	86.61	86.78	77.98	74.43
–length ratio	88.83	90.11	77.31	76.00
–word edit similarity	91.50	91.85	83.40	80.09
–punctuation similarity	91.81	91.64	83.73	79.48
–position ratio	92.48	92.45	84.04	80.05
–orthographic similarity	91.51	92.49	84.59	80.42
–POS similarity	91.55	92.69	84.75	81.35

Table 2. F-Scores after removing different features the CRFalign feature set.

Furthermore, we investigate the contribution of each feature by ablation, i.e. by leaving out each feature in turn. Table 2 shows the results. Removed features are listed in descending order of their influence on F-score. The analysis shows that the most important individual feature in our feature set is the Markov feature. This feature represents the essential sequential characteristic of our data. Other features contribute to the performance to various, but much smaller, degrees. Most features complement each other so that a cumulative improvement is generally achieved by using all features in the model.

5. Conclusion

This paper has presented an approach sentence alignment that piggybacks on the output of state-of-the-art sentence aligners for bootstrapping a discriminative sentence aligner from machine labeled data. The semi-supervised nature of our approach allows us to aim for high precision alignments while still obtaining improved F-score without the need for manual alignment. Our approach is language- and domain independent and even applicable to datasets with varying degree of parallelism. As shown in the feature ablation experiment, the only language-dependent feature (POS agreement) contributes nearly nothing to overall quality. Our approach addresses the problem of searching a large space of possible sentence alignments by employing a moving window of 50 sentences above and 50 sentences below a diagonal of 1-to-1 alignments. This makes our approach feasible for large datasets even in the absence of paragraph breaks. Finally, the features used in our approach are efficiently computable offline, so that the additional burden of discriminative re-alignment becomes worthwhile if high precision alignments are desired.

Bibliography

Bannard, Colin and Chris Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 597–604, Ann Arbor, MI, 2005.

Blunsom, Phil and Trevor Cohn. Discriminative word alignment with conditional random fields. In *Proceedings of the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL'06)*, pages 65–72, Sydney, Australia, 2006.

Braune, Fabienne and Alexander Fraser. Improved unsupervised sentence alignment for symmetrical and asymmetrical parallel corpora. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*, pages 81–89, Beijing, China, 2010.

Gale, William A. and Kenneth W. Church. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19(1):75–102, 1993.

Gao, Jianfeng, Jian-Yun Nie, and Ming Zhou. Statistical query translation models for cross language information retrieval. *ACM Transactions on Asian Language Information Processing*, 5(4):323–359, 2006.

Koehn, Philipp. *Statistical Machine Translation*. Cambridge University Press, 2010.

Kraaij, Wessel, Jian-Yun Nie, and Michel Simard. Embedding web-based statistical translation models in cross-language information retrieval. *Computational Linguistics*, 29(3):381–419, 2003.

Lavergne, Thomas, Olivier Chappé, and François Yvon. Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10)*, pages 504–513, Uppsala, Sweden, 2010.

Lu, Bin, Banjamin K. Tsou, Jingbo Zhu, Tao Jiang, and Oi Yee Kwong. The construction of a chinese-english patent parallel corpus. In *Proceedings of the MT Summit XII*, pages 17–24, Ottawa, Canada, 2009.

Moore, Robert. Fast and accurate sentence alignment of bilingual corpora. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas (AMTA'02)*, pages 135–144, Tiburon, CA, 2002.

Munteanu, Dragos Stefan and Daniel Marcu. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504, 2005.

Resnik, Philip and Noah A. Smith. The web as a parallel corpus. *Computational Linguistics*, 29 (3):349–380, 2003.

Schmid, Helmut. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK, 1994.

Smith, Jason R., Chris Quirk, and Kristina Toutanova. Extracting parallel sentences from comparable corpora using document level alignment. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT'10)*, pages 403–411, Los Angeles, CA, 2010.

Tiedemann, Jörg. Improved sentence alignment for movie subtitles. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP'07)*, pages 582–588, Borovets, Bulgaria, 2007.

Tiedemann, Jörg. News from OPUS - a collection of multilingual parallel corpora with tools and interfaces. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP'09)*, pages 1–12, Borovets, Bulgaria, 2009.

Utiyama, Masao and Hitoshi Isahara. A Japanese-English patent parallel corpus. In *Proceedings of MT Summit XI*, pages 475–482, Copenhagen, Denmark, 2007.

Varga, Dániel, László Németh, Péter Halácsy, András Kornai, Viktor Trón, and Viktor Nagy. Parallel corpora for medium density languages. In *Proceedings of the Recent Advances in Natural Language Processing 2005 Conference*, pages 590–596, Borovets, Bulgaria, 2005.

Xu, Jinxi, Ralph Weischedel, and Chanh Nguyen. Evaluating a probabilistic model for cross-lingual information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 105–110, New Orleans, LA, 2001.

Address for correspondence:
Stefan Riezler
riezler@cl.uni-heidelberg.de
Department of Computational Linguistics,
Heidelberg University,
Im Neuenheimer Feld 325,
69120 Heidelberg, Germany

The Prague Bulletin of Mathematical Linguistics
NUMBER 99 APRIL 2013 17–38

Source-Side Discontinuous Phrases for Machine Translation: A Comparative Study on Phrase Extraction and Search

Matthias Huck, Erik Scharwächter, Hermann Ney

Human Language Technology and Pattern Recognition Group, RWTH Aachen University

Abstract

Standard phrase-based statistical machine translation systems generate translations based on an inventory of continuous bilingual phrases. In this work, we extend a phrase-based decoder with the ability to make use of phrases that are *discontinuous* in the source part.

Our dynamic programming beam search algorithm supports separate pruning of coverage hypotheses per cardinality and of lexical hypotheses per coverage, as well as coverage constraints that impose restrictions on the possible reorderings. In addition to investigating these aspects, which are related to the decoding procedure, we also concentrate our attention on the question of how to obtain source-side discontinuous phrases from parallel training data. Two approaches (hierarchical and discontinuous extraction) are presented and compared.

On a large-scale Chinese→English translation task, we conduct a thorough empirical evaluation in order to study a number of system configurations with source-side discontinuous phrases, and to compare them to setups which employ continuous phrases only.

1. Introduction

In standard statistical phrase-based machine translation with continuous phrases (Koehn et al., 2003), lexical translation decisions are based on local context only. Source-side discontinuous phrases, in contrast, can explain lexical dependencies between words that appear in a wider context in the source sentence. For example, the French negation is formed with the particle “ne” followed by a verb and a subsequent negative word like “pas” for “not” or “rien” for “nothing”. The lexical decision for correctly translating the negation must be based upon the wider context “ne … pas”

Corresponding author: huck@cs.rwth-aachen.de

or "ne … rien". This can be achieved by using discontinuous phrases like ⟨ne ♢ pas, do not⟩ and ⟨ne ♢ rien, nothing⟩, where the ♢ symbol represents a gap.

Statistical machine translation with discontinuous phrases as we define it in this paper has been introduced by Galley and Manning (2010). We present a generalization of the source cardinality synchronous search algorithm as described by Zens and Ney (2008)—with coverage pruning per cardinality and lexical pruning per coverage—which is able to cope with source-side discontinuities. We give a formulation of a discontinuous phrase extraction algorithm and conduct an in-depth analysis of the differences to hierarchical phrase extraction (Chiang, 2005, 2007). On the NIST Chinese→English translation task, we empirically compare a broad range of setups and configuration parameters.

Note that, while Galley and Manning (2010) allow bilingual phrases with discontinuities both in the source and in the target part, we restrict our study to phrases which are allowed to be discontinuous in the source part only, but are required to be continuous on the target side.

Our implementation has been released as part of version 2.2 of Jane (Vilar et al., 2010, 2012; Wuebker et al., 2012), the RWTH Aachen University open source statistical machine translation toolkit.

2. Source cardinality synchronous search using discontinuous phrases

To translate a source sentence f_1^J of length J with the help of discontinuous phrases, known discontinuous source parts are identified in f_1^J and a target sentence e_1^I of length I is generated out of the corresponding target parts. As in the continuous case, the process yields a segmentation of the sentence.

2.1. Generalized segmentation

Let f_1^J be a source sentence, e_1^I be a target sentence. In the continuous phrase-based model, the segmentation of the sentence pair into K phrases is defined as a sequence s_1^K with $s_k = (i_k; b_k, j_k)$ for $k = 1 \dots K$, where the source phrases $\tilde{f}_k = f_{b_k}^{j_k}$ are continuous. i_k denotes the end of the kth target phrase, b_k denotes the beginning of the kth source phrase and j_k its end. Now, discontinuous source phrases are allowed, so a generalized segmentation $\dot{s}_1^K$ is introduced:

$$k \to \dot{s}_k := (i_k; \tilde{C}_k), \text{for } k = 1 \dots K \qquad (1)$$

Still, i_k denotes the end of the kth target phrase, but now the kth source phrase is given by a phrase coverage set $\tilde{C}_k \subseteq \{1, \dots J\}$. The phrase coverage contains all source positions that are part of the kth source phrase. If $\tilde{C}_k = \{b_k, \dots j_k\}$ for some $b_k, j_k \in \{1, \dots J\}$, the source phrase it represents is continuous. In that case, the set $\tilde{C}_k$ is called continuous.

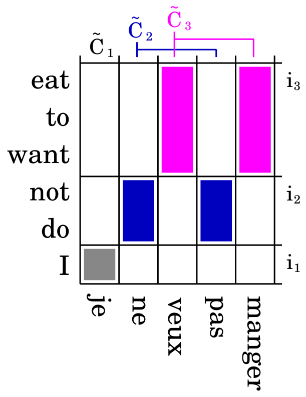

Figure 1. Discontinuous segmentation of a bilingual sentence pair.

Any $\tilde{C}_k$ can be decomposed into a union of N_k maximal continuous subsets

$$\tilde{C}_k = \{b_{k,1}, \dots j_{k,1}\} \cup \dots \cup \{b_{k,N_k}, \dots j_{k,N_k}\} \tag{2}$$

such that

$$\forall\, n:\ 1 \leq n \leq N_k : b_{k,n} \leq j_{k,n} \tag{3}$$

$$\forall\, n:\ 1 \leq n < N_k : j_{k,n} + 1 < b_{k,n+1} \tag{4}$$

Each maximal continuous subset represents a maximal continuous source unit. The combination of these continuous source units gives the complete k^{th} source phrase. The k^{th} target phrase is continuous by definition, as we do not allow for gaps on the target side. This gives us the following notation:

$$\tilde{f}_{k,n} := f_{b_{k,n}}^{j_{k,n}} \qquad \forall\, n:\ 1 \leq n \leq N_k \tag{5}$$

$$\tilde{f}_k := \tilde{f}_{k,1} \lozenge \dots \lozenge \tilde{f}_{k,N_k} \tag{6}$$

$$\tilde{e}_k := e_{i_{k-1}+1} \dots e_{i_k} \qquad (\text{with } i_0 = 0) \tag{7}$$

The segmentation $\dot{s}_k$ describes a partition of the source and target sentence. On the target side, nothing changes from the standard model. It must hold that $i_0 = 0$, $i_K = I$ and $i_{k-1} < i_k$ for $1 \leq k \leq K$. On the source side, constraints on the phrase coverage sets must be imposed:

$$\bigcup_{k=1}^{K} \tilde{C}_k = \{1, \dots J\} \tag{8}$$

$$\tilde{C}_k \cap \tilde{C}_{k'} = \emptyset \qquad \forall\, k \neq k' \tag{9}$$

Figure 1 shows the segmentation of a French–English sentence pair, where two discontinuous phrases are used: ⟨ne ◊ pas, do not⟩ and ⟨veux ◊ manger, want to eat⟩.

2.2. Discontinuous translation model

With the new segmentation, the maximization is carried out over $\dot{s}_1^K$, and the log-linear feature functions (Och and Ney, 2002) have to be adapted accordingly:

$$\hat{e}_1^{\hat{I}} = \operatorname*{argmax}_{I, e_1^I} \left\{ \max_{K, \dot{s}_1^K} \sum_{m=1}^{M} \lambda_m h_m(e_1^I, \dot{s}_1^K; f_1^J) \right\} \tag{10}$$

All standard features functions for phrase-based machine translation can be utilized in the discontinuous phrase-based model with some minor notational changes. A formal redefinition of these features is omitted here. Instead, two new features are introduced which are unique for discontinuous translation.

The first one is the *gappy flag*, which counts the number of discontinuous phrases used in the segmentation:

$$h_{isGappy}(e_1^I, \dot{s}_1^K; f_1^J) = \sum_{k=1}^{K} [N_k > 1] \tag{11}$$

It can be used to reward or penalize the application of discontinuous phrases, depending on its scaling factor. As before, N_k denotes the number of maximal continuous subsets of $\tilde{C}_k$, and $[\mathcal{C}]$ evaluates to 1, if condition $\mathcal{C}$ is true, and 0 if it is false.

The second one is the *gap size* feature. It counts the number of words in between consecutive maximal continuous source units of discontinuous phrases:

$$h_{GS}(e_1^I, \dot{s}_1^K; f_1^J) = \sum_{k=1}^{K} \sum_{n=1}^{N_k - 1} (b_{k,n+1} - j_{k,n} - 1) \tag{12}$$

The gap size feature differs from the standard translation model features, as it cannot be precomputed and stored in the phrase table. Instead, it must be calculated during decoding, similar to the distortion model.

2.3. Dynamic programming beam search

When moving from continuous phrases to phrases with discontinuous source parts, the search space is only slightly altered. Still, nodes of the search graph represent triples $(C, \tilde{e}', j')$, where C is a coverage set, $\tilde{e}'$ is the language model history and j' the last translated source position. A translation decision now is a tuple $(\tilde{C}_k; \tilde{e}_k)$. It can be used if $C \cap \tilde{C}_k = \emptyset$; the successor state is given by:

$$(C \cup \tilde{C}_k, \tilde{e}' \oplus \tilde{e}_k, \max \tilde{C}_k) \tag{13}$$

```
INPUT: source sentence $f_1^J$, maximum source phrase length $\tilde{l}_{max}$, sorted translation candidates $E(\cdot)$,
    models $q_{TM}(\cdot)$, $q_{LM}(\cdot)$, $q_{DM}(\cdot)$, rest cost estimate $R(\cdot)$
$Q(\emptyset, \$, 0) = 0$ ; all other $Q(\cdot,\cdot,\cdot)$ entries are initialized to $-\infty$
FOR cardinality $c = 1$ TO $J$ DO
    FOR source phrase length $\tilde{l} = 1$ TO $\tilde{l}_{max}$ DO
        previous cardinality $c' = c - \tilde{l}$
        FOR ALL coverages $C' \subset \{1, \dots J\} : |C'| = c'$ DO
            FOR ALL start positions $j' \in \{1, \dots J\}$ DO
                FOR ALL end positions $j \in \{j' + \tilde{l} - 1, \dots J\}$
                    FOR ALL phrase coverages $\tilde{C}$ with $|\tilde{C}| = \tilde{l}$, $\min \tilde{C} = j'$, $\max \tilde{C} = j$
                        IF $C' \cap \tilde{C} \neq \emptyset$ THEN CONTINUE
                        coverage $C = C' \cup \tilde{C}$
                        FOR ALL states $\tilde{e}', j'' \in Q(C', \cdot, \cdot)$ DO
                            partial score $q = Q(C', \tilde{e}', j'') + q_{DM}(j'', j')$
                            IF $q + R(C, j) + q_{TM}(\tilde{C})$ isTooBadForCoverage $C$ THEN
                                CONTINUE
                            FOR ALL phrase translations $\tilde{e}'' \in E(\tilde{C})$ DO
                                IF $q + R(C, j) + q_{TM}(\tilde{e}'', \tilde{C})$ isTooBadForCoverage $C$ THEN
                                    BREAK
                                score $= q + q_{TM}(\tilde{e}'', \tilde{C}) + q_{LM}(\tilde{e}''|\tilde{e}')$
                                IF score $+R(C, j)$ isTooBadForCoverage $C$ THEN
                                    CONTINUE
                                language model state $\tilde{e} = \tilde{e}' \oplus \tilde{e}''$
                                IF score $> Q(C, \tilde{e}, j)$ THEN
                                    $Q(C, \tilde{e}, j) =$ score
                                    $B(C, \tilde{e}, j) = (C', \tilde{e}', j'')$
                                    $A(C, \tilde{e}, j) = \tilde{e}$
    pruneCardinality c
```

Figure 2. Dynamic programming beam search algorithm with support for source discontinuities.

The search can be carried out using dynamic programming. Let the helper function $Q(C, \tilde{e}, j)$ denote the score of the best path to node $(C, \tilde{e}, j)$ in the search graph. This node can now be reached by translating any source phrase with phrase coverage $\tilde{C} \subseteq C$ in a predecessor node $(C \backslash \tilde{C}, \tilde{e}', j'')$. It must only hold that $\max \tilde{C} = j$ and, for the translation candidate $\tilde{e}''$, $\tilde{e}' \oplus \tilde{e}'' = \tilde{e}$. The new dynamic programming recursion equation is straightforward:

$$Q(\emptyset, \$, 0) = 0 \tag{14}$$

$$Q(C, \tilde{e}, j) = \max_{\substack{\tilde{C} : \tilde{C} \subseteq C \wedge \max \tilde{C} = j \\ j'', \tilde{e}', \tilde{e}'' : \tilde{e}' \oplus \tilde{e}'' = \tilde{e}}} \Big\{ Q(C \backslash \tilde{C}, \tilde{e}', j'') + q_{TM}(\tilde{e}'', \tilde{C}) + q_{LM}(\tilde{e}''|\tilde{e}') + q_{DM}(j'', \min \tilde{C}) \Big\} \tag{15}$$

The translation model, language model, and distortion model helper functions $q_{TM}(\cdot)$, $q_{LM}(\cdot)$, and $q_{DM}(\cdot)$ are defined as the weighted sums of the involved feature functions. The score of the best translation is given by:

$$\hat{Q} = \max_{\tilde{e},j} \left\{ Q(\{1,\ldots J\}, \tilde{e}, j) + q_{LM}(\$|\tilde{e}) + q_{DM}(j, J+1) \right\} \tag{16}$$

The dynamic programming beam search algorithm can be changed to work with discontinuous source phrases. Figure 2 shows the updated algorithm. The main difference to the standard algorithm (Zens, 2008; Zens and Ney, 2008) is in lines 7 to 10. The standard algorithm loops over the source phrase length $\tilde{l}$ and the start position j', and these two parameters determine the end position $j = j' + \tilde{l} - 1$. The continuous source phrase $f_{j'}^{j}$ starts at position j' and ends at position j.

In the new algorithm, the source phrase length $\tilde{l}$ determines the cardinality of the phrase coverage $\tilde{C}$. A discontinuous phrase starting at position j' with cardinality $|\tilde{C}| = \tilde{l}$ does not necessarily end at position $j = j' + \tilde{l} - 1$, but can end at any position $j \geq j' + \tilde{l} - 1$. Therefore, a second loop over the end position is carried out in line 7. In line 8, all phrase coverages with cardinality $\tilde{l}$ starting at position j' and ending at position j are considered. A phrase matching algorithm is executed before the actual search takes place. The phrase matching algorithm finds for all $\tilde{l}$, j', and j the phrase coverages for which translation candidates are available.

3. Phrase extraction

3.1. Standard phrase extraction

In the standard phrase-based approach, only continuous phrases are extracted (Och et al., 1999; Och, 2002). The set of continuous bilingual phrases $\mathcal{BP}(f_1^J, e_1^I, A)$, given a training instance consisting of a source sentence f_1^J, a target sentence e_1^I, and a word alignment $A \subseteq \{1,\ldots,I\} \times \{1,\ldots,J\}$, is defined as follows:

$$\begin{aligned} \mathcal{BP}(f_1^J, e_1^I, A) = \Big\{ \langle f_{j_1}^{j_2}, e_{i_1}^{i_2} \rangle : \exists (i,j) \in A : i_1 \leq i \leq i_2 \wedge j_1 \leq j \leq j_2 \\ \wedge \forall (i,j) \in A : i_1 \leq i \leq i_2 \leftrightarrow j_1 \leq j \leq j_2 \Big\} \end{aligned} \tag{17}$$

Consistency for continuous phrases is based upon two constraints in this definition: (1.) At least one source and target position within the phrase must be aligned, and (2.) words from inside the source phrase may only be aligned to words from inside the target phrase and vice versa.

3.2. Discontinuous phrase extraction

A general discontinuous phrase with source and target discontinuities is expressed as a pair of coverage sets $(\tilde{C}_{src}, \tilde{C}_{tgt})$ with $\tilde{C}_{src} \subseteq \{1,\ldots,J\}$ and $\tilde{C}_{tgt} \subseteq \{1,\ldots,I\}$.

The phrase is consistent with the alignment A if and only if two conditions hold:

$$\exists(i,j) \in A : i \in \tilde{C}_{tgt} \wedge j \in \tilde{C}_{src} \tag{18}$$

$$\forall(i,j) \in A : i \in \tilde{C}_{tgt} \leftrightarrow j \in \tilde{C}_{src} \tag{19}$$

These are exactly the same constraints as for the continuous case, only with a relaxed view on what is considered a phrase.

We now define the set of discontinuous bilingual phrases $\mathcal{D}(f_1^J, e_1^I, A)$ with discontinuous source parts and continuous target parts. Let N be the total number of gaps allowed in the source part of a phrase, and let $\mathcal{D}_n(f_1^J, e_1^I, A)$ denote the sets of discontinuous phrases with exactly n gaps, $n = 0 \dots N$. The complete set of discontinuous phrases is the union of these smaller sets:

$$\mathcal{D}(f_1^J, e_1^I, A) = \bigcup_{n=0}^{N} \mathcal{D}_n(f_1^J, e_1^I, A) \tag{20}$$

Before moving on to the definition of $\mathcal{D}_n$, the constraint from Equation 18 will be made stronger. For a phrase $\langle f_{j_{1,1}}^{j_{1,2}} \lozenge f_{j_{2,1}}^{j_{2,2}}, e_{i_1}^{i_2} \rangle$, it should hold that the two maximal continuous subsequences of the source part, $f_{j_{1,1}}^{j_{1,2}}$ and $f_{j_{2,1}}^{j_{2,2}}$ are both connected to $e_{i_1}^{i_2}$ with a word alignment, i.e. there exists a pair $(i,j) \in A$ with $i_1 \leq i \leq i_2 \wedge j_{1,1} \leq j \leq j_{1,2}$ and a pair $(i,j) \in A$ with $i_1 \leq i \leq i_2 \wedge j_{2,1} \leq j \leq j_{2,2}$. This stronger constraint is also imposed by Galley and Manning (2010). Furthermore, a gap must span over at least one aligned source position, i.e. there exists a pair $(i,j) \in A$ with $j_{1,2} < j < j_{2,1}$. The constraint from Equation 19 will be kept as it is. With these additional constraints in mind, the sets $\mathcal{D}_n$ can be defined in a general way as follows:

$$\begin{aligned}
\mathcal{D}_n(f_1^J, e_1^I, A) = \Big\{ \langle f_{j_{1,1}}^{j_{1,2}} \lozenge \dots \lozenge f_{j_{n+1,1}}^{j_{n+1,2}}, e_{i_1}^{i_2} \rangle \;\Big|\; & 1 \leq i_1 \leq i_2 \leq I \\
& \wedge\ \forall k : 1 \leq k \leq n+1 : \Big(1 \leq j_{k,1} \leq j_{k,2} \leq J \\
& \quad \wedge\ \exists(i,j) \in A : i_1 \leq i \leq i_2 \wedge j_{k,1} \leq j \leq j_{k,2} \Big) \\
& \wedge\ \forall k : 1 \leq k \leq n : \Big(\exists(i,j) \in A : j_{k,2} < j < j_{k+1,1} \Big) \\
& \wedge\ \forall(i,j) \in A : \Big(i_1 \leq i \leq i_2 \leftrightarrow \big(\exists k : j_{k,1} \leq j \leq j_{k,2} \big) \Big) \Big\}
\end{aligned} \tag{21}$$

3.2.1. Discontinuous extraction algorithm

The algorithm for extracting all discontinuous phrases can be found in Figure 3. The idea of the algorithm is to build up phrases with n gaps from phrases with $n-1$ gaps. It does so by using helper sets $\mathcal{W}_n$ ($n = 0 \dots N$) which contain candidate phrases

```
INPUT: source sentence $f_1^J$, target sentence $e_1^I$, alignment $A$,
       maximum number of source gaps $N$
$\mathcal{W}_0 = \emptyset$
FOR $j_1 = 1$ TO $J$ DO
    IF $j_1$ is unaligned THEN CONTINUE
    $i_1 = \infty; i_2 = -\infty$
    FOR $j_2 = j_1$ TO $J$ DO
        IF $j_2$ is unaligned THEN CONTINUE
        $i_1 = \min\{i_1, \min\{i|(i, j_2) \in A\}\}$
        $i_2 = \max\{i_2, \max\{i|(i, j_2) \in A\}\}$
        $\mathcal{W}_0 := \mathcal{W}_0 \cup \left\{\langle f_{j_1}^{j_2}, e_{i_1}^{i_2}\rangle\right\}$
FOR $n = 1$ TO $N$ DO
    $\mathcal{W}_n = \emptyset$
    FOR $\langle f_{j_{1,1}}^{j_{1,2}} \Diamond \ldots \Diamond f_{j_{n,1}}^{j_{n,2}}, e_{i_1}^{i_2}\rangle$ IN $\mathcal{W}_{n-1}$ DO
        FOR $j_1 = j_{n,2} + 2$ TO $J$ DO
            IF $\nexists(i, j) \in A : j_{n,2} < j < j_1$ THEN CONTINUE
            IF $j_1$ is unaligned THEN CONTINUE
            $i_1' = i_1; i_2' = i_2$
            FOR $j_2 = j_1$ TO $J$ DO
                IF $j_2$ is unaligned THEN CONTINUE
                $i_1' = \min\{i_1', \min\{i|(i, j_2) \in A\}\}$
                $i_2' = \max\{i_2', \max\{i|(i, j_2) \in A\}\}$
                $\mathcal{W}_n = \mathcal{W}_n \cup \{\langle f_{j_{1,1}}^{j_{1,2}} \Diamond \ldots \Diamond f_{j_{n,1}}^{j_{n,2}} \Diamond f_{j_1}^{j_2}, e_{i_1'}^{i_2'}\rangle\}$
FOR $n = 0$ TO $N$ DO
    FOR phrase $r$ IN $\mathcal{W}_n$ DO
        IF check-consistency($r$) THEN $\mathcal{D}_n = \mathcal{D}_n \cup \{r\}$
```

Figure 3. Discontinuous phrase extraction algorithm.

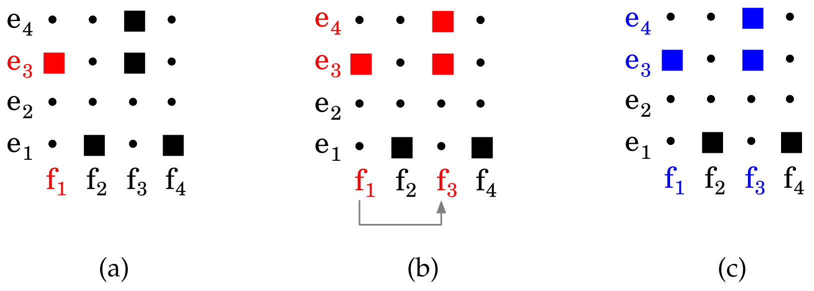

Figure 4. Visualization of discontinuous phrase extraction. Starting from the inconsistent phrase $\langle f_1, e_3\rangle$ from $\mathcal{W}_0$ (red in (a)), the algorithm skips one aligned position and reads another continuous source sequence. The result is first stored in $\mathcal{W}_1$ (red in (b)) and then, since it is a consistent discontinuous phrase, in $\mathcal{D}_1$ (blue in (c)).

that do not necessarily fulfill all consistency constraints. In the end, from these helper sets only the phrases that fulfill all constraints are extracted to $\mathcal{D}_n$.

The first part of the algorithm, lines 2–9, is identical to the standard phrase extraction algorithm. All continuous source sequences from the source sentence are enumerated and the aligned continuous target parts collected, but without checking the consistency constraints. All these phrases are stored in $\mathcal{W}_0$. In lines 10–21, each candidate phrase with $n - 1$ gaps is extended to a phrase with n gaps by skipping at least one aligned position (lines 13 and 15), finding a new continuous source sequence (line 17), and collecting the newly covered target positions (lines 19 and 20). Finally, for all phrase candidates the consistency constraints are checked and the valid phrases are added to the discontinuous phrase set (lines 22–24). Figure 4 visualizes this process.

This algorithm is inspired by the one presented by Lopez (2007) for hierarchical phrase extraction using suffix arrays, which itself is based upon the pattern matching algorithm for variable length gaps by Rahman et al. (2006).

3.3. Hierarchical phrase extraction

Hierarchical phrases (Chiang, 2005, 2007; Vilar, 2011) are essentially special discontinuous phrases where gaps are denoted by the non-terminals. The crucial difference is that non-terminals on the source side and on the target side of hierarchical rules are linked with a one-to-one relation. Typically, a single generic non-terminal symbol X is used as a placeholder for the gaps within the right-hand side of hierarchical translation rules as well as on all left-hand sides of the translation rules that are extracted from the training corpus.

3.3.1. Hierarchical rules for the discontinuous search algorithm

Interpreting hierarchical rules as discontinuous phrases is straight-forward. From a given hierarchical rule

$$X \rightarrow \langle \alpha X^{\sim 1} \beta X^{\sim 2} \gamma, \delta X^{\sim 1} \epsilon X^{\sim 2} \zeta \rangle \tag{22}$$

with $\alpha, \beta, \gamma \in \mathcal{F}^+$ and $\delta, \epsilon, \zeta \in \mathcal{E}^+$, where $\mathcal{F}$ denotes the source vocabulary and $\mathcal{E}$ the target vocabulary, the left-hand side is discarded and all non-terminals are replaced by gap symbols:

$$\langle \alpha \lozenge \beta \lozenge \gamma, \delta \lozenge \epsilon \lozenge \zeta \rangle \tag{23}$$

Hierarchical rules naturally have gaps in their target parts. When using hierarchical extraction to obtain a phrase inventory for application in our discontinuous search procedure, discontinuous target parts must be discarded, and non-terminals at the phrase boundary be removed. This can either be done as a post-processing step, enforcing a renormalization of the phrase probabilities, or it can directly be integrated into the extraction algorithm.

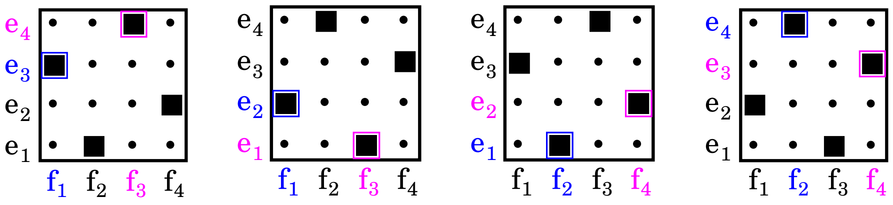

Figure 5. Tricky phrase reorderings. The hierarchical extractor can only generate the phrases $\langle f_2 \lozenge f_4, e_1 e_2 \rangle$ *and* $\langle f_2 \lozenge f_4, e_3 e_4 \rangle$ *from the first two examples by considering the larger phrase* $\langle f_1 f_2 f_3 f_4, e_1 e_2 e_3 e_4 \rangle$ *and first cutting out* f_1*, then* f_3*. The same holds for the last two examples and the phrases* $\langle f_1 \lozenge f_3, e_3 e_4 \rangle$ *and* $\langle f_1 \lozenge f_3, e_1 e_2 \rangle$ *with words* f_2 *and* f_4 *respectively.*

Some properties of hierarchical rules should be considered before using them with the discontinuous search algorithm. For a hierarchical translation system, the two rules $X \rightarrow \langle \alpha, \beta \rangle$ and $X \rightarrow \langle \alpha X^{\sim 1}, \beta X^{\sim 1} \rangle$ are different. For a discontinuous system, both represent the same phrase pair $\langle \alpha, \beta \rangle$. It is tempting to automatically discard all hierarchical rules with non-terminals at the boundaries of the source part to avoid counting the same phrase pair multiple times. In fact, when preparing the hierarchical rule set for discontinuous translation, most of these rules must be discarded. However, there are cases where these phrases are needed because they enable extracting rules which otherwise could not be extracted.

When extracting with at most two non-terminal symbols, there are two alignment configurations that enforce keeping a source part with a non-terminal at its boundary, because there is no other way to extract the resulting phrase pair. Figure 5 shows these two configurations. Wu (1997) characterized them as inside-out reorderings, because they involve a phrase moving from inside the source part to the boundary of the target part and vice versa. Table 1 shows all hierarchical source and target parts that can be used in our source cardinality synchronous discontinuous search algorithm, when extracting with at most two non-terminals (or gaps) per phrase. To avoid confusion, we use the term *hierarchical phrase* only for those phrases, and the term *discontinuous phrase* only for phrases extracted with discontinuous phrase extraction.

The set of hierarchical phrases with up to N non-terminal symbols per rule is a subset of the set of discontinuous phrases with up to N gaps. The difference in the phrase tables from the discontinuous and the hierarchical extraction is analyzed in Section 4.1.

4. Empirical evaluation

We present an empirical evaluation on the NIST Chinese→English translation task.[1] We work with a parallel training corpus of 3.0 M Chinese–English sentences pairs (77.5 M Chinese / 81.0 M English running words). Word alignments are calculated with GIZA++[2] in both directions with four IBM model 1 iterations, five HMM iterations and four IBM model 4 iterations (Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2003). The two directions are combined using the refined heuristic by Och and Ney (2003) to obtain a symmetrized alignment.

4.1. Phrase coverage

We first compare the phrase tables that result from the hierarchical and from the discontinuous approach to phrase extraction.

A single-word extraction heuristic, forced single-word extraction heuristic, and an extraction heuristic for unaligned words as described by Stein et al. (2011) are active in all approaches. Phrases are restricted to a maximum source and target length of 10 words (including gap symbols) with at most two gaps in the source part. For the discontinuous phrase extractor, a maximum gap size of 10 words is used, meaning that the extractor may skip at most 10 words to introduce a new gap.

Let the *span* of a source phrase $\tilde{f}$ in a training sentence f_1^J be the distance between its first word and its last word. For a continuous source phrase $\tilde{f} = f_{j_1}^{j_2}$ the span is given by $j_2 - j_1 + 1$. For a discontinuous phrase with n gaps $\tilde{f} = f_{j_{1,1}}^{j_{1,2}} \lozenge \ldots \lozenge f_{j_{n+1,1}}^{j_{n+1,2}}$ the span is given by $j_{n+1,2} - j_{1,1} + 1$. For hierarchical phrase extraction, a restriction of the maximum source phrase length is also a restriction for the span of the source phrases. A hierarchical phrase is generated by taking a standard phrase and cutting out another standard phrase that is contained in the first one. When the initial standard phrase has a maximum source length of 10, there is no way to generate a hierarchical phrase with a span larger than 10. To extract more discontinuous phrases, this constraint was not imposed in the discontinuous phrase extractor. A discontinuous phrase may span over more than 10 words as long as it consists of at most 10 of them and as long as in each gap at most 10 words are skipped.

After extraction, the phrase tables are filtered towards a larger collection of test sets, i.e. phrases that are not applicable for the translation of any input sentence from one of the test sets are removed from the phrase table. Table 2 shows the number of phrase pairs in the filtered tables extracted with the standard, hierarchical and discontinuous approach. By definition, the hierarchical and discontinuous approach extract all standard phrases from the standard approach.

[1] `http://www.itl.nist.gov/iad/mig/tests/mt/`

[2] `http://code.google.com/p/giza-pp/`

source part	allowed target parts	resulting phrase
α	β	$\langle\alpha, \beta\rangle$
$\alpha X^{\sim 1}\beta$	$X^{\sim 1}\gamma$	$\langle\alpha \Diamond \beta, \gamma\rangle$
	$\gamma X^{\sim 1}$	
$\alpha X^{\sim 1}\beta X^{\sim 2}\gamma$	$X^{\sim 1}\delta X^{\sim 2}$	$\langle\alpha \Diamond \beta \Diamond \gamma, \delta\rangle$
	$X^{\sim 2}\delta X^{\sim 1}$	
	$X^{\sim 1}X^{\sim 2}\delta$	
	$X^{\sim 2}X^{\sim 1}\delta$	
	$\delta X^{\sim 1}X^{\sim 2}$	
	$\delta X^{\sim 2}X^{\sim 1}$	
$X^{\sim 1}\alpha X^{\sim 2}\beta$	$\gamma X^{\sim 1}X^{\sim 2}$	$\langle\alpha \Diamond \beta, \gamma\rangle$
	$X^{\sim 2}X^{\sim 1}\gamma$	
$\alpha X^{\sim 1}\beta X^{\sim 2}$	$X^{\sim 1}X^{\sim 2}\gamma$	$\langle\alpha \Diamond \beta, \gamma\rangle$
	$\gamma X^{\sim 2}X^{\sim 1}$	

Table 1. Hierarchical phrases for the discontinuous model. The overview is complete if no more than two non-terminals are allowed. The last four rows represent the special cases from Figure 5.

approach	total	standard	gappy	% gappy
standard	34.0 M	34.0 M	0	0
hierarchical	48.0 M	34.0 M	14.0 M	29
discontinuous	85.4 M	34.0 M	51.4 M	60
discontinuous*	53.0 M	34.0 M	19.0 M	36

Table 2. Chinese–English phrase table statistics. In the row marked with an asterisk (*), the maximum span of the discontinuous phrases is limited to 10 for hierarchical compatibility.

	absolute	relative
different length constraints	32.4 M	87 %
gaps over non-standard phrases	2.7 M	7 %
obstacle alignment dots	2.3 M	6 %
total additional	37.4 M	100 %

Table 3. Reasons for additional discontinuous phrases.

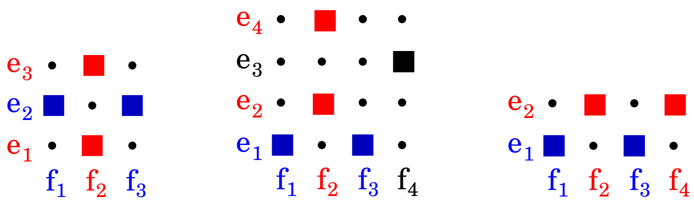

Figure 6. Gaps over non-standard phrases. Out of these three training examples, the hierarchical phrase extractor cannot extract the blue colored phrases, because the red colored initial phrases are non-standard.

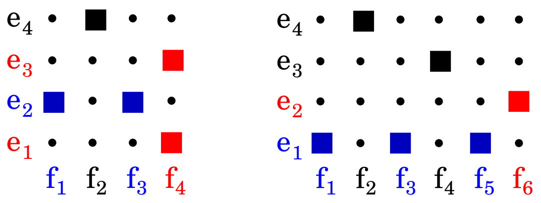

Figure 7. Obstacle alignment dots. In both examples, the gaps span over standard phrases, but the hierarchical extractor cannot extract the blue colored phrases due to the red colored alignment dots.

An analysis of the 37.4 M additional discontinuous phrases has revealed three classes of phrases that are extracted with the discontinuous approach, but not with the hierarchical approach in this setting. These classes can be characterized by *different length constraints*, *gaps over non-standard phrases* and *obstacle alignment dots*. The different length constraints were already mentioned above. A description of the other classes follows in the remainder of this section. Table 3 shows how the additional discontinuous phrases are distributed over the three classes. In fact, 32.4 M of the 37.4 M additional phrases result from the different length constraints. From the remaining 5.0 M additional discontinuous phrases that obey the hierarchical length constraints, 2.7 M have gaps over non-standard phrases and 2.3 M have obstacle alignment dots.

4.1.1. Gaps spanning over non-standard phrases

Some discontinuous phrases cannot be extracted with hierarchical phrase extraction because they include gaps over non-standard phrases. These discontinuous phrases cannot result from cutting out a standard phrase from another standard phrase, regardless of the chosen extraction parameters. Figure 6 depicts three alignments with gaps over non-standard phrases, where the hierarchical extractor cannot extract a phrase with the source part $f_1 \lozenge f_3$. In the first two examples, the gap comprises a phrase with a discontinuous target part. In the third example, there are two discontinuous phrases in a cross-serial configuration (Søgaard and Kuhn, 2009).

4.1.2. Obstacle alignment dots

The third class consists of all discontinuous phrases that do not belong to the first two classes, i.e. they span over at most 10 words and have no gaps over non-standard phrases. Some of these phrases can be extracted with hierarchical phrase extraction if different extraction parameters are chosen. Some are discontinuous phrases with gaps over standard phrases which are not hierarchical for other reasons. All phrases of the third class are extracted from alignments with obstacle alignment dots at some position. These obstacle dots either prevent the phrase from being extracted with the hierarchical approach due to the chosen extraction parameters or prevent them from being hierarchical at all. See Figure 7 for two examples.

The distinction between phrases with gaps over non-standard phrases and phrases with obstacle alignment dots is not very strict. In the first example from Figure 7 the gap consists of part of the discontinuous phrase $\langle f_2 \lozenge f_4, e_1 \lozenge e_3 e_4 \rangle$, while in the second example the second gap could result from cutting out the discontinuous phrase $\langle f_4 \lozenge f_6, e_2 e_3 \rangle$. However, the *smallest* phrases both gaps comprise are standard phrases.

4.2. Translation quality

4.2.1. Experimental setup

In our translation setups, we use the following features (apart from the ones that have been or will be explicitly mentioned): phrase translation probabilities, lexical translation probabilities from IBM model 1 (Brown et al., 1993) and discriminative word lexicon models (Mauser et al., 2009) in the manner of Huck et al. (2011), each for both translation directions, length penalties on word and phrase level, source-to-target and target-to-source phrase length ratios, insertion models (Huck and Ney, 2012), four binary features marking phrases that have been seen more than one, two, three or five times, respectively, a distance-based distortion penalty, and an n-gram language model. The language model is a 4-gram with modified Kneser-Ney smoothing (Kneser and Ney, 1995) which was trained with the SRILM toolkit (Stolcke, 2002) on a large collection of English data including the target side of the parallel corpus. Phrase tables have not been prepruned to contain a maximum number of translation candidates per source side, but the decoder is configured to load at most the 200 best candidates with respect to the weighted phrase-level model scores. We do not impose any hard restriction on the jump width, but the distance-based distortion cost is linear up to a certain limit and quadratic beyond that. The soft jump distance limit is set to 10 in the experiments. Our decoder computes a rest score estimate for the language model, translation model, and distortion model (Zens and Ney, 2008; Moore and Quirk, 2007). Model weights are optimized against Bleu (Papineni et al., 2002) with Minimum Error Rate Training (MERT) (Och, 2003), performance is measured in truecase with Bleu and Ter (Snover et al., 2006). We employ MT06 as development set, MT08 is used as held-out test set.

In the experiments with source-side discontinuous phrases, we depart from the description as given in Section 2 in one aspect: After the application of a discontinuous phrase with coverage vector $\tilde{C} = \{b_1, \ldots, j_1\} \cup \ldots \cup \{b_N, \ldots, j_N\}$ we set the last translated position to j_1, i.e. the maximum of the leftmost maximal continuous subset of the coverage vector, not to $\max \tilde{C}$. The modifications to the notation of Section 2 as well as to the beam search algorithm from Figure 2 are straightforward, and we will omit them here. We extensively compared both variants and found that they are performing at the same level in terms of translation quality. The reason why we decided in favor of this variant is that it may encourage the usage of discontinuous phrases to a larger extent due to a reduced distortion cost (and likewise a reduced distortion rest cost estimate for partial hypotheses).

4.2.2. Experimental results

In a first series of experiments, different reordering constraints are evaluated. Results can be found in Table 4. The last column indicates the *gappy usage* (GU), i.e. the amount of sentence translations in the respective hypothesis for the test set which use at least one discontinuous phrase. Starting with monotonic decoding[3] in the first row, more and more non-monotonicity is allowed in the following rows. First, a soft jump distance limit of 10 is used with phrase-level IBM reordering constraints (Zens et al., 2004) set to 2 (thus allowing only one gap at a time in each coverage set). Then, the number of allowed uncovered blocks according to the reordering constraints is increased step by step. The histogram size for reordering pruning (RH) is set to 64, for lexical pruning (LH) it is also set to 64 for these experiments.

In a second series of experiments, the pruning settings are analyzed. Using the phrase-level IBM reordering constraint with a maximum of 4 runs (thus allowing up to three gaps at a time in each coverage set), different combinations of reordering histogram size and lexical histogram size are tested. We keep the same scaling factors in the log-linear model combinations for all pruning settings. These optimized model weights have been obtained by running MERT on configurations (with and without discontinuous phrases, respectively) with a relatively large search space (RH=64, LH=64).

[3] In pseudo-monotonic decoding with discontinuous phrases, a new phrase must always start at the leftmost uncovered position in the coverage set. Any explicit jumps are not permitted, and we do not compute distance-based distortion costs. Discontinuous phrases may introduce gaps, though.

IBM reordering constraints	gaps	MT06 (dev) BLEU [%]	MT06 (dev) TER [%]	MT08 (test) BLEU [%]	MT08 (test) TER [%]	GU
monotonic	no	30.3	62.1	24.7	66.1	–
	yes	31.6	61.3	25.4	65.7	38.8 %
2	no	32.7	61.0	25.8	66.2	–
	yes	32.9	60.9	25.8	66.1	26.2 %
3	no	32.7	61.1	26.1	65.8	–
	yes	32.8	61.1	25.9	66.0	25.4 %
4	no	32.6	61.1	26.1	65.8	–
	yes	32.8	61.0	26.1	65.7	26.2 %
5	no	32.5	61.0	26.1	65.4	–
	yes	32.7	60.9	25.8	65.7	25.7 %

Table 4. Effect of reordering constraints (with LH=64, RH=64). The experiments have been carried out with a soft jump distance limit of 10. Results are reported in truecase.

Table 5 shows the results. As the gappy usage indicates, a considerable amount of discontinuous phrases is applied for the generation of the single-best hypotheses for all combinations of pruning parameters. However, clear advantages over the setups without discontinuous phrases become evident with very restrictive pruning settings only (RH=4 or LH=4).

We next examine the impact of the gappy flag and the gap size feature (with RH=64 and LH=64). Both features can be used by the decoder to either penalize or reward the use of discontinuous phrases. Table 6 shows that the decoder uses discontinuous phrases most if these two features are not present (last row). In this case, there is no way to distinguish discontinuous from continuous phrases, and the translation quality drops by 1.1 %BLEU on the development set and 1.6 %BLEU on the test set. The features seem to be required to penalize the application of discontinuous phrases.

Finally, the effect of the different phrase inventories is analyzed. The results are presented in Table 7. With the hierarchical phrase table, the number of sentence translations that use phrases with gaps is quite low compared to the discontinuous phrase table.[4] With the discontinuous phrase table, no improvement is achieved by adding a binary feature which enables the system to distinguish those gappy entries which are also extracted with the hierarchical approach.

[4]We would like to emphasize that *hierarchical* in Table 7 denotes the utilization of a phrase inventory with source-side gaps that has been produced with the hierarchical extractor. The search is conducted with the source cardinality synchronous search algorithm from Figure 2. No synchronous context free grammar (SCFG) formalism is pursued. See (Huck et al., 2012) for results on the same data with the SCFG hierarchical pipeline and a parsing-based cube pruning decoder.

pruning			MT06 (dev)		MT08 (test)		
RH	LH	gaps	BLEU [%]	TER [%]	BLEU [%]	TER [%]	GU
4	4	no	31.2	61.8	25.2	66.1	–
		yes	31.7	61.2	25.7	65.7	25.9 %
4	16	no	31.7	61.5	25.5	66.0	–
		yes	32.1	60.8	25.8	65.8	24.8 %
4	64	no	31.8	61.4	25.5	66.1	–
		yes	32.2	60.9	25.7	65.8	24.9 %
4	128	no	31.9	61.4	25.4	66.1	–
		yes	32.2	61.0	25.8	65.9	25.3 %
16	4	no	31.7	61.5	25.3	66.0	–
		yes	32.0	61.1	25.9	65.8	28.4 %
16	16	no	32.4	61.1	25.9	65.9	–
		yes	32.5	60.9	26.0	65.5	26.3 %
16	64	no	32.7	61.1	26.1	65.8	–
		yes	32.6	60.9	26.0	65.7	25.4 %
16	128	no	32.6	61.1	26.0	65.9	–
		yes	32.6	61.0	26.0	65.7	25.2 %
64	4	no	31.8	61.5	25.4	66.0	–
		yes	32.1	61.1	25.7	65.9	28.2 %
64	16	no	32.4	61.2	25.9	65.8	–
		yes	32.6	61.0	26.0	65.7	26.4 %
64	64	no	32.6	61.1	26.1	65.8	–
		yes	32.8	61.0	26.1	65.7	26.2 %
64	128	no	32.5	61.1	26.1	65.8	–
		yes	32.7	61.0	26.1	65.7	26.3 %
128	4	no	31.8	61.5	25.4	66.0	–
		yes	32.0	61.2	25.7	65.8	28.4 %
128	16	no	32.4	61.2	25.9	65.8	–
		yes	32.6	61.0	26.1	65.7	26.5 %
128	64	no	32.6	61.2	26.0	65.9	–
		yes	32.7	61.0	26.1	65.7	25.9 %

Table 5. Effect of pruning parameters. Results are reported in truecase.

4.2.3. Discussion

The discontinuous model does not yield significant improvements over the continuous baseline model. Indeed, both models perform on a similar level in almost all directly comparable system configurations. Galley and Manning (2010), in contrast,

features			MT06 (dev)		MT08 (test)		
isGappy	gapSize	gaps	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]	GU
–	–	no	32.6	61.1	26.1	65.8	–
yes	yes	yes	32.8	61.0	26.1	65.7	26.2 %
yes	no	yes	32.7	60.9	25.9	65.6	23.9 %
no	yes	yes	32.6	61.2	25.5	65.8	25.9 %
no	no	yes	31.7	62.1	24.5	66.9	69.2 %

Table 6. Effect of gappy features (with LH=64, RH=64). Results are reported in truecase.

		MT06 (dev)		MT08 (test)		
phrase table	gaps	Bleu [%]	Ter [%]	Bleu [%]	Ter [%]	GU
standard	no	32.6	61.1	26.1	65.8	–
hierarchical	yes	32.6	61.1	25.8	65.9	6.1 %
discontinuous	yes	32.8	61.0	26.1	65.7	26.2 %
discontinuous^{+HF}	yes	32.8	60.9	25.9	65.6	23.4 %

Table 7. Effect of the phrase inventory (with LH=64, RH=64). In the experiment marked with ($^{+HF}$), a binary feature has been added which enables the system to distinguish those gappy entries of the discontinuous phrase table which are also extracted with the hierarchical approach. Results are reported in truecase.

have reported uncased gains of +0.6 %Bleu on MT06 and +0.4 %Bleu on MT08 with source-side gaps (and without lexicalized reordering) in their system.[5]

First, we need to discuss whether the differences of the search organization in our decoder as compared to the system by Galley and Manning (2010) may be harmful. Galley and Manning (2010) do not prune reordering hypotheses and lexical hypotheses separately, and their decoder does not impose any reordering constraints in the manner of our phrase-level IBM reordering constraints. Apart from that, their pruning and maximum jump distance settings are rather more restrictive than those we utilized in our setups. Zens and Ney (2003) found that IBM constraints are quite limiting, e.g. as compared to ITG constraints. Regardless of that, reordering constraints and separate pruning of reordering and lexical hypotheses typically guide towards promising translations in an early stage of the search process. At least we would have

[5]The uncased Bleu scores of the system with *standard* phrase table from Table 7 are 34.7 on MT06 and 27.6 on MT08, the uncased Bleu scores of the system with *discontinuous* phrase table are 34.6 on MT06 and 27.6 on MT08. We furthermore ran these systems on MT02, MT04, and MT05, but did not observe any larger gains on any of these alternative test sets.

expected to see an advantage with discontinuous phrases over setups with standard phrases only as we increase the permissible amount of reordering. This is however not the case.

Another aspect we should consider is the quality of the word alignment and its suitability for the discontinuous translation model. We trained our word alignment with four IBM model 1, five HMM and four IBM model 4 iterations, Galley and Manning (2010) theirs with two IBM model 1 and two HMM iterations. We symmetrized our word alignment with the refined heuristic by Och and Ney (2003), which is comparable to the widely-used `grow-diag-final-and` heuristic (Koehn et al., 2003). It usually performs very well for standard phrase-based systems in our experience. Galley and Manning (2010) employ `grow-diag-final` for their hierarchical setup and `grow-diag` for the standard baseline and the discontinuous setup. It is possible that our standard heuristic works well for standard phrase-based translation, while discontinuous phrase-based translation might come up to its best performance based on different properties of the word alignment. We will try to empirically verify this supposition in future work.

5. Conclusion

In this work, a dynamic programming beam search algorithm for phrase-based statistical machine translation with coverage pruning per cardinality and lexical pruning per coverage (Zens and Ney, 2008) has been extended to support phrases with a discontinuous source part similar to (Galley and Manning, 2010). Two approaches to extract phrases with source parts that are allowed to contain gaps have been presented: the hierarchical approach and the discontinuous approach. The hierarchical phrase table is in fact a subset of the discontinuous phrase table. The differences have been discussed and analyzed empirically.

The experimental evaluation on the NIST Chinese→English translation task has been conducted with a focus on reordering constraints, pruning settings, and feature functions, as well as on the different phrase inventories. We found that the setups which employ source-side discontinuous phrases unfortunately barely outperform comparable setups which employ continuous phrases only. The translation quality as measured in Bleu remains at the same level. In future work, we intend to examine a possible impact of word alignment symmetrization heuristics.

Our implementations of the algorithms which we described in this paper have been released as part of Jane, the RWTH Aachen University statistical machine translation toolkit. The Jane toolkit is publicly available under an open source non-commercial license and can be downloaded from `http://www.hltpr.rwth-aachen.de/jane/`.

Acknowledgments

This work was partly achieved as part of the Quaero Programme, funded by OSEO, French State agency for innovation.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o} 287658.

Bibliography

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, June 1993.

Chiang, David. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proc. of the Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 263–270, Ann Arbor, MI, USA, June 2005.

Chiang, David. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33(2): 201–228, June 2007.

Galley, Michel and Christopher D. Manning. Accurate Non-Hierarchical Phrase-Based Translation. In *Proc. of the Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics (HLT-NAACL)*, pages 966–974, Los Angeles, CA, USA, June 2010.

Huck, Matthias and Hermann Ney. Insertion and Deletion Models for Statistical Machine Translation. In *Proc. of the Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics (HLT-NAACL)*, pages 347–351, Montréal, Canada, June 2012.

Huck, Matthias, Saab Mansour, Simon Wiesler, and Hermann Ney. Lexicon Models for Hierarchical Phrase-Based Machine Translation. In *Proc. of the Int. Workshop on Spoken Language Translation (IWSLT)*, pages 191–198, San Francisco, CA, USA, Dec. 2011.

Huck, Matthias, Jan-Thorsten Peter, Markus Freitag, Stephan Peitz, and Hermann Ney. Hierarchical Phrase-Based Translation with Jane 2. *The Prague Bulletin of Mathematical Linguistics*, (98):37–50, Oct. 2012.

Kneser, Reinhard and Hermann Ney. Improved Backing-Off for M-gram Language Modelling. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 181–184, Detroit, MI, USA, May 1995.

Koehn, Philipp, Franz Joseph Och, and Daniel Marcu. Statistical Phrase-Based Translation. In *Proc. of the Human Language Technology Conf. / North American Chapter of the Assoc. for Computational Linguistics (HLT-NAACL)*, pages 127–133, Edmonton, Canada, May/June 2003.

Lopez, Adam. Hierarchical Phrase-Based Translation with Suffix Arrays. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 976–985, Prague, Czech Republic, June 2007.

Mauser, Arne, Saša Hasan, and Hermann Ney. Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models. In *Proc. of the Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pages 210–217, Singapore, Aug. 2009.

Moore, Robert C. and Chris Quirk. Faster Beam-Search Decoding for Phrasal Statistical Machine Translation. In *Proc. of MT Summit XI*, Copenhagen, Denmark, Sept. 2007.

Och, Franz Josef. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. PhD thesis, RWTH Aachen University, Aachen, Germany, Oct. 2002.

Och, Franz Josef. Minimum Error Rate Training for Statistical Machine Translation. In *Proc. of the Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July 2003.

Och, Franz Josef and Hermann Ney. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proc. of the Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, PA, USA, July 2002.

Och, Franz Josef and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, Mar. 2003.

Och, Franz Josef, Christoph Tillmann, and Hermann Ney. Improved Alignment Models for Statistical Machine Translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28, University of Maryland, College Park, MD, USA, June 1999.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proc. of the Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, PA, USA, July 2002.

Rahman, Mohammad Sohel, Costas S. Iliopoulos, Inbok Lee, Manal Mohamed, and William F. Smyth. Finding Patterns with Variable Length gaps or Don't Cares. In *Proc. of the International Computing and Combinatorics Conf. (COCOON)*, pages 146–155, Aug. 2006.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proc. of the Conf. of the Assoc. for Machine Translation in the Americas (AMTA)*, pages 223–231, Cambridge, MA, USA, Aug. 2006.

Søgaard, Anders and Jonas Kuhn. Empirical Lower Bounds on Alignment Error Rates in Syntax-Based Machine Translation. In *Proc. of the Third Workshop on Syntax and Structure in Statistical Translation (SSST)*, pages 19–27, Boulder, CO, USA, June 2009.

Stein, Daniel, David Vilar, Stephan Peitz, Markus Freitag, Matthias Huck, and Hermann Ney. A Guide to Jane, an Open Source Hierarchical Translation Toolkit. *The Prague Bulletin of Mathematical Linguistics*, (95):5–18, Apr. 2011.

Stolcke, Andreas. SRILM – an Extensible Language Modeling Toolkit. In *Proc. of the Int. Conf. on Spoken Language Processing (ICSLP)*, Denver, CO, USA, Sept. 2002.

Vilar, David. *Investigations on Hierarchical Phrase-Based Machine Translation*. PhD thesis, RWTH Aachen University, Aachen, Germany, Nov. 2011.

Vilar, David, Daniel Stein, Matthias Huck, and Hermann Ney. Jane: Open Source Hierarchical Translation, Extended with Reordering and Lexicon Models. In *Proc. of the Workshop on Statistical Machine Translation (WMT)*, pages 262–270, Uppsala, Sweden, July 2010.

Vilar, David, Daniel Stein, Matthias Huck, and Hermann Ney. Jane: an Advanced Freely Available Hierarchical Machine Translation Toolkit. *Machine Translation*, 26(3):197–216, Sept. 2012.

Vogel, Stephan., Hermann Ney, and Christoph Tillmann. HMM-Based Word Alignment in Statistical Translation. In *Proc. of the Int. Conf. on Computational Linguistics (COLING)*, pages 836–841, Copenhagen, Denmark, Aug. 1996.

Wu, Dekai. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377–404, Sept. 1997.

Wuebker, Joern, Matthias Huck, Stephan Peitz, Malte Nuhn, Markus Freitag, Jan-Thorsten Peter, Saab Mansour, and Hermann Ney. Jane 2: Open Source Phrase-based and Hierarchical Statistical Machine Translation. In *Proc. of the Int. Conf. on Computational Linguistics (COLING)*, pages 483–491, Mumbai, India, Dec. 2012.

Zens, Richard. *Phrase-Based Statistical Machine Translation: Models, Search, Training*. PhD thesis, RWTH Aachen University, Aachen, Germany, Feb. 2008.

Zens, Richard and Hermann Ney. A Comparative Study on Reordering Constraints in Statistical Machine Translation. In *Proc. of the Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 144–151, Sapporo, Japan, July 2003.

Zens, Richard and Hermann Ney. Improvements in Dynamic Programming Beam Search for Phrase-based Statistical Machine Translation. In *Proc. of the Int. Workshop on Spoken Language Translation (IWSLT)*, pages 195–205, Honolulu, HI, USA, Oct. 2008.

Zens, Richard, Hermann Ney, Taro Watanabe, and Eiichiro Sumita. Reordering Constraints for Phrase-Based Statistical Machine Translation. In *Proc. of the Int. Conf. on Computational Linguistics (COLING)*, pages 205–211, Geneva, Switzerland, Aug. 2004.

Address for correspondence:
Matthias Huck
`huck@cs.rwth-aachen.de`
Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany

The Prague Bulletin of Mathematical Linguistics
NUMBER 99 APRIL 2013 39–58

The Design of Eman, an Experiment Manager

Ondřej Bojar, Aleš Tamchyna

Charles University in Prague, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics

Abstract

We present eman, a tool for managing large numbers of computational experiments. Over the years of our research in machine translation (MT), we have collected a couple of ideas for efficient experimenting. We believe these ideas are generally applicable in (computational) research of any field. We incorporated them into eman in order to make them available in a command-line Unix environment.

The aim of this article is to highlight the core of the many ideas. We hope the text can serve as a collection of experiment management tips and tricks for anyone, regardless their field of study or computer platform they use. The specific examples we provide in eman's current syntax are less important but they allow us to use concrete terms. The article thus also fills the gap in eman documentation by providing some high-level overview.

1. Introduction

Computational sciences including computational linguistics and computer science require broad experimenting to support theories and evaluate various techniques or methods. Very often, even the authors of some novel idea cannot guess the best possible method parameters and some form of search for them is desirable. This becomes more apparent if the method combines several independent modules or processing steps, each of which may or may not have been evaluated independently of the overall goal.

Another common aspect of natural sciences is the overarching strive for reproducibility. A novel method is never completely trusted until validated by a few independent laboratories, a program has to be tested and evaluated on a range of inputs and so on.

We pinpoint these two aspects of science by noting that: research = **re**producible **search**.

Corresponding author: bojar@ufal.mff.cuni.cz

In this article, we describe a very general tool that facilitates both reproducibility and search for the best configuration and parameters of complex experimental pipelines. Our eman also supports the collaboration of several people on the experiment.

Eman is open-source software, freely available for both non-commercial and commercial use.[1] The most recent version of the tool as well as other documentation is accessible at:

```
http://ufal.mff.cuni.cz/eman
```

The article is structured as follows: in Section 2, we explain what we perceive as the state-of-the-art techniques in efficient experimenting, highlighting the design goals of our new tool. Section 3 introduces our terminology and the basic building blocks of experiments in eman's terms. Section 4 summarizes the first area of eman's utility: navigation in the space of steps and experiments. Section 5 is devoted to the idea of cloning experiments and Section 6 describes the third key contribution: a general technique for collecting and interpreting the results. We conclude by introducing eman's support for teamwork (Section 7), related tools (Section 8) and our future plans (Section 9).

While we show some calls of eman commands in their exact syntax, the main goal of this article is to describe the underlying general ideas, not to serve as a reference guide for the tool. For this, the user is advised to the manual page of eman which can be obtained by running:

```
    eman --man
```

2. Design Objectives

The design of eman builds on our experience that the following features of experimentation environment are essential:

Reuse of results. In order to save both computation time and disk space, we need to reuse as many intermediate results as possible.

Encapsulation. Scientific experiments usually consist of complex sequences of processing steps, each carried out using a different tool that itself often needs some analysis, debugging, tweaking or optimization. To simplify switching and keeping focus, eman promotes encapsulation of each logical step into a separate directory. This directory should be as self-contained as reasonable, so when the researcher later inspects it, all the inputs and outputs are in one place.

Detailed records. Detailed logging of program outputs as well as of commands issued is essential for ensuring reproducibility, debugging and analysis of errors and comparison of results. We extend this to recording also the exact versions of (third-party) tools used in the experiment and also the procedure needed to

[1] Eman is licensed under the Creative Commons Attribution-ShareAlike License 3.0 (CC-BY-SA).

obtain and install the tools. This is achieved by treating the (source) code of the tools as input data of the experiment and including the compilation of the tools in the pipeline of the experiment. The reuse of intermediate results ensures the code is compiled only once.

Immutability. To simplify the record keeping, we opt for immutability of all data that is created in the experiment. Whenever some intermediate result is created based on some settings, `eman` never changes it. Modifications of the run are of course possible, but they always obtain a new identifier and reside in a new directory.

Hacking welcome. Admittedly, research prototype software is often quickly patched and far from anything that could be called a stable release. Furthermore, and this is a more important issue, research software does not always fit the purpose in new experiments. It is thus common that the tools have to be adapted or that a manual intervention is necessary after a random unexpected failure. `Eman` introduces a great deal of flexibility of experiment design – experiments are composed of individual steps which are further split into several lifetime stages – to allow for such an intervention.

Cloning. Research partially comprises of examining a range of minor modifications of a setup. In `eman`'s view, as it will be described below, experiments are defined by arbitrary variables and such setup modifications usually amount to setting these variables differently. Section 5 provides examples of one-line commands that take an existing experiment and apply a given set of modifications to it (such as setting a parameter differently or reversing the source and target language in an MT experiment). Finally, the necessary minimum of new processing steps are created and launched, reusing the steps common to both setups.

Cloning is in fact such a powerful idea that the relatively simple implementation of it in `eman` (regular expressions applied to experiment configuration files) allowed to create the tool Prospector, an automatic researcher (Tamchyna and Bojar, 2013). Prospector automatically searches the "space of possible MT systems" by evaluating various settings specified by its configuration file. The search can be guided by any metric, e.g. the well-known BLEU (Papineni et al., 2002) as calculated in the final evaluation step. Several search algorithms are implemented (greedy, exhaustive, genetic, random).

Prospector allows researches to avoid the tedious work of e.g. finding optimal parameters or meta-parameters for the MT decoder (beam size etc.) or any other experimental settings. It is freely available and distributed along with `eman`.

Parallelism. The parallelizations common in contemporary computer science (multiple processor cores, clusters of computers) allow for parallel execution of experiments. This is highly desirable because each individual experiment often takes a long time. Carrying out experiments in a strictly serial order would waste researchers' time and not fully exploit the available computational resources.

On the other hand, the researcher can easily lose track and focus when running many experiments in parallel.

Eman naturally allows to submit individual processing steps to a computer cluster, but more importantly, eman is designed to simplify the orientation in the large number of experiments already performed or in execution (see Section 4) and to some extent also the foreseen ones (see Section 6.3). The design also allows to derive (clone) new experiments from old ones even before the old ones complete.

Collaboration. The most recent feature of eman is the support for distributed experimenting. Currently we require a common filesystem (such as NFS), but that is reasonably easy to set up even across large distances. Individual processing steps can be launched by different researchers at different sites. The simple command "eman add-remote" issued once allows to include all the partial results of a remote site in the local environment. Circular inclusion is permitted allowing multiple researchers to "work at a common desk", reusing other people's processing steps (not just the programs but also the outputs of their particular runs), or to reinterpret their results (e.g. by creating new tabular views).

The same mechanism can be beneficial even for a single researcher as it allows to strictly separate some core source data (such as multiple training sets that nevertheless needed some preparation) from different branches of experiments.

Succinct notation. Shortcuts and abbreviations are very useful for improving the efficiency of the operating researcher. Eman provides shortcuts at several occasions, which is very useful e.g. for checking the status of the experiments over SSH in the cell phone.

3. Seeds, Steps, Experiments

Each *experiment* consists of atomic tasks called *steps*. In the context of MT, steps correspond e.g. to training a language model, translating a test set or running tuning. The individual steps depend on each other – the experiment is then a DAG (directed acyclic graph) of steps.

Each step has a type such as tm (translation model) or translate. The code which is executed when the step is run is generated by the corresponding *seed*. In the terminology of object-oriented programming (OOP), seeds can be viewed as classes and steps as their instances. Unlike in OOP, eman's positive stance to hacking allows different steps (instances) of the same type (class) run code customized arbitrarily, not just using proper subclassing.

In our particular implementation of eman, each step is simply a directory named using the pattern "s.steptype.abcHASH.20121215-1234" where the date and the hash value make the name unique. Seeds are then simply programs (in any language of the researcher's choice) which interpret some Unix environment variables and generate

```
+- s.compress.370c2483.20121108-1216
|  | CMD=bzip2
|  | CMDARGS=
|  | DATASTEP=s.data.aaf8c8b1.20121108-1149
|  +- s.data.aaf8c8b1.20121108-1149
|  |  | CMD="cat ../binary.test"
|  |  | SIZE=10000
|  |  | TYPE=binary
```

Figure 1. Example of an eman traceback.

executable code (again, in any language). The code is stored in the step directory and later run (once all predecessors are ready and the step is started).

`Eman` is used in a directory called *playground* – all steps are created there, based on seeds in the subdirectory `eman.seeds`. The "`eman add-remote`" allows to link remote playgrounds to the current one. By adding a remote playground, the directory structure is not changed but `eman` suddenly knows about steps coming from the remote playgrounds, it can show their properties and include them in local experiments.

3.1. A Sample Experiment

For illustration, we implemented a "compression playground" which provides an environment for evaluating compression algorithms. This sample playground contains two seeds:

data Imports data into the playground – the data can be generated by any command (specified by the variable `CMD`) and the user can limit the amount of data using the variable `SIZE`.

compress Given some data, compress it using the command given in the variable `CMD` with some optional `CMDARGS` and calculate the compression rate.

Figure 1 shows an example of an experiment in this playground in `eman`'s format. This *traceback* is a full definition of the experiment. The seeds were "instantiated" to steps with some variable values (e.g. the compression command is `bzip2`) and connected to form a DAG – note that the dependency is explicitly captured in the variable `DATASTEP`.

3.2. Lifetime of a Step

Figure 2 depicts the lifetime of a step.

New steps are created using "`eman init STEPTYPE`". `Eman` creates a new directory in the current playground and copies the corresponding seed into it. Then the seed is executed – at this stage, the seed only performs basic sanity checks to determine

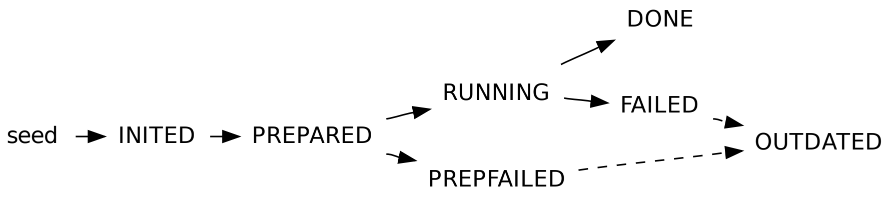

Figure 2. The lifetime of a processing step in eman

whether all required variables are defined etc. If everything succeeds, the step is registered in `eman` and receives the status `INITED`.

The seed is executed once more when the user runs "`eman prepare SPEC`". At this point, the seed creates an executable file `eman.command` which contains the step code. `Eman` checks whether the file was created and sets the step status to `PREPARED`.

Finally, the user runs "`eman start SPEC`" and the step is started. Its status changes to `RUNNING`. Once the step terminates, its status is either `DONE` or `FAILED`.

Steps at any stage, including the `FAILED` ones, can still serve as a basis for creating new steps with the same or similar variables, see below. For the purposes of marking that the user has already handled a failure, one more state, `OUTDATED`, was introduced. Upon request ("`--outdate`") `eman` not only creates a new instance of a failed step but also moves the failed one to the outdated state.

There are many shortcuts for user convenience – "`eman start`" on an `INITED` step automatically runs "`eman prepare`". The whole process of creating and running a step can even be done in one command (and it often is): "`eman init --start`".

The acyclicity of the lifetime diagram (no directed loops) is in line with the design objectives of immutability and detailed records. One should want to keep the logs of a failure and redo the job in a fresh instance of the step. (Indeed, this is what the command "`eman redo`" does, see Section 5.2.) In practice, a step can be very costly and fail at some late stage of execution. It would be wasteful to rerun it from scratch. The fact that each step includes its code (the file "`eman.command`", cf. the encapsulation objective) allows to manually fix this code and to jump to a recorded point shortly before the failure. After these manual changes in "`eman.command`", calling "`eman continue`" puts the failed step back to the `RUNNING` state and submits it to the cluster again (or runs it locally, depending on the user's environment).

3.3. Motivation for the Three Stages

What are the benefits of breaking the execution of a single processing step into the stages of initialization, preparation and the run itself?

The *initialization* is vital, it turns a blank directory into a valid `eman` step with variables defined. From this point on, the step can be incorporated into complex experiments and it can be used as a basis for cloning. There is thus no need to wait until the step finishes, we can plan ahead (and even submit for execution) other steps that will build on the future outputs.

After the initialization, the user has a chance to tweak the seed (and thus influence the actual command that will be performed). This is the point where we depart the OOP by allowing different instances run customized code.

The init phase should be very quick, it is run interactively and often repeated for many steps when cloning whole experiments.

The *preparation* phase is thus meant to get all input data in place, so that the user can check them before the actual computation, e.g. submitting the step to the cluster. In our MT experiments, the preparation phase was originally responsible for things like cutting a given subsection or subset of annotation features from the training data. As our training data grew, running these filters during the (still interactive) preparatory phase became inconvenient, so we changed our seeds and shifted even the obtaining of input data into the actual run. For `eman`, this makes no difference. The choice what happens at what phase is entirely up to the user; any of the phases can be even empty.

3.4. From Steps to Experiments

Steps are combined to form experiments. There is no pre-defined interface for communication between steps. Each step has access to its (direct) predecessors via variables – it can extract whatever files the previous step has created from its directory. The same step can thus serve several purposes at once, it just needs to produce outputs relevant to the respective successors.

The initial setup of experiments is somewhat tedious, the user has to run a sequence of commands like:

```
    SOURCEDATASTEP=s.mydata.12345678.20121215-1234 eman init myproces-
sor
```

The user has to manually set the variable `SOURCEDATASTEP` to the name of the previously initialized `mydata` step. Once the full cascade of steps, i.e. an experiment, is set up, it is easier to derive variations of it using cloning, see Section 5.

3.5. Referring to Steps vs. Experiments

Note that the pointer to a step directory "`s.steptype.123`" can mean either just the single step that was carried out in the directory, or the whole experiment, i.e. the directed acyclic structure of steps that culminates with the given step. These two notions should not be confused.

It is the particular `eman` command that resolves the ambiguity between a step and an experiment. So for instance, "`eman prepare`" prepares an individual step regardless the status of its predecessors. Depending on what a particular step requires, the preparation may fail because the predecessors are still in the `INITED` state only and do not provide relevant data. The command "`eman start`" is more useful as it operates on the whole *experiment*. In other words, it ensures that the whole DAG of steps is first `PREPARED` and then submits all steps that were not finished yet to the cluster, introducing any necessary job dependencies.

4. Navigation in the Playground of Steps and Experiments

As the user creates experiments or derives clones of them (Section 5), the playground becomes quickly filled with step directories of unhelpful names like "`s.tm.1a53fg63.201`
Finding a particular step can then be difficult and time-consuming. This section briefly summarizes the four main techniques `eman` provides to ease the navigation in the playground: listing details of individual steps, finding (selecting) steps with given properties, examining the structure of experiments, i.e. how steps depend on one another, and manually tagging steps.

One more aspect of playground structure remains to be harnessed in a future version of `eman`: the history how steps were derived from other steps. Currently, `eman` only records the immediate origin of a derived step in the file "`eman.derived_from`".

4.1. Listing Steps and Their Details

The command "`eman ls`" prints steps in the current playground. The user can filter the listing based on the step type and request additional information – most importantly step variables, status and tags (see Section 4.4) – using command-line options. The following example query returns all steps of the type "`align`" and prints their variables, status and disk usage:

```
eman ls align --vars --stat --dus
```

Some shortcuts are again provided by means of commands `eman vars`, `stat` and `tags` that print the required information about all steps, or a particular step:

```
eman vars s.tm.1a5
```

Note that it is not necessary to specify the full step name, any part of it (not necessarily the beginning) long enough to make it unique within the playground is sufficient.

4.2. Finding Existing Steps

The command "`eman select`" (optionally abbreviated to "`sel`") provides a flexible means for finding steps with specified properties. The following few examples are just a brief demonstration of the query language.

- Steps which were created today and failed:

```
eman sel today f
```

- Last (most recent) five steps of the type "`align`":
  ```
  eman sel t align l 5
  ```
- Language model steps (i.e. steps of the type "`lm`") trained on word lemmas (`vre` stands for "a **v**ariable matches **r**egular **e**xpression"):
  ```
  eman sel t lm vre lemma
  ```
- Language models of order other than three (note the word `not`):
  ```
  eman sel t lm not vre ORDER=3
  ```
- MERT (Och, 2003) steps with a (possibly indirect) predecessor of the type `align` whose variables match the expression "lemma" (presumably a word alignment step done on word lemmas; `br` stands for backward recursion and matches properties in preceding steps):
  ```
  eman sel t mert br t align vre lemma
  ```
- Translation model ("`tm`") steps which were evaluated on a given test set (`fr` means forward recursion):
  ```
  eman sel t tm fr vre TESTCORP=wmt12
  ```

The syntax is very succinct which allows to write complex queries with very little effort. Users of `eman` frequently log in to their cluster using cell-phone SSH and type simply "`eman sel f`" to see if any experiments need their attention.

4.3. Dependencies and Users of Steps

`Eman` provides commands to list predecessors and successors of steps. Direct predecessors (dependencies) can be obtained using the command "`eman deps`" while "`eman traceback`" or "`eman tb`" prints the full *traceback* (i.e. DAG) of steps.

`Eman` assumes that an experiment is defined by the structure of dependencies and the values of their variables; this implies that the command "`eman tb --vars FINAL-STEP`" outputs a full, unambiguous specification of the whole experiment.

An example of a traceback with variables was already given in Figure 1.

Analogously to the predecessors, direct and indirect successors can be listed using the commands "`eman users`" and "`eman tf`" (traceforward), respectively.

4.4. Tagging of Steps

Steps that are somehow special or often referred to, e.g. because they were manually tweaked before submission or because they represent the baseline or the current best result, can be "tagged". Tags are simple labels associated with a particular step.

Tags are assigned to steps using the command "`eman add-tag`":

```
eman add-tag BASELINE s.evaluator.123456
```

Later, tags can be used as step identifiers as long as they are unambiguous. So we can e.g. double check what was our baseline configuration:

```
eman tb --vars BASELINE
```

The tags are stored in the step directory in the file `eman.tags`. Upon re-tagging ("`eman retag`"), the labels are recursively propagated to the step successors (however their `eman.tags` files do not change, the propagation is done in `eman`'s internal index only). While this is useful for organizing results, see Section 6, it makes tags refer to more steps and thus no longer usable as step specifiers. In future versions of `eman`, we may thus remove or somehow restrict the tag propagation feature.

5. Cloning of Experiments

The previous sections described techniques for reusing intermediate results across experiments. Now we describe `eman` commands that allow to reuse the configurations of individual steps and whole experiments. The added twist is that when an existing experiment is "cloned", the variables may be arbitrarily changed.

5.1. Replicating Individual Steps

Cloning of an existing step means creating a new instance of the same step type, reusing most of the variable values. For instance, we may want to create somewhat larger test case for our compression experiment, and we already have the data step "`s.data.aaf8`" ready, as illustrated in Figure 1. The following command will create a new instance using the data seed and run it right away:

```
SIZE=500000 eman clone s.data.aaf8 --start
```

The above command works as an abbreviation of "`eman init data`" where all the variables would have to be specified:

```
 SIZE=500000 CMD="cat ../binary.test" TYPE=binary eman init data
--start
```

The cloning will work even without changing any of the variables. In general, it is better to avoid multiple runs of the same configuration, but some computations are non-deterministic and running several copies allows to estimate confidence intervals of the result (Clark et al., 2011). In MT, the prototypical example is the minimum error-rate training, MERT. Creating four more replications of a MERT run is trivial:

```
for i in 2 3 4 5; do eman clone --start s.mert.123; done
```

Replicating a step with identical variables is also useful when a step fails. The command "`eman clone`" as described so far operates on individual steps, so any (failed) dependencies will not get recreated. A better option is described in the following section.

5.2. Redoing Experiments

When an experiment fails, "`eman redo`" can be used to re-create the necessary steps in the whole experiment pipeline. Redo will check the whole traceback of the given experiment and replicate any steps that are failed or outdated. When doing this, the correct links between dependencies are honored, so whenever a step gets redone, its successors will get redone as well.

With large-scale experiments, various technical problems often come into play, making the redo command very useful in day-to-day experimenting. A particular common reason for a failure is a full local temporary disk or memory limits set too low for the given input data, which leads to jobs being killed by the cluster. `Eman`, in cooperation with the scheduling environment, can set the requirements on available memory and disk, so the following usage pattern is quite common:

```
eman redo s.myFailedExp.123 --mem 30g --disk 80g --start --outdate
```

Note that "`eman redo`" walks only the traceback, not the traceforward of the given experiment. It is thus important to ask for a redo of the *final* steps of failed experiments.

5.3. Deriving Whole Experiments

By mixing the idea of modifying variables and redoing whole experiments, we arrive at the full power of experiment cloning.

We have already mentioned, that the traceback with variables (i.e. the output of "`eman tb --vars FINALSTEP`") is the complete description of an experiment. The user can modify some variables in the textual form of the traceback and clone it:

```
eman clone < traceback.modified
```

When constructing steps from such a textual traceback, `eman` automatically discovers steps which can be re-used and only creates the parts of the experiment which are really needed. Experienced `eman` users often create the traceback, substitute some values and create the modified experiment on one line:

```
eman tb -s /oldvalue/newvalue/ | eman clone --dry-run
```

The parameter `-s` defines a substitution which is applied on the whole traceback and supports full Perl regular expressions. The "`--dry-run`" is useful for a quick check before creating the many step directories or "`--start`"ing the new experiment.

We have found cloning of experiments to be extremely useful and versatile in practice. Multiple settings of an MT system can be created and evaluated easily by defining the base experiment and cloning it several times with modified variable values.

With cloning, e.g. reversing the translation direction of an experiment is a trivial change. Similarly, one can easily repeat an experiment for multiple language pairs, change datasets, adjust language model order or modify factors for word alignment.

```
data     var /SIZE=(.*)/SIZE$1B/            /000B\$/kB/
data     var /TYPE=(.*)/TYPE$1/
compress var /CMD=(.*)/CMD$1/
compress var /CMDARGS=(.*)/ARGS$1/
compress var /CMDARGS=.*?-([0-9]).*/LEVEL$1/
```

Figure 3. Sample “`eman.autotags`” *configuration.*

6. Making Sense of Results

By a *result*, we mean a small token, usually a number, that was observed or measured during the run of an experiment. In `eman`'s view, results are small bits of information available somewhere in the output files of a step.

`Eman` provides a set of tools for collecting and interpreting results.

6.1. Autotagging (Tags Based on Variables)

We have already introduced manual tags (Section 4.4) that can be later used to identify e.g. results based on a particular dataset or using a particular version of a program. In addition to tags, `eman` provides “autotags” that are created automatically from variables of steps using regular expressions and substitutions. The main purpose of automatic tags is to select relevant information from the variables and make it available for the interpretation of results, see below.

The user configures automatic tagging by writing rules into the file “`eman.autotags`”. Each rule consists of the type of steps to which it applies, a regular expression that is matched against the step variables and optionally a regular-expression substitution to be applied on the match – to beautify it in a way.

The tagging rules implemented for our compression example are shown in Figure 3. Each line defines one rule – on the first line, we tell `eman` to match variables of `data` steps and look for the pattern “`SIZE=.*`”. We extract the size and prefix it with the word “`SIZE`”. We also perform a simple substitution to shorten the value – we replace “`000B`” with “`kB`”. The `data` step shown in Figure 1 is assigned the tag “`SIZE10kB`” according to this rule.

6.2. Collecting Results

The first task when working with results is to collect them from all the many steps in the playground to a single place. This is achieved using the command “`eman collect`”: all results will appear in the file “`eman.results`” in the playground directory.

```
s.compress.ba3c96ac.20121217-1415 DONE ratio .47770000000000000000  ARGS-5 CMDbzip2 LEVEL5 SIZE10kB TYPEhexrand
s.compress.bb9c7f1e.20121217-2145 DONE ratio .52478125000000000000  ARGS"-5 --rsyncable" CMDgzip LEVEL5 SIZE512kB TYPEhexrand
s.compress.c45a5afd.20121217-1414 DONE ratio .53680000000000000000  ARGS-9 CMDgzip LEVEL9 SIZE10kB TYPEhexrand
s.compress.99ab03f6.20121217-1436 DONE ratio .43619140625000000000  ARGS-5 CMDbzip2 LEVEL5 SIZE512kB TYPEhexrand
s.compress.02a5f93f.20121217-1436 DONE time  0.042                  ARGS-5 CMDgzip LEVEL5 SIZE512kB TYPEhexrand
s.compress.0545633f.20121217-1413 DONE time  0.002                  ARGS-4 CMDgzip LEVEL4 SIZE10kB TYPEhexrand
s.compress.07151d39.20121217-0116 DONE time  0.003                  ARGS CMDgzip SIZE10kB TYPEhexrand
s.compress.c45a5afd.20121217-1414 DONE TAG   ARGS-9 CMDgzip LEVEL9 SIZE10kB TYPEhexrand
s.compress.7694fe26.20121217-1436 DONE TAG   ARGS CMDbzip2 SIZE512kB TYPEhexrand
```

Figure 4. A few random sample lines from the file "`eman.results`*". Each line contains the step name, its status, the name of the result and its value and finally all the tags and autotags assigned to this step.*

The specification, what exactly should `eman` extract from a step directory, is provided by the user in the file "`eman.results.conf`". For instance, the configuration line:

```
ratio  →  s.compress.*/ratio  →  CMD: cat
```

specifies that steps of the type "`compress`" measure a particular property, namely the compression ratio that they achieved on some give data. The value can appear anywhere in a file in the step directory as long as a Unix one-line command can extract it. Here, the file "`ratio`" contains just the value of interest, so simply `cat`ting it does the job.

The possibility to run a custom "result extractor" makes collecting of results very flexible: anything can be made important. One can easily introduce new properties to observe at any later time, as long as they were recorded somewhere. Together with remote playgrounds (Section 7), one can re-interpret other people's experiments.

The file "`eman.results`" is useful on its own already. For instance, the user can quickly check if the top-scoring setup is still the same, e.g.:

```
grep ratio eman.results | sort -rn -k4 | head -n 1
```

For the purposes of the following section, we provide a snippet of the results file in Figure 4.

6.3. Tabulation of Results

Lonesome numbers do not have any meaning. In order to be able to interpret the observations and discuss them, individual results have to be compared and contrasted to other results. One practical issue is that a set of results can be dissected and contrasted in an endless number of ways.

`Eman` provides a succinct but extremely powerful tool for "putting relevant numbers next to each other". The technique is based on the "`eman.results`" file and one more user configuration file, "`eman.tabulate`". Running "`eman tabulate`" reorganizes the results based on the configuration and produces "`eman.niceresults`".

```
=== Compression ratios of different algorithms ===
(512k of random hex data)

TABLE
required: compress ratio
required: SIZE512
forbidden: OUTDATED LEVEL[2-46-8]
cols: CMD([^\s]*)
rows: LEVEL([0-9]) rsyncable
rowsort: CMDgzip
ENDTABLE
```

Figure 5. Sample "`eman.tabulate`" *configuration.*

6.3.1. Prose with Automatic Tables

The file "`eman.tabulate`" is a regular text file. Any comments, observations or discussion can be simply written there. `Eman` copies everything verbatim, except for sections surrounded by lines saying "`TABLE`" and "`ENDTABLE`". These sections will get expanded to tables of results. The number of tables in the file is not limited and each table can provide a different view of the results.

One can in principle use "`eman.tabulate`" as the LaTeX source of a scientific paper where tables are constructed automatically from the available results.

In the following sections, we describe how `eman` processes the configuration given in Figure 5 to obtain the table in Figure 6.

6.3.2. Selecting Results to Show

The first stage of tabulation is the selection of lines from "`eman.results`" that should be listed in the table. This filtering allows to provide different views on the playground.

The filtering is achieved by two sets of regular expressions. Only the lines that contain all the "`required`" expressions and do not contain any of the "`forbidden`" expressions make it to the table.

Technically, the regular expressions are delimited by space in the "`eman.tabulate`" config, so a single "`required:`" line can specify several requirements. To match a space, one can use e.g. "`\s`".

As each line of the results file contains a lot of details (see Figure 4), the filtering is quite powerful: we can even match e.g. the date in the step name to require steps inited during a particular day. In our example in Figure 5, we are interested in the "`ratio`" results of any "`s.compression.*`" step. The autotags provide the information

```
=== Compression ratios of different algorithms ===
(512k of random hex data)

Common properties: compress ratio SIZE512
Forbidden properties: OUTDATED LEVEL[2-46-8]

                 CMDbzip2              CMDgzip
LEVEL1           .46033593750000000000 .55463281250000000000
LEVEL5 rsyncable                     - .52478125000000000000
LEVEL5           .43619140625000000000 .51907031250000000000
                 .43503125000000000000 .51825585937500000000
LEVEL9                               - .51824414062500000000
```

Figure 6. Sample results of the tabulation.

about the file size that was used in the experiments and we require the 512kB tests. Finally, we avoid all steps in the `OUTDATED` state and we also exclude some compression levels ("`forbidden: LEVEL[2-46-8]`") to make the table shorter.

6.3.3. Constructing Row and Column Label for a Result

Each result (i.e. a single line from the results file) that survives the filtering is examined in order to construct its "`row`" and "`col`"umn label. The same regexp mechanism as above is used here, except now the successfully matched regexp is appended to the respective label.

In our example, the columns are simply the compression algorithms – these are found in the autotag starting with "`CMD`". Rows are a little bit more interesting. In general, we want to see the compression level ("`LEVEL([0-9])`"), but in a few experiments, we also used the gzip flag "`--rsyncable`", so we need to distinguish these runs. Adding a second regex "`rsyncable`" that may or may not be found in the result line makes the distinction.

The round brackets in the regexes express important parts of the match. The respective portion of the regex will be replaced by the actual string matched. So the "level" regex appears as a few distinct tokens like "`LEVEL1`", or "`LEVEL9`" in the final table, see Figure 6. Without the round brackets ("`LEVEL[0-9]`"), we would get the same token for all lines, namely "`LEVEL[0-9]`". This can be useful to skipping unimportant differences, i.e. specifying a "gappy pattern".

The order of the regexes is also important, because labels are constructed left-to-right. So regexes that construct the beginning of row or column label need to appear in the configuration first.

Not all result lines match all row/column regexes. That is fine, the label is then simply shorter. As an example, we see the default run of the two compression algorithms where the row label is empty – no level was specified at all.

Not all settings are meaningful or used across all experiments. This is also fine, the cells will then contain just a dash. In our example, it is the "rsyncable" option, which is not available in bzip2, and the level-9 bzip2 experiment which we forgot to run for the purposes of Section 6.3.6.

There are a few other minor tricks for handling cases like multiple matches of the same regex, but these are beyond the scope of this article.

6.3.4. Putting the Table Together, Solving Conflicts

Having established the row and column labels for each value, it is trivial to construct the table. Values sharing the column label will appear in the same column, values sharing the row label will appear in the same row. This gives us a two-dimensional view on the results.

If two or more distinct values share the same row *and* column label, `eman` reports a conflict and the user has two options. If such a conflict is not desirable then some regex (and perhaps also some tag or autotag) should be added to filter out unwanted values or put the conflicting values on different rows or columns. There are however cases where we have deliberately run the very same experiment several times and some randomness or outside condition leads to different results. In this case, one adds the following line to the table specification:

```
collectdelim:,
```

This switch instructs `eman` to indeed show all the results in a single cell, delimited by the given string (a comma in our example).

6.3.5. Sorting Rows and Columns

Finally, the user can specify the full label of the column that should be used to sort the rows ("`rowsort`") and/or the full label of the row that should be used to sort the columns ("`colsort`").

Note that adding regexes that construct row and column labels can easily change the labels so sorting fails to find the given criterion.

6.3.6. Back to Experimenting

`Eman` consults the file "`eman.results`" when resolving step specifiers. This neat trick allows to go directly back from the (tabulated) results to experimenting.

We can ask questions like: what exact configuration did I use to produce the compression ratio 0.5368:

```
eman tb --vars 5368
```

It is wise to double-check that the numbers we contrasted by putting them on the same line or column actually differ only in the properties we are mentioning. In bash, this amounts to inspecting the diff of the two tracebacks, e.g. in the editor vim:

```
vimdiff <(eman tb --vars 4361) <(eman tb --vars 5190)
```

It is also easy to use the cloning mechanism (Section 5.3) to start experiments whose results will fill missing cells. We pick an existing result from the given row (or column, whichever is more convenient) and apply the necessary change to it. We exemplify it by filling the level-9 compression experiment by bzip2 that was missing in Figure 6. The bzip2 run is derived from the corresponding gzip experiment:

```
eman tb 518244 -s /gzip/bzip2/ | eman clone --start
```

7. Team Experimenting

The command "`eman add-remote`" is implemented in a very light-weight fashion. The user provides the path to the remote playground and an alias – `eman` then simply creates a symbolic link to the directory in the local playground and registers the link in the file `eman.subdirs`.

Remote steps then become equivalent to steps in the local playground – they can be used in experiments, cloned and even modified (e.g. started, outdated) if the file system permissions allow it (otherwise, `eman` automatically switches to read-only mode).

`Eman` does not search the remote playground recursively (i.e. it does not explore its remote playgrounds), which makes this feature quite flexible; even circular dependencies are possible, although they do create a soft-link loop in the filesystem.

Commands such as "`eman ls`" or "`eman select`" list only local steps by default. To consider remote playgrounds, the option "`--remote`" has to be used. `Eman` can also display the playground of each step in the listing if "`--dir`" is given.

Finally, since step directories are no longer local subdirs of the playground, the command "`eman path`" is useful to get the full pathname of a step.

8. Related Tools

Two similar tools come from the MT environment: Ducttape and EMS. Ducttape (formerly LoonyBin; `https://github.com/jhclark/ducttape`; Clark et al., 2010) is functionally similar to the combination of `eman` and Prospector (included in `eman` package). The user specifies "hyperworkflows", packed sets of experiments, where a number of variables has a number of requested values. Hyperworkflows are actually more flexible than that, separate hyperworkflow branches can have different step structure. Given a hyperworkflow, Ducttape runs either the full Cartesian product of variable values or a subset of it based on some "realization plan". Implemented in Java, it originally provided only a graphical user interface but now there is also a command-line interface and a minimal web-interface available.

Experiment Management System (EMS; Koehn, 2010), is distributed with the Moses translation system (Koehn et al., 2007) and it is primarily intended for it. Its general management capabilities are again centered around distinct runs of the complete experiment. Data reuse is achieved by noticing that some partial output from a previous run is still valid. This is against our encapsulation objective.

Taverna (`http://www.taverna.org.uk/`) is a widely used complex workflow management tool. It introduces the Taverna language to describe workflows, provides a graphical user interface including an editor of workflows and various servers and clients for running workflows or providing services that can be used as processing blocks in workflows remotely. The remote processing is perhaps the biggest advantage: research institutes provide web-based services directly usable in user's workflows. Compared to `eman`'s 4k lines of Perl, Taverna's command-line tool is 151 MB. Taverna originated in bioinformatics but it is being used in many other fields of research. The only Taverna application in NLP so far are probably the PANACEA tools (`http://www.panacea-lr.eu/`) for compiling various linguistic resources from texts.

Cluster or grid computing environments, e.g. Pegasus (`http://pegasus.isi.edu/`), also have workflow managers like DAGMan (Couvares et al., 2007). These allow to express dependencies between jobs but focus on automatic recovery from job failures in an unreliable cluster environment, not on experiment variation or any interpretation of results.

9. Open Issues and Future Development

There are certainly limitations of the current version of `eman`. The most serious issue from the practical point of view is that the indexing of steps walks many directories and files.[2] With a larger number of steps, this becomes inconveniently slow. A principled solution would use clever incremental updates of only the bits that got invalid due to some change. Unfortunately, this is rather tricky: e.g. changing the autotag configuration would require to propagate new tags to existing steps etc. but `eman` does not get automatically called when the user edits the file "`eman.autotags`".

We have also mentioned, that some inspection and reuse of the *derivation history* for steps is desirable. This would allow further shortcuts in experimenting and new types of observations, e.g. why does the foobar switch make the baseline experiment faster but it slows down our improved setup?

Finally, `eman` has no visual output, but it would be quite easy to display the various dependencies between steps using e.g. the graphviz library (Gansner and North, 2000).

[2] `eman` accumulates an index of steps during regular operations. Full reindexing is required only occasionally and done upon request ("`eman reindex`" for the core index of steps and their variables, "`eman retag`" for autotag application and propagation and "`eman collect`" for the collection of results).

10. Conclusion

We presented `eman`, an open-source experiment manager for command-line Unix environment.

Hopefully, we highlighted and explained a couple of ideas that are generally useful for speed up and a better guidance of experimenting. We feel the following features are the most important ones: keeping detailed records, reusing intermediate results and reusing whole experiments by cloning new variants of them. We also provided a couple of suggestions for organizing and examining obtained results.

For readers interested in `eman` specifically, this article should provide a high-level overview spiced with example calls and commands.

11. Acknowledgment

The work on this project was partially supported by the grants P406/11/1499 of the Grant Agency of the Czech Republic, FP7-ICT-2011-7-288487 (MosesCore) of the European Union and 7E11042 of the Ministry of Education (EU project FP7-ICT-2010-6-257528).

Bibliography

Clark, Jonathan H., Jonathan Weese, Byung Gyu Ahn, Andreas Zollmann, Qin Gao, Kenneth Heafield, and Alon Lavie. The Machine Translation Toolpack for LoonyBin: Automated Management of Experimental Machine Translation HyperWorkflows. *Prague Bulletin of Mathematical Linguistics*, 93:117–126, 2010. ISSN 0032-6585.

Clark, Jonathan H., Chris Dyer, Alon Lavie, and Noah A. Smith. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proc. of ACL/HLT 2011*, pages 176–181, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-2031`.

Couvares, Peter, Tevfik Kosar, Alain Roy, Jeff Weber, and Kent Wenger. Workflow Management in Condor. In *Workflows for e-Science*, pages 357–375. Springer London, 2007. ISBN 978-1-84628-519-6. doi: 10.1007/978-1-84628-757-2_22. URL `http://dx.doi.org/10.1007/978-1-84628-757-2_22`.

Gansner, Emden R. and Stephen C. North. An open graph visualization system and its applications to software engineering. *SOFTWARE - PRACTICE AND EXPERIENCE*, 30(11): 1203–1233, 2000.

Koehn, Philipp. An Experimental Management System. *Prague Bulletin of Mathematical Linguistics*, 94:87–96, Sept. 2010. ISSN 0032-6585.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of ACL 2007, 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`.

Och, Franz Josef. Minimum error rate training in statistical machine translation. In *Proc. of ACL 2003*, pages 160–167, Sapporo, Japan, 2003. ACL. URL `http://aclweb.org/anthology-new/P/P03/#1000`.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proc. of ACL 2002*, pages 311–318. Association for Computational Linguistics, 2002. URL `http://aclweb.org/anthology-new/P/P02/`.

Tamchyna, Aleš and Ondřej Bojar. No free lunch in factored phrase-based machine translation. In *Proc. of CICLing 2013*, volume 7817 of *Lecture Notes in Computer Science*, pages 210–223, Samos, Greece, 2013. Springer-Verlag.

Address for correspondence:
Ondřej Bojar
`bojar@ufal.mff.cuni.cz`
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics,
Charles University in Prague
Malostranské náměstí 25
118 00 Praha 1, Czech Republic

The Prague Bulletin of Mathematical Linguistics
NUMBER 99 APRIL 2013 59–75

Accounting for Contiguous Multiword Expressions in Shallow Parsing

Matthieu Constant[a], Olivier Blanc[b], Patrick Watrin[b]

[a] Université Paris-Est, LIGM, CNRS, France
[b] Université de Louvain, CENTAL, Belgique

Abstract

In this paper, we focus on chunking including contiguous multiword expression recognition, namely *super-chunking*. In particular, we present different strategies to improve a super-chunker based on Conditional Random Fields by combining it with a finite-state symbolic super-chunker driven by lexical and grammatical resources. We display a substantial gain of 7.6 points in terms of overall accuracy.

1. Introduction

Multiword expressions (MWEs) are key parts of Natural Language Processing (Sag et al., 2002). But they have long been neglected in empirical parsing researches. Preliminary works like Nivre and Nilsson (2004); Arun and Keller (2005) integrated such expressions in parsers but by considering a gold MWE recognition. In recent pioneer studies, realistic MWE recognition has started to be integrated in shallow parsers (Korkontzelos and Manandhar, 2010) and in deep parsers (Cafferkey et al., 2007; Green et al., 2011; Constant et al., 2012).

In this paper, we focus on shallow parsing, more precisely on chunking including multiword expression recognition, namely *super-chunking* (Blanc et al., 2007). The considered MWEs are contiguous and consist of compounds, nominal collocations, multiword terms, named entities like dates, person and organization names, etc. Historically, chunking can conveniently be implemented with cascades of finite-state transducers (Abney, 1996; Joshi and Hopely, 1997; Ait-Mokhtar and Chanod, 1997; Nasr and Volanschi, 2005). Blanc et al. (2007) successfully extended this finite-state frame-

Corresponding author: mconstan@univ-mlv.fr
Cite as: Matthieu Constant, Olivier Blanc, Patrick Watrin. Accounting for Contiguous Multiword Expressions in Shallow Parsing. The Prague Bulletin of Mathematical Linguistics No. 99, 2013, pp. 59–75.
doi: 10.2478/pralin-2013-0004.

work to *super-chunking*. They implemented a super-chunker, namely POM, mainly driven by large-scale lexical and grammatical finite-state resources. Shallow parsing can also be seen as a sequence annotation task (Ramshaw and Marcus, 1995) that is very well modeled by discriminative models (e.g. Kudo and Matsumoto, 2001; Sha and Pereira, 2003; Tsuruoka et al., 2009).

In this paper, we consider the use of Conditional Random Fields (CRF). Such approach relies on a reference annotated corpus. But, what happens if this corpus does not entirely comply with our needs in terms of super-chunk annotations? Especially, what happens if some expected types of MWEs are not encoded? In the following, we propose different strategies to adapt a CRF-based super-chunker to our needs by combining it with a symbolic super-chunker driven by lexical and grammatical resources (like POM). The proposed solutions are all evaluated on French as many MWE resources are available for this language.

The paper is organized as follows. In Section 2, we define super-chunking and our target annotations. Section 3 is devoted to the description of the available resources. Then, in Section 4 we present a simple CRF-based super-chunker that we consider our *baseline*. Next, we detail POM architecture (Section 5). Finally, we present two combination solutions (Section 6) and evaluate them (Section 7).

2. Super-chunking

2.1. Super-chunks

Super-chunks are non-recursive syntactic constituents, like standard chunks (Abney, 1991), with the difference that they can contain multiword expressions, as we defined it in Blanc et al. (2007). For instance, *marge d'exploitation* (trading margin) is considered a compound noun, so the utterance *la marge d'exploitation* (the trading margin) is annotated as a nominal super-chunk (XN), while standard chunking would have produced a sequence of a noun phrase (XN) followed by a prepositional phrase (XP).[1] Considering super-chunks instead of standard chunks has two main interests: (1) it reduces combinatorial complexity for shallow parsing because some ambiguities are resolved with MWE recognition[2] (Korkontzelos and Manandhar, 2010) ; (2) it allows for the identification of semantic units as MWEs form idiomatic units (Baldwin and Nam, 2010).

[1]The utterance would be annotated in standard chunks like below:

[XN la marge] [XP d'exploitation]
(the trading margin)

[2]Korkontzelos and Manandhar (2010) showed that recognizing MWEs could improve chunking. They limited the experiment to some types of MWEs that do not change the chunk annotation definition.

Example 1 illustrates super-chunking.[3] The annotated sentence contains 4 MWEs (between brackets): the adverbial time expression *durant le premier trimestre 2007* (during the first trimester 2007), the nominal determiner *l'ensemble des* (the whole), the noun *chiffre d'affaires brut* (gross sales) and the complex numerical determiner *6 121 millions de*. It is composed of a sequence of 6 super-chunks (instead of 11 chunks with standard chunking).

(1) *[(Durant le premier trimestre 2007)], [(l'ensemble des) activités] [au Maroc] [ont généré] [un (chiffre d'affaires brut)] [de (6 121 millions de) dirhams]*
([During the first trimester 2007], [the whole activities] [in Marocco] [generated] [gross sales] [of 6,121 million dirhams].)

Several multiword expressions can be combined into a same super-chunk because any lexical item can be a multiword expression. For instance, let's consider the following annotated sequence:

[XN La température] [XP (à l'intérieur de) (beaucoup de) maisons] [XP en Moldavie]
(*[XN the temperature] [XP inside a lot of houses] [XP in Moldavia]*)

The whole phrase *à l'intérieur de beaucoup de maisons* is considered to be a prepositional super-chunk (XP) because *à l'intérieur de* (inside) is a multiword preposition, *beaucoup de* (a lot of) is a multiword determiner and *maisons* (houses), a simple noun.

Verbal chunks are also very specific because they can include auxiliaries in the sense of Gross (1999), inserts, clitics and negation. For example, the sentence

Jean n'a pas pu les trouver
(John could not find them)

is annotated:

[XN Jean] [XV n'a pas pu les trouver]
([John] [could not find them])

The discontinuous sequence *n'... pas* (not) is a negation, *a ... pu* is the preterit form of the modal verb *pouvoir* (to can) and *les* is an accusative clitic.

2.2. Target annotation

The annotation tagset of the super-chunker is given in Table 1. Lexical items that do not belong to any chunk like conjunctions, punctuations or relative pronouns are labelled with the tag O (meaning other). It is possible to have multiword O like for multiword conjunctions (e.g. *bien que* – although). In that case, the whole unit is annotated O: *bien_que/O*. Example 1 is then fully annotated as provided in Table 2.

The target evaluation corpus is a mix of a part of the novel "Le Tour du Monde en 80 jours" (Around the World in 80 days) by Jules Verne, as well as some reports of French

[3]This example shows the super-chunk segmentation without providing the labels.

TAG	DESCRIPTION
XA	adjectival chunk
XADV	adverbial chunk
XN	nominal chunk
XP	prepositional nominal chunk
XV	verbal chunk
XVP	prepositional verbal chunk

Table 1. Super-chunker tagset

CHUNK	TAG
Durant le premier trimestre 2007	XADV
,	O
l'ensemble des activités	XN
au Maroc	XP
ont généré	XV
un chiffre d'affaires brut	XN
de 6 121 millions de dirhams	XP

Table 2. Chunking result example

Parliament sessions. It is composed of 424 sentences, 8,319 tokens and 4,394 super-chunks. It has been semi-automatically annotated by T. Nakamura and S. Voyatzi at the LIGM laboratory. Both annotators have independently validated the whole corpus. In case of disagreement, a final decision was made after discussion. The super-chunk annotations exactly match the target annotations. Despite its small size, this corpus is therefore adequate for the evaluation of our tool. It is much harder to parse than a journalistic corpus because it contains specific terms (Parliament report section) and very long named entities (Around the World in 80 days). It is also full of dialogs. We assume that we have no development corpus of the same type. This means that the super-chunking process is completely blind.

3. Resources

In this section, we present the resources available to train and tune our super-chunker. They are composed of a treebank and lexical resources. The chunk annotations directly derived from the treebank do not exactly match our target ones.

3.1. Annotated corpus

The French Treebank (FTB) is a syntactically annotated corpus made of journalistic articles (Abeillé et al., 2003). Our edition comprises 584,987 tokens and 19,108 sentences. One benefit of this corpus is that compounds are marked. Their annotation was driven by linguistic criteria such as the ones in Gross (1986). Some organization and location names are also encoded, e.g. *Royal Zenith of Great Neck*, *San Francisco*. In total, around 5.6% of all lexical units are marked as multiword expressions. Some types of MWEs are missing like nominal collocations,[4] time expressions (date, duration, etc.), person names, etc.

We automatically converted the treebank to super-chunks in the target chunk tagset. In most cases, the chunk definition corresponds to the expected ones, except for verbal chunks when the verbs are combined with auxiliaries and modal verbs. For instance, the sentence *Marie peut changer* (Marie can change) is annotated *(XN Marie) (XV peut) (XV changer)* in the French Treebank, whereas the expected annotation is *(XN Marie) (XV peut changer)* because *peut* (can) is a modal verb.

We split the corpus in two distinct sections: a training section (90%) and a development section (10%). The training section was used to learn the CRF model, whereas the development section was used for the tuning of the features, the lexical and grammatical resources and the super-chunker POM.

3.2. Lexical resources

The lexical resources include large-scale dictionaries developed by linguists. They are lists of lexical entries, each of them being composed of an inflected form, a lemma, a part-of-speech (POS), morphological information (*e.g.* gender, number), syntactic information (*e.g.* transitive or intransitive verbs) and semantic information (*e.g.* human feature for nouns). They encode not only simple words but also multiword expressions like compounds. They are compressed in the form of FSTs in order to be efficiently applied to the text.

All dictionaries that we used are listed in Table 3. The larger ones were developed between the mid-80's and the mid-90's by linguists at the University of Paris 7: DELAF (Courtois, 1990) is composed of 746,198 inflected simple forms; DELACF (Courtois

[4] Collocations are combinations of words that co-occur more often than by chance. They are usually defined through statistical criteria.

et al., 1997) contains 255,163 inflected compounds (mostly composed of compound nouns). Compounds are of the following types :

- nouns: *pomme de terre* (potato), *faux témoignage* (perjury)
- prepositions: *au milieu de* (in the middle of), *à cause de* (because of)
- adverbs: *par ailleurs* (moreover), *en pratique* (in practice)
- conjunctions: *bien que* (although), *pendant que* (while)

The dictionary PROLEX (Piton et al., 1999) is a list of toponyms used in order to recognize location names. There exist several additional dictionaries containing organizations, first names, domain-specific terms and additional lexical entries found during the development process.

Name	Description	#entries	Reference
DELAF	Simple words	948,177	(Courtois, 1990)
DELACF	Compound words	255,163	(Courtois et al., 1997)
PROLEX	Toponyms	192,249	(Piton et al., 1999)
MISC	Additional dictionaries	34,151	

Table 3. Dictionaries

Our lexical resources also contain a library of strongly lexicalized local grammars. Local grammars (Gross, 1997) are Recursive Transition Networks (RTNs) (Woods, 1970) and theoretically recognize algebraic languages. They are of great interest for representing local lexical and syntactic constraints in a simple and compact way. We use them mostly to describe MWEs. They can define syntactic classes such as noun determiners and even syntactico-semantic classes such as time adverbials. Linguistic descriptions are in the form of Finite-State Graphs (Silberztein, 1994) on an alphabet made of terminal and non-terminal symbols. A terminal symbol is a word or a lexical mask. A lexical mask is an underspecified lexical entry (some features are missing) equivalent to a feature structure representing a set of lexical entries: *e.g.* the lexical mask *<noun+plural>* matches all nouns in the plural. Finally, a non-terminal symbol is a reference to another graph. A graph represents a transducer and its output is the annotation assigned to utterances described in the graph. An example of a local grammar is given in Figure 1.[5] This grammar describes time adverbials and recognizes structures like *en mars 2007* (in March 2007) and *cinq minutes plus tard* (five minutes later). The sequences recognized by this graph are tagged as time adverbs (`ADV+time`). Strings between < and > are lexical masks: for instance, *<minute>* stands for the inflected forms whose lemma is *minute*. Greyed vertices are call to other graphs. For ex-

[5] The local grammars are drawn using the graph editor of the Unitex platform (Paumier, 2003).

ample, Dnum and month are graphs that recognize numerical determiners and month names.

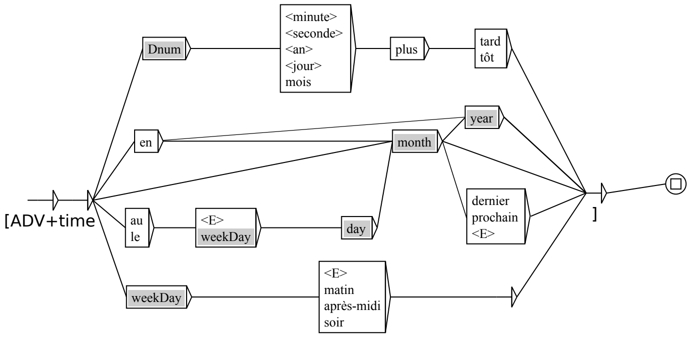

Figure 1. Local grammar of time adverbials

Practically, our lexical resources include a network of 302 graphs (in total, 2,853 states and 11,425 transitions in the RTN representation). Local grammars recognize sequences of the following types:

- nouns: function names [*ministre anglais de l'Agriculture* (English minister of Agriculture)], location names [*lac des Bois* (lake of the Woods)] and person names [*M. John Smith*]
- prepositions: locative prepositions [*à dix kilomètres au nord de* (ten kilometers north of)]
- determiners: numerical determiners [*vingt-sept* (twenty seven), *des milliers de* (thousands of)], noun determiners [*dix grammes de* (ten grams of)]
- adverbs: time adverbials [*en octobre 2006* (in October 2006)]

Local grammars identifying named entities (i.e. time adverbials and organization, location and person names) were partly constructed from Martineau et al. (2009). Graphs recognizing noun determiners come from Silberztein (2003); Laporte (2007). All of them are compiled into equivalent FSTs[6] (see Table 4).

4. Super-chunking with Conditional Random Fields

Chunking, and by extension super-chunking, can be seen as a sequential annotation task (Ramshaw and Marcus, 1995). Each word is assigned a tag Y-TAG: TAG is the tag of the chunk it belongs to and Y indicates its relative position in the chunk (*B* for beginning; *I* for the remaining positions). Table 5 shows an example of a chunked

[6]The local grammar representing determiners *det* is strictly recursive and there is no equivalent FST (just an approximation).

type	#graphs	#states in RTN	#transitions in RTN	#states in equiv. FST	#transitions in equiv. FST
adv	56	594	232	1,722	33,971
det	180	1718	746	/	/
noun	24	146	414	112	573
prep	43	386	1,215	2,599	62,657

Table 4. Compilation of local grammars into equivalent FSTs

text using this annotation scheme: column CHUNK corresponds to the output chunk label.

WORD	MWE+POS	CHUNK
l'	B-DET	B-XN
ensemble	I-DET	I-XN
des	I-DET	I-XN
activités	B-N	I-XN
ont	B-V	B-XV
généré	B-V	I-XV
121	B-DET	B-XN
millions	I-DET	I-XN
de	I-DET	I-XN
dirhams	B-N	I-XN

Table 5. Chunking result example

Several studies (Sha and Pereira, 2003; Tsuruoka et al., 2009) have shown that Linear-chain Conditional Random Fields (LCRF) are very competitive for chunking. The models usually incorporate features computed from predicted POS and the words themselves like in Tsuruoka et al. (2009). Nevertheless, standard chunkers do not account for MWE recognition. In Constant and Tellier (2012), we showed that MWE recognition could be successfully performed jointly with POS tagging. The column MWE+POS in Table 5 corresponds to the output of such joint task. A first super-chunking strategy is therefore to use such MWE-aware labels instead of simple POS. The model is trained on the training corpus *as is*, although chunk annotations do not entirely match our target ones. We use a similar set of feature templates as Tsuruoka et al. (2009) for their base chunker, except that we deal with MWE-aware POS labels instead of simple ones: word unigrams, bigrams and trigrams; MWE+POS unigrams, bigrams and trigrams. We trained[7] two MWE-aware POS tagger models using features defined in Constant and Tellier (2012): the first model (called WITH) contains all features including features based on data available in the external lexical resources (namely lexicon-based features); the second one (called WITHOUT) contains all features but the lexicon-based ones. In order to select the best model for chunking, we applied the two models on the FTB development corpus. Results are provided in Table 6. The upper part corresponds to overall scores: MWE+POS is the joint MWE recognition and POS tagging accuracy in terms of labeled F_1-measure; U_1 stands for the chunking unlabeled F_1-measure indicating the super-chunking segmentation accuracy; F_1 stands for the chunking labeled F_1-measure. The lower part details the F_1-measure for each chunk label. The column *#Chunks* indicates the number of chunks for each label in the development corpus.

	#Chunks	WITHOUT	WITH
MWE+POS	–	93.9	**94.5**
U_1	–	**92.1**	91.2
F_1	–	**90.1**	88.8
O	10,827	97.4	97.0
XA	2,506	81.3	79.3
XADV	1,854	81.4	77.0
XN	7,161	86.6	85.6
XP	8,397	86.9	85.2
XV	5,377	91.5	90.7
XVP	788	92.3	90.7

Table 6. Baseline results on the FTB development section

The best super-chunker reaches around 90% accuracy on the development corpus. We observe that chunk segmentation costs around 8 points[8]. We consider it our *baseline*. Surprisingly, the best super-chunker uses a MWE-aware POS tagger including no lexicon-based features, whereas joint MWE and POS labeling is much better with lexicon-based features. This might show that errors caused by the lexicon-based tagger are critical and cause much more damages for super-chunking.

[7] We trained the CRF models by using the software Wapiti (Lavergne et al., 2010), with the same settings as in (Constant and Tellier, 2012).

[8] The chunk segmentation cost is $100 - U_1$.

5. Super-chunking with a finite-state lexicon-driven approach

Blanc et al. (2007) proposed a finite-state architecture to handle super-chunking, that was developed in the tool POM. It is based on a cascade of finite-state transducers (FSTs), similarly to the historical finite-state approach of shallow parsing (Joshi and Hopely, 1997; Abney, 1996; Ait-Mokhtar and Chanod, 1997). It relies on external large-coverage lexical resources, as in Silberztein (1994). The chunker is composed of three successive stages as illustrated in the diagram in Figure 2: (1) an enhanced ambiguous lexical analysis, (2) an ambiguous chunk analysis, (3) a chunk disambiguation module. The whole system is mainly driven by linguistic resources in the form of lexicons and local grammars. There might also be preprocessing and post-processing stages, for instance, to deal with disfluencies in speech transcripts (Blanc et al., 2010). In this paper, we used the same super-chunker architecture.

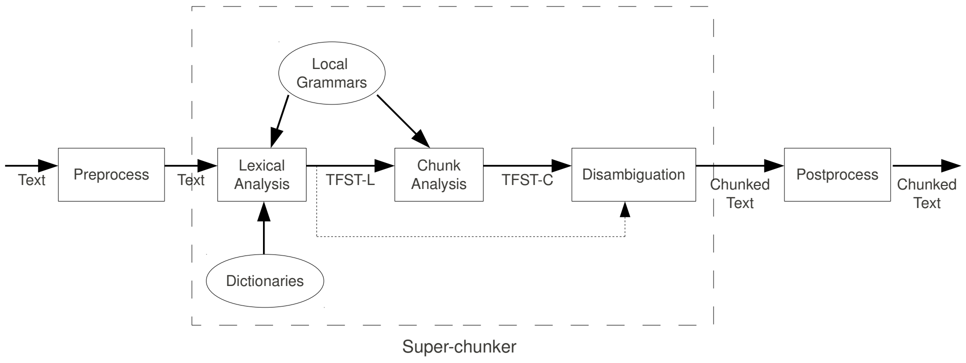

Figure 2. Process diagram

The lexical analysis module takes as input a text segmented into sentences and tokens. It uses large-coverage lexical resources in the form of morphosyntactic dictionaries and lexicalized local grammars, all compiled into FSTs (cf. Section 3). These resources are applied iteratively to the text. The module generates an acyclic text finite-state transducer (TFST-L) representing lexical ambiguities for simple words and multiword expressions for each sentence of the input text. First, a dictionary lookup associates each token with all its possible morphosyntactic tags and recognizes MWEs. The output of the lookup is a finite state transducer (TFST). Then, a cascade of strongly lexicalized grammars is iteratively applied to the TFST, which is augmented with the analyses of the matching MWEs.

Chunk analysis is also based on a cascade of finite transducers applied to TFST-L, which is augmented each time a new chunk is found. It returns the text finite transducer TFST-C. The module is composed of 13 successive stages, each stage corresponding to the application of a given FST recognizing a type of syntactic structure. We successively identify :

1. adverbials (`XADV`): simple adverbs or multiword adverbials that have been recognized during the lexical analysis
2. adjectival chunks (`XA`): adjectives that can be preceded by an adverb
3. nominal chunks (`XN`): simple noun phrases, named entities, some types of pronouns
4. prepositional chunks (`XP`): `XN` preceded by a preposition
5. verbal chunks (cascade of 9 FSTs): passive and active forms of infinitive, past participle, gerund and simple verbal chunks, complex structures integrating auxiliaries in the sense of Gross (1999).

The syntactic patterns were constructed manually in the form of local grammars constituting a global network of 115 graphs (1,109 states and 6,054 transitions in the RTN representation). Then, they were compiled into equivalent FSTs which comprise 823 states and 10,703 transitions in total. Our FSTs represent not only purely syntactic patterns (e.g. `XN` can be composed of a noun preceded by a determiner) but also lexico-syntactic patterns. For example, we used the lexico-syntactic patterns to:

- describe auxiliaries in the sense of Gross (1999) followed by a verb in the infinitive, such as *viser à* (to aim at), *avoir peur de* (to be afraid of)
- describe fixed `XN`s such as *les uns et les autres* (one and every)
- describe intensive adverbials that can modify adjectives such as *très* (very), *un peu* (a little).

The generated TFST-C is composed of both POS and chunk tags found in the lexical and chunk analyses. In order to remove ambiguity, the chunker includes an incremental disambiguation module removing paths until total linearization is reached. Blanc et al. (2007) proposed the three following stages: disambiguation with handcrafted rules (given an ambiguity and a context, selection of a tag by removing the transitions corresponding to the other tags); algorithm keeping the shortest paths (in order to favor multiword analyses); then a simple statistical linearization (based on the probability to associate the tag of the chunk with its head word).

In this paper, we developed a simpler disambiguation stage. It was limited to one module: the application of the shortest path algorithm on TFST-C, which was shown to be the best. The TFST-C weighting is manually set, giving priority to chunk tags. This lighter disambiguation module forced us to use a specific simple word dictionary with very few ambiguities. To do so, we applied a standard POS tagger[9] on simple words at the preprocessing stage. From it, we built a dictionary of simple words that we applied at the lexical stage. We also applied all MWE lexical resources described above. The standard POS tagger model integrates features based on all our lexical resources (cf. Section 3).

[9] We used *lgtagger* (Constant and Tellier, 2012).

6. Combining CRF-based super-chunker and lexicon-driven super-chunker

Our intuition is that POM super-chunker is more accurate for segmentation than the CRF-based one (our *baseline*) because it is driven by large-coverage lexical resources full of MWEs, all compatible with our target annotation. Furthermore, the CRF-based super-chunker should be more accurate to label the identified chunk segments, as POM does not have any advanced disambiguation tools. Therefore, it sounds interesting to combine both systems.

For this purpose, we developed two different combination strategies. The first one consists in merging the outputs of both super-chunkers. It is called *merge*. The second one consists in adapting the training corpus by merging it with the POM output, and then in learning a new CRF model. It is called *adapt*. The two strategies are based on the same merging procedure. This procedure takes two annotations as input and generates a new annotation, as illustrated in Table 7. It works as follows. We first perform super-chunk segmentation by gathering the two input annotations in the same directed graph. Each node corresponds to a chunk segment at a given position. An arc links two chunks c_1 and c_2 if the end of c_1 coincides with the beginning of c_2 in the text. We find the final segmentation by applying a shortest path algorithm that favors the longest chunks, and, in case of segmentation ambiguity, the chunks found by POM. The graph computed for our example is provided in Figure 3. Once the chunk segmentation is selected, we assign to each chunk its label found in the annotations. In case of ambiguity, the CRF-based label (or the FTB reference one) is chosen, except for some specific cases like adverbials. This algorithm is very easy to implement and formulates our initial intuition. Moreover, it could be easily extended from 2 to n annotations.

POM output		FTB Reference annotation		Merge	
Le 10 juillet 'On July 10'	XADV	Le 10 juillet 'On July 10'	XN	Le 10 juillet 'On July 10'	XADV
Luc Ferry 'Luc Ferry'	XN	Luc Ferry 'Luc Ferry'	XN	Luc Ferry 'Luc Ferry'	XN
put rencontrer 'could meet'	XV	put 'could'	XV	put rencontrer 'could meet'	XV
		rencontrer 'meet'	XV		
le ministre 'the minister'	XN	le ministre des affaires sociales 'the minister for social affairs'	XN	le ministre des affaires sociales 'the minister for social affairs'	XN
des affaires sociales 'for social affairs'	XP				

Table 7. Example of two annotations for the merging procedure

For the method *adapt*, the CRF model might incorporate additional features (as compared with the *baseline* model). These features are based on the annotations generated by POM. Given a position i, let $\mathrm{chk}(i)$ be the POM-predicted tag of the chunk

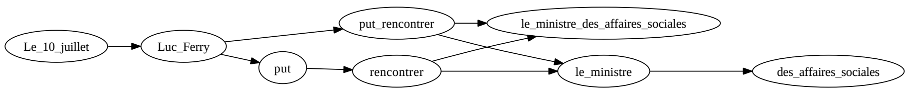

Figure 3. Example of graph for the merging procedure

the current token belongs to. chkbi(i) indicates the relative position of the token in the POM-predicted chunk: B for the starting position and I for the remaining positions. c(i) is the current output label (a chunk tag). The additional feature templates are provided in the Table 8. We tested this strategy both with the baseline features (*adapt-base*) and with all features including the additional ones described in Table 8 (*adapt-advanced*).

$\mathrm{chk}(i+j)/\mathrm{chkbi}(i+j)/c(i)$	with $j \in [-2, 2]$
$\mathrm{chkbi}(i+j)/c(i)$	with $j \in [-2, 2]$
$\mathrm{chk}(i+j)/c(i)$	with $j \in [-2, 2]$

Table 8. Additional feature templates

7. Evaluation

This section is devoted to the evaluation of the three proposed solutions on the target MIX corpus. We compare them with the two simple super-chunkers presented in Sections 4 and 5. The results are provided in Table 9.

We can first notice that the performances of the baseline CRF-based super-chunker and, respectively, our version of POM are quite low (76%) as compared with the scores obtained on the FTB-dev (90%) and, respectively, the scores obtained on journalistic articles by the POM version described in Blanc et al. (2007) (around 95%). The difference with the results on FTB-dev can be partly explained by the fact that some types of multiword expressions are not marked in the FTB-train and some verbal super-chunks do not match with what is expected (cf. low accuracy for XV and XVP). Moreover, in Blanc et al. (2007), a great effort was made on the tuning of POM resources and disambiguation rules. We developed them with the help of a corpus of the same type as the evaluation corpus. In our case, there are no means to tune our resources. In addition, there are some disambiguation modules missing, even though this is partly

	# chunks	baseline	POM	merge	adapt base	adapt advanced
U_1	–	80.5	84.7	**87.9**	85.9	**88.2**
F_1	–	76.0	76.1	83.2	81.3	**83.6**
O	1,591	95.4	86.8	95.5	95.5	**95.8**
XA	157	58.5	60.0	66.5	62.2	**68.6**
XADV	343	49.3	48.8	59.5	56.4	**59.7**
XN	832	76.3	74.2	81.8	78.8	**82.0**
XP	591	62.6	69.8	72.2	66.4	**72.8**
XV	794	65.0	82.2	82.4	82.2	**82.8**
XVP	86	72.4	80.2	**80.5**	76.1	77.6

Table 9. Final results on the evaluation corpus MIX

compensated by the use of a POS tagger to have less ambiguous lexical resources. Nevertheless, the main cause of this performance drop is simply that the MIX corpus is hard to parse.

Furthermore, we can observe that the baseline CRF-based super-chunker and our version of POM have very comparable results. They have comparable accuracies on adjective, adverbial and noun phrases. But they show very different results on the other categories. POM is very bad for identifying non-chunks, i.e. tag O: around 9 point difference with the CRF-based. This is mainly due to the light disambiguation module of POM. The CRF-based super-chunker obtains bad results for prepositional phrases (-7 point difference with POM) and verbal chunks (at worse, -17 points). This is not surprising for verbal chunks as the training corpus is annotated with a verbal super-chunk definition different from the evaluation corpus. For the prepositional phrases, this is due to incorrect multiword preposition recognition.

We can notice that our assumptions on the performances of the baseline and POM are verified: POM is better for segmentation (84.7% vs. 80.5%), whereas the baseline is better at disambiguation (-3.5 vs. -8.6 points as compared with the segmentation scores). As expected, the combination strategies show great improvements. The strategy *adapt-advanced* reaches the best accuracy with a very substantial gain of +7.6 points as compared with the baseline. The *merge* method obtains slightly lower scores, but they are comparable. We can see that *adapt-base* has lower improvement: +5.5 points. This shows that the use of features based on the super-chunks predicted by POM are of great interest (gain of around 2 points). For non-chunks, verbal and prepositional phrases, the combined super-chunker reaches results comparable with the ones obtained by the best simple chunker for these categories. For adjective, adverbial and noun phrases, we observe that the two super-chunkers are complementary as their combination achieves quite better scores as compared with the best simple chunker for each of these categories: +10 points for adverbials, +9 points for adjective phrases, +6 points for noun phrases.

8. Conclusions and Future Work

In this paper, we focused on shallow parsing, more precisely on chunking including MWE recognition, namely *super-chunking*. As it is sometimes hard to have a training corpus with the exact same annotations as expected, it can be easier to use a chunker driven by lexical resources that are usually adapted to one's needs. We have shown how to improve a CRF-based super-chunker by coupling it with a finite-state symbolic lexicon-driven super-chunker. We used a procedure merging two chunk annotations: either to merge the two super-chunker outputs, or to adapt the training corpus with the lexicon-driven super-chunker. In the second case, the CRF model is even improved when integrating additional features based on the lexicon-driven super-chunker. We display a substantial gain of 7.6 points in terms of general accuracy for this latter solution. Future work might consist in improving the lexicon-driven super-chunker in order to have a more precise merging procedure and have finer features for CRF models. It would be interesting to extend the evaluation to other target domains.

Bibliography

Abeillé, A., L. Clément, and F. Toussenel. Building a treebank for French. In Abeillé, A., editor, *Treebanks*. Kluwer, Dordrecht, 2003.

Abney, S. P. Parsing by chunks. In Berwick, Robert C., Steven P. Abney, and Carol Tenny, editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 257–278. Kluwer Academic Publishers, Dordrecht, 1991.

Abney, S. P. Partial parsing via finite-state cascades. *Natural Language Engineering*, 2(4):337–344, 1996.

Ait-Mokhtar, S. and J.-P. Chanod. Incremental finite-state parsing. In *Proceedings of the fifth Conference on Applied Natural Language Processing (ANLP'97)*, 1997.

Arun, A. and F. Keller. Lexicalization in crosslinguistic probabilistic parsing: The case of french. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL'05)*, 2005.

Baldwin, T. and K.S. Nam. Multiword expressions. In *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group, 2010.

Blanc, O., M. Constant, and P. Watrin. Segmentation in super-chunks with a finite-state approach. In *Proceedings of the Workshop on Finite-State Methods for Natural Language Processing (FSMNLP'07)*, 2007.

Blanc, O., M. Constant, A. Dister, and P. Watrin. Partial parsing of spontaneous spoken French. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10)*, 2010.

Cafferkey, C., D. Hogan, and J. van Genabith. Multi-word units in treebank-based probabilistic parsing and generation. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP'07)*, 2007.

Constant, M. and I. Tellier. Evaluating the impact of external lexical resources into a crf-based multiword segmenter and part-of-speech tagger. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, 2012.

Constant, M., A. Sigogne, and P. Watrin. Discriminative strategies to integrate multiword expression recognition and parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL'12)*, pages 204–212, 2012.

Courtois, B. Un système de dictionnaires électroniques pour les mots simples du français. *Langue Française*, 87:11–22, 1990.

Courtois, B., M. Garrigues, G. Gross, M. Gross, R. Jung, M. Mathieu-Colas, A. Monceaux, A. Poncet-Montange, M. Silberztein, and R. Vivés. Dictionnaire électronique DELAC : les mots composés binaires. Technical Report 56, LADL, University Paris 7, 1997.

Green, S., M.-C. de Marneffe, J. Bauer, and C. D. Manning. Multiword expression identification with tree substitution grammars: A parsing tour de force with French. In *Proceedings of the conference on Empirical Method for Natural Language Processing (EMNLP'11)*, 2011.

Gross, M. Lexicon grammar. the representation of compound words. In *Proceedings of the conference on Computational Linguistics (COLING'86)*, 1986.

Gross, M. The construction of local grammars. In Roche, E. and Y. Schabes, editors, *Finite-State Language Processing*, pages 329–352. The MIT Press, Cambridge, Mass., 1997.

Gross, M. Lemmatization of compound tenses in english. *Lingvisticae Investigationes*, 22, 1999.

Joshi, A. and P. Hopely. A parser from antiquity. *Natural Language Engineering*, 2(4), 1997.

Korkontzelos, I. and S. Manandhar. Can recognising multiword expressions improve shallow parsing ? In *Proceedings of the Conference on Human Language Technologies and the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'10)*, pages 636–644, 2010.

Kudo, T. and Y. Matsumoto. Chunking with support vector machines. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'01)*, 2001.

Laporte, E. Extension of a grammar of French determiners. In *Proceedings of the international conference on Lexicon and Grammar (LGC'07)*, pages 65 – 72, 2007.

Lavergne, T., O. Cappé, and F. Yvon. Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10)*, pages 504–513, 2010.

Martineau, C., T. Nakamura, L. Varga, and S. Voyatzi. Annotation et normalisation des entités nommées. *Arena Romanistica*, 4:234–243, 2009.

Nasr, A. and A. Volanschi. Integrating a POS tagger and a chunker implemented as weighted finite state machines. In *Proceedings of the Workshop on Finite-State Methods and Natural Language Processing (FSMNLP'05)*, pages 167 – 178, 2005.

Nivre, J. and J. Nilsson. Multiword units in syntactic parsing. In *Proceedings of Methodologies and Evaluation of Multiword Units in Real-World Applications (MEMURA)*, 2004.

Paumier, Sébastien. *De la reconnaissance de formes linguistiques à l'analyse syntaxique*. PhD thesis, Université Paris-Est Marne-la-Vallée, 2003.

Piton, O., D. Maurel, and C. Belleil. The Prolex data base : Toponyms and gentiles for nlp. In *Proceedings of the Third International Workshop on Applications of Natural Language to Data Bases (NLDB'99)*, pages 233–237, 1999.

Ramshaw, L. A. and M. P. Marcus. Text chunking using transformation-based learning. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 88 – 94, 1995.

Sag, I. A., T. Baldwin, F. Bond, A. Copestake, and D. Flickinger. Multiword expressions: A pain in the neck for nlp. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing (CICLing '02)*, pages 1–15, London, UK, 2002. Springer-Verlag.

Sha, F. and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the Conference on Human Language Technologies and the Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'03)*, pages 213 – 220, 2003.

Silberztein, M. Intex: a corpus processing system. In *Proceedings of the conference on Computational Linguistics (COLING'94)*, 1994.

Silberztein, M. Finite-state description of the French determiner system. *Journal of French Language Studies*, 13(2), 2003.

Tsuruoka, Y., J. Tsujii, and S. Ananiadou. Fast full parsing by linear-chain conditional random fields. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL'09)*, pages 790–798, 2009.

Woods, W.A. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10), 1970.

Address for correspondence:
Matthieu Constant
`mconstan@univ-mlv.fr`
Cité Descartes
5, boulevard Descartes
Champs-sur-Marne
77454 MARNE-LA-VALLEE Cedex 2
France

The Prague Bulletin of Mathematical Linguistics
NUMBER 99 APRIL 2013 77–86

Significance of the Correlation in Turn-taking Dialogues

Lourembam Surjit Singh[a], R.K. Brojen Singh[b]

[a] Centre for Advanced Studies in Linguistics, University of Delhi, New Delhi-110007, India.
[b] Centre for Interdisciplinary Research in Basic Sciences, Jamia Millia Islamia, New Delhi 110025, India.

Abstract

This piece of work investigates the relation between characters in a play based on turn-taking dialogues. As the dialogues are related to each other in the play, numbers of turn-taking are significant of the characters' relationship and the utterances give indication about the information content of the interaction. In this Manipuri radio play, the sequence of turns taken by the characters convey different amount of information with different functions. Numbers of dialogues oscillate significantly with a period of 2 scenes during the play. The degree of oscillation present in turn-taking dialogues carries significant information functions. The nature of the relationship between the characters involved and the theme of the play may be characterised by calculations on turn-taking dialogues.

1. Introduction

Conversational Analysis (CA) is a systematic study of the talk produced in everyday situations of human interaction which naturally occur in talk-in-interaction (Hutchby and Wooffitt, 1998). Under its scope falls also talk-in-interaction whose objective is to discover how participants of the conversation understand and respond to one another in their interaction, the focus being on how sequence of actions are generated. CA is in fact a methodology used by sociologists, anthropologists, and linguists who study group culture, people's conversational norms or people's style of life. In this regard, Moerman (1988) proposes conversation is a central part of communication that makes people understand the social order. However, according to founder philosophers like Sacks and Jefferson (1974); Schegloff (1992), an ideal model of conversation is focused on the exchange of turns, turn management and turn order along with the patterns of sequence of turns such as minimal gap, conversational

Corresponding author: lsurjit24@yahoo.com
Cite as: Lourembam Surjit Singh, R.K. Brojen Singh. Significance of the Correlation in Turn-taking Dialogues. The Prague Bulletin of Mathematical Linguistics No. 99, 2013, pp. 77–86. doi: 10.2478/pralin-2013-0005.

pairs, overlaps, etc. The principle of CA can be applied specifically to spoken dialogues within dramatic conversations as general interaction. In the study of general interaction, the order of conversational sequences, pairs of utterances (also known as adjacency pairs (Sacks, 1992)) and preference of pairs are organized according to the pattern of turns or turn-taking management. The speaker manages turns when he/she speaks, he/she takes a turn at dialogic speech, and as speech alternates, turns alternate as well. That's how various numbers of turns are distributed in order to fulfill the participant's rights to speak and take turns. For this, characters who are the speaker in the play managed the turns to mitigate the threats of speech chaos when numerous interactions take place in the play.

It is a phenomenon that the order of dialogic speech in dramatic text is generally organized to reflect the order of turns to be taken by the characters in the play. Then turns are ordered into sequences to reflect the necessary dialogues. A dialogue in a play is always well constructed, and also has finite properties of meaning. The information conveyed by dialogic speech in a conversation is a goal-oriented communication. We may assume a framework where each utterance in a turn has one specific goal in the communication. This communicative goal intends to share some persuasive information among the characters and represent it to the audience with a specific value of information. In the dialogic communication, when an audience listens to a dialogue, he/she only tries to grab the shared information either implicitly or explicitly, then he/she understood either some part of the intended meaning or the whole information in the conversation. In this case, the information transferred from a turn or dialogue which is produced by a specific character is numerically accounted as one unit of information and can be calculated with some statistic assumptions. This specific unit of information can be derived either from a general or a long turn, even though the turns had different propositions. The quantitative value of one turn should be mapped onto one numeric value, and then alternate for other turns, so that all the numbers of turns taken by the characters can be marked with a specific quantitative value (i.e. as one value) in order to count whole numerical values in the play. Once this mapping is established for every turn, the numbers of turns which have different information functions produced by distinct characters in the play can be separately evaluated for each of the characters. Based on this different numeric value for different characters, we can distinguish the level of importance that a character has in the story of the play. The result will be beneficial when we show that the play have protagonists who took a maximum number of turns, supporting characters who took a smaller number of turns as well.

We organize our work as follows. In Section 2, the theory and methods which we used to analyze turn-taking, turn management and talk-in-interaction in the proposed play, are discussed. The detailed results and discussion are presented in Section 3. Some conclusions are drawn based on the results that we analyzed.

2. Theory and Methods

Dialogue is generally made by the tidied up speech and it is organized according to the plot of the play. The study of such dialogues of a voice play as general interaction of a conversation is a complex matter which involves various backgrounds of analysis. For extracting some information from such dramatic dialogues is also involved a complete study of discourse. However, focus on the study of the sequence of utterances and the organization of such sequences, the target of conversational interaction is turned into the concept of turn-taking in the play. Study of turns and turn-taking in the play is somewhat adequate with the study of interaction in a communication. This background will support in rating the degree of dramatic information. However, calculating an accurate information which is contained in a turn or dialogue is also a bit complex in the context of human interaction.

A speaker has right to speak by taking an opportunity to speak in the speech event or situation (Herman, 1998b,a). When a speaker speaks, he or she takes turn at speech and as speech alternates turn alternates as well. In these numbers of turn alternates, the turn allocational component regulates the changeover of turns such as one participant talks stops, the next participant talks, stops, and so on. Thus, speakers always arranged the sequences for the 'next' turns to maintain the required understanding of what the 'prior' was or will about. This structure of the arrangement of turns is known as next-turn proof procedure in CA. The conflict that problematized by CA at the changeover point of the turn is termed as transition-relevance place (Sacks, 1992; Levinson, 1983). In such theory of CA, to understand these whole conversational patterns of turn-taking, the conversational sequences are introduced as adjacency pair particularly in turn-taking and proof-procedure analysis. An adjacency is the adjacent part of one pair (half of one) of conversation that ordered the pairs of utterances in the sense that the first and the second pairs that produced by different speakers. The first pair required the second but not all the parts of sequences. In such situation the first part of the turn counts only the second which is responded in the pair. Thus, the concept of conversational pairs was used and developed in dialogues (Linell, 1998; Schiffrin, 1994) into two categories i.e. initiative and response. He classified most of the utterances under initiative and response. This phenomenon of classification is to be considered as true because the principle of double contextuality is caused into the utterances of a conversation.

The dialogues of this play have some textual contexts in talk-in-interaction such as requests and acceptance, greetings and proposals, questions and answers, accusations and complaint, statements and reports, etc. In such distinct textual propositions of conversation, the initiator and responder interchanged their required information according to the situation that's the plot demanded in the play. They also initiate the turns as per the turns that specifically allotted to them. In this distinct rate of initiation and response of turns taken by the characters in a play have various information functions. According to this different rate of the changeover of turns and turn-takings, the value and the role of a character is marked in the play. It illustrates that as per the rate of turn allocation the role of a character is considered to be as protagonist or supportive in the play. Generally, protagonist has more numbers of turns other than supportive characters has, so the amount of information produced by protagonist is larger than that of supportive characters. For calculating the whole data in this study these variable amount of information can be counted quantitatively or qualitatively from the numbers of the changeover of turns. For instance, the dialogues of female protagonist which is produced in the mode of solo turn such as the dialogue of individual secret love which involve expressions, propose, decline, acceptance, etc. may reflect coordinating behaviour between the characters. This total numbers of the changeover of turns can be measured as one value specifically as per one turn. Based on this phenomenon, while we investigate the play, the data of R.K. and Phajabi shows oscillation roughly in a period of two scenes. These particular data of two scenes which is started from scene one to two have variable in values and then the other data from scene four to scene eight shows the behaviour of correlation in the play. Then the plot starts diverting from scene nine onwards until end of the play. It means the maximum scenes of the play are correlated. The exhibition of these true functions is shown in Figure 1.

2.1. Correlation function technique

If x and y are the variables indicating the number of turn-taking dialogues of R.K. and Phajabi respectively such that $\{x, y : \mathbb{R}\}$. The two characters are paired via the pairs of turn-taking dialogues which take place between them, and both the characters understand to each other in the interaction and act according to it. Therefore the rate of correlation behaviour can be calculated numerically from the following cross correlation function $c_{xy}(\tau)$ (Quiroga et al., 2002),

$$c_{xy}(\tau) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{x_i - \langle x\rangle}{\sigma_x}\right]\left[\frac{y_{i+\tau} - \langle y\rangle}{\sigma_y}\right]. \tag{1}$$

From here, one deduce the following autocorrelation function of the two variables x and y respectively,

$$c_{xx} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{x_i - \langle x\rangle}{\sigma_x}\right]^2, \quad c_{yy} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \langle y\rangle}{\sigma_y}\right]^2 \tag{2}$$

where $\langle x\rangle = \frac{1}{N}\sum_{j=1}^{N} x_j$ and $\langle y\rangle = \frac{1}{N}\sum_{j=1}^{N} y_j$ are the means of the data of x and y variables respectively. $\sigma_x^2 = \frac{1}{N}\sum_{k=1}^{N}[x_k - \langle x\rangle]^2$ and $\sigma_y^2 = \frac{1}{N}\sum_{k=1}^{N}[y_k - \langle y\rangle]^2$ are variances of the variables x and y respectively. N is the total number of data points in

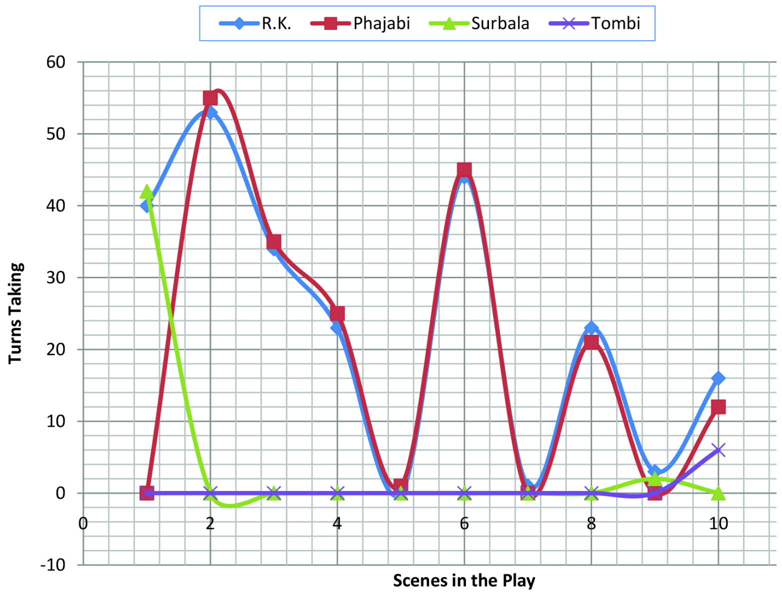

Figure 1. The play, 'Nongaallabasu Thaballei Manam' (Lingering Fragrance), a script of Moirangthem Inao, is represented in the form of a graph. The number of dialogues of the individual characters, namely R.K., Phajabi, Surbala and Tombi, are represented by the four lines in the graph. Scenes four to eight (specifically a period of two scenes) show the oscillation behaviours among the protagonists.

x and y. τ is the time lag. The value of c_{xy} are in between 0 and 1. If the value of c_{xy} is zero then the two variables are uncorrelated, i.e. the information carried by the two variables are in fact not related at all, however, if the value is 1, then the two variables are exactly correlated.

2.2. Complexity measurement technique

The study of the process of human interaction is a complex phenomena but the information embedded in such interaction can be measured through the process of permutation entropy technique (Bandt and Pompe, 2002). Permutation entropy, derived from entropy known as a measure of randomness in general, can be used as a technique to measure complexity or information contained in a symbolic sequence of data. The calculation is done as follows considering the data that is given by a symbolic sequence, $\mathbf{x}(\mathbf{t}) = \{x_1, x_2, \ldots, x_N\}$, where N is the size of data. This sequence is then partitioned into M number of short sequences, $\mathbf{x}(\mathbf{t}) = \{q_1, q_2, \ldots, q_M\}$

of size L, $q_i = \{x_1, x_2, \dots, x_L\}$ by sliding a window of size L with maximum overlapping (any two consecutive short sequences overlap $(L - 1)$ symbols in our calculation). The information contained in the case of any such short sequences, q_i can be measured by defining sequence of embedded dimension s, mapping onto q_i sequence and calculating Shannon entropy, H of it (Bandt and Pompe, 2002; Cao et al., 2004). This can be done by generating all possible inequalities of sequence of length s and calculating the probabilities of occurring each inequalities $p_j : j = 1, 2, \dots, u$ in q_i sequence so that one is allowed to calculate permutation entropy, $H_i(s) = -\frac{1}{\ln(s!)} \sum_{j=1}^{u} p_j \ln(p_j)$, where u is the number of distinct possibilities out of $s!$ permutations and $0 \leq H_i(s) \leq 1$. It is also to be noted that as $H(s)$ increases complexity is also increased according to it. Then one can map the sequence onto permutation entropy spectrum: $\mathbf{x}(\mathbf{t}) = \{H_1, H_2, \dots, H_M\}$.

The data used in the study are usually small in size and partition of sequences to be made even in smaller, we take all possible inequalities of neighbouring, next to nearest neighbouring in ascending order to obtain each H_i of $\mathbf{x}(\mathbf{t}) = \{H_1, H_2, \dots, H_M\}$. This permutation entropy spectrum may not highlight different emotions of individuals during interaction but for a certain specified emotion such as romantic, annoying, aggressive, etc. that feel between the characters, one can estimate qualitatively the amount of information of such emotion contained in the interaction by this technique.

3. Results and Discussion

The one-act radio play named *Nongallabasu Thaballei Manam* (The Lingering Fragrance) of Moirangthem Inao (Inao, 1995), which was broadcasted at All India Radio, Imphal on 2nd April, 1995, is a heart rendering and classic romantic play. We investigate this play within the framework of CA in terms of turn-taking, talk-in-interaction and adjacency pairs. In total over the whole play, R.K. takes 237 turns, Surbala takes 44 turns, Phajabi takes 193 turns and Tombi takes 6 turns. The total number of turns taken in this play is 480 turns in about one hour. The average number of words per turn suggests that there are unequal numbers of dialogues; there are very significant differences in turn-taking between the male protagonist and the female protagonist and also numerous differences in the number of turns between the two supporting characters. The notions of turn-taking and overlapping underlie the concept of sequences in the dialogues. However, the romantic dialogues that were uttered in the interaction between the male protagonist and the female protagonist are nearly equal in number and correlate to each other. These pairs of dialogues make the play complete up to the end and serve as the backbone for the plot of the play. To give an example of various pairs of turns, some dialogues are emotional, argumentative, interactional, narratives, and use figures of speech such as when one is to show his neglectance and the other one is showing her intensions with full expression of love and hatred.

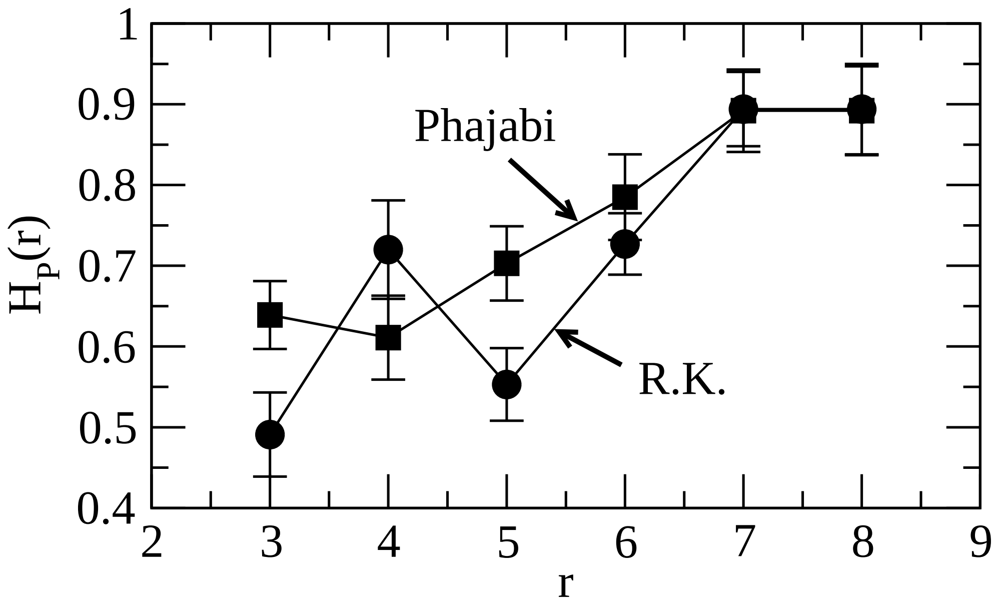

Figure 2. Measurement of information: the plots of permutation entropy HP(r) *as a function of scene number* r *of two main protagonists, R.K. and Phajabi.*

The data we collected for the study describe numbers of turns taken by protagonists in the play, namely R.K. (male protagonist) and Phajabi (female protagonist), and the supporting characters of the play, namely Surbala (wife of R.K.) and Tombi (husband of Phajabi), as shown in Table 1. This data is used as our primary data for analysing information embedded in each character's utterances. It is mainly used for inferring the number of pairs of turns which capture interaction among the characters. The data for each character is used in the plot as a function of scenes in the play, as shown in Figure 1. In the different stages of the plot, the data of R.K. and Phajabi particularly showed oscillation behaviour indicating association of information in the utterances. For this, the phenomenon of persuasive communication that is being in the interactions among the characters can be fully analyzed through correlation, given in Equation (1). Through this analysis, the content of correlated information may be inferred directly from the data or indirectly from the play.

We consider the data of R.K. and Phajabi individually and represent them by variables x and y respectively. Using the above equations, the values of the means and variances of the two variables are obtained as $\langle x\rangle = 23.7$, $\langle y\rangle = 19.3$, $\sigma_x = 17.91$ and $\sigma_y = 19.29$ respectively. Then using Equation (1), we calculated c_{xy} and found to be

Scene: Timing	Flash Back (FB) and Live Scenes	Number of turns taken by Tombi	R.K.	Surbala	Phajabi
1: 0:00:01–0:11:03	Live Scene	0	40	42	0
2: 0:11:08–0:22:11	1st FB start	0	53	0	55
3: 0:22:18–0:27:10	1st FB cont.	0	34	0	35
4: 0:27:17–0:31:52	1st FB cont.	0	23	0	25
5: 0:32:39–0:33:58	1st FB cont.	0	0	0	1 long
6: 0:34:20–0:46:35	1st FB end	0	44	0	45
7: 0:46:40–0:47:11	Live Scene cont.	0	1 long	0	0
8: 0:47:19–0:54:17	2nd FB	0	23	0	21
9: 0:54:22–0:55:01	3rd FB	0	3	2	0
10: 0:55:09–1:00:03	Live Scene	6	5+11	0	12
Total number of turns	480 100%	6 1.25%	237 49.37%	44 9.17%	193 40.28%

Table 1. Turns taken by characters in the play in different scenes represent patterns of the play in the form of a graph, given in Figure 1. This table shows number of turns taken by each character, including main protagonists, in the play scene by scene.

$c_{xy} = 0.97438$ which is very close to 1. This is meant that the information carried by R.K. and Phajabi are very closely related to each other and understand each other's interactions. Since the underlying theme of the play is about love and romance, R.K. and Phajabi clearly understood the manners of propose, rejection, acceptance, greeting, wishing, questions and answers associated with the theme in the play.

We again calculated autocorrelation function using Equation (2), and found the values to be $c_{xx} = 0.31623$ and $c_{yy} = 0.99998$ respectively. It indicates that whatever turn-takings made by R.K. are not respond necessarily positive towards the main theme in the play regarding the emotion expressed by Phajabi. However, the emotion content in Phajabi in the main theme of the play is much more than R.K. and shows very serious interaction as the play actually does in the story and the plot.

Generally, protagonist has more dialogues so as they take more turns than other characters in most of the plays. In this play, two protagonists take more numbers of turns other than that of two supporting characters. So there are inadequate in the numerical values of turns in the whole play. Hence, the functions of each turn correlates according to the size and texture of the turns which have taken by the dramatis personae in the play. So, every qualitative and quantitative values made by the characters is reflected from their numbers of turns, therefore, the functions of information can be marked as per their number of turns.

Since the data of R.K. and Phajabi synchronize in their behaviours in between scenes 4 to 8, the functions of data shows that the information function contained in the interaction and the understanding lies between them is highly correlated; no matter the dialogues express rejection, support, agreement, mocking, satirical etc. However, during the initial scenes (1–4) and the last scenes (8–10), the data approximately uncorrelated which means that they are not able to persuade the communication properly towards the main theme or they diverted from the theme in order to maintain the time and situation. The nature of the data of Surbala shows uncorrelated behaviour towards that of the R.K. or Phajabi. Similarly, the data of Tombi also shows a nature of independent character in the play.

The information contents in data of R.K. and Phajabi which we have taken from Table 1 are calculated using permutation entropy technique. Since the length of each data is not large, we have chosen the length of short sequences partitioned ($q_i : i = 1, 2, \dots, M$) to be $L = 5$ and embedded dimension s is taken to 2,3 and 4. The permutation entropy spectrum of $s = 2, 3, 4$ are calculated using the procedure described in the previous section and the behaviors of all three values of s are approximately the same. Then we take the average of the corresponding values of all s and average permutation entropy spectra of R.K. and Phajabi are shown in Figure 2.

The plot of Phajabi shows that at the beginning of play in scene 3 the value of H is at lower in values (scene 3–4), however the value starts increasing as the number of turns increase in upcoming scene. This behaviour may be marked according to the amount of their emotions which are sharing among them. The content of shared emotion is also increased (due to anxiety, frustration, instability etc.) as the number of interactions increase between them and it may be a result of that the number of scenes are increased. Then after scene 7–8 the information contained is remained in constant. It may the cause of stabilizing emotion between them. The same behaviour can be obtained in the case of R.K. too. However, in the whole case of R.K. there is a cause of fluctuation in his behaviour where we compared to that of Phajabi.

In this work, the relation that is having between any two characters such as the type of relationship in which how much the relationship is strong between them can be captured through the study of correlation function and permutation entropy techniques based on the analysing of the turn-taking dialogues. This analysis could be qualitatively true because human relationships are generally delicate in nature and sometimes it is not much expressive. The techniques may not be pin pointed on what exactly is going on between the characters but could say how characters processes information between them. Moreover, permutation entropy technique can tell how complex human brain functions which is reflected in their respective turn-taking dialogues.

4. Conclusion

In conclusion, what the characters may do in turn-taking reveals their emotion as well as showing the functions of information conveyed in each dialogue of the individual character. The numeric values computed from numbers of turn-taking are used as the data for each character. From this data we estimate the variation of information contents in the play by computing correlation. Since the data is not large enough, the result may be too approximate, but the analysis gives us important understanding of relations between characters in the play in a qualitative sense.

Bibliography

Bandt, Christoph and Bernd Pompe. Permutation entropy: A natural complexity measure for time series. *Physical Review Letters*, 88(17):174102, 2002.

Cao, Yinhe, Wen-wen Tung, J.B. Gao, V.A. Protopopescu, L.M. Hively, et al. Detecting dynamical changes in time series using the permutation entropy. *PHYSICAL REVIEW-SERIES E-*, 70(4; PART 2):46217–46217, 2004.

Herman, Vimala. *Dramatic discourse: Dialogue as interaction in plays*. Routledge, 1998a.

Herman, Vimala. Turn management in drama. *Exploring the language of Drama, from text to context*, pages 19–33, 1998b.

Hutchby, Ian and Robin Wooffitt. *Conversation analysis: Principles, Practice and Analysis*. Polity, 1998.

Inao, Moirangthem. Nongaallabasu thaaballei manam (the lingering fragrance), 1995.

Levinson, Stephen C. *Pragmatics*. Cambridge University Press, 1983.

Linell, Per. *Approaching dialogue: Talk, interaction and contexts in dialogical perspectives*, volume 3. John Benjamins Publishing Company, 1998.

Moerman, Michael. *Talking culture: Ethnography and conversation analysis*. University of Pennsylvania Press, 1988.

Quiroga, Quian R., A. Kraskov, T. Kreuz, and P. Grassberger. Performance of different synchronization measures in real data: a case study on electroencephalographic signals. *Physical Review E*, 65(4):041903, 2002.

Sacks, Harvey. Lectures on conversation. volume i, ii. edited by g. jefferson with an introduction by e.a. schegloff, 1992.

Sacks, Harvey, Schegloff Emanuel A and Gail Jefferson. A simplest sytematics for the organisation of turn-taking for conversation. *Language*, 50:696–735, 1974.

Schegloff, Emanuel A. Introduction to Sacks, 1992.

Schiffrin, Deborah. Approach to discourse, 1994.

Address for correspondence:
Lourembam Surjit Singh
`lsurjit24@yahoo.com`
C/o, Administrative Office
Centre For Advanced Studies in Linguistics,
University of Delhi,
Arts Faculty, Ext- Building,
Delhi-110007, India.

PBML

The Prague Bulletin of Mathematical Linguistics

NUMBER 99 APRIL 2013 87–99

Utilisation des ressources externes pour la reformulation des requêtes dans un système de recherche d'information

Mohammed El Amine Abderrahim

Laboratoire de Traitement Automatique de la Langue Arabe
Faculté de Technologie, Université de Tlemcen
Algérie

Abstract

Dans un Système de Recherche d'Information (SRI), les démarches pour la reformulation de la requête sont nombreuses. Elles peuvent être classées selon les ressources utilisées en trois grandes approches : l'utilisation des ressources externes, l'analyse globale et l'analyse locale. Dans ce contexte et dans le cadre des SRI pour les textes Arabes, nous nous intéressons à l'évaluation des performances de la première approche. A cet effet deux ressources différentes ont été utilisées à savoir : WordNet Arabe et le Dictionnaire (thésaurus) des Sens Arabe. Les expérimentations réalisées sur un corpus de texte Arabe nous ont permis de mesurer l'apport de cette approche de reformulation de requête dans un SRI arabe.

1. Introduction

Afin de réduire la distance entre la pertinence système et la pertinence utilisateur, un SRI peut guider l'utilisateur vers une bonne formulation de ses besoins. Les solutions proposées tournent autour de trois approches à savoir : la Reformulation de la requête (RQ), le ré-ordonnancement des documents, la combinaison des résultats issus de différents SRI ou l'intégration du profil utilisateur dans le processus de recherche d'information. Dans cet article on s'intéresse particulièrement à la RQ. Les démarches pour cette dernière sont nombreuses et peuvent être classées selon les ressources utilisées en trois grandes classes (voir figure 1) :

1. L'analyse globale : cette approche consiste à analyser tout l'ensemble des documents de la collection pour extraire les termes pertinents à ajouter à la re-

Corresponding author : medamineabd@yahoo.fr
Cite as : Mohammed El Amine Abderrahim. Utilisation des ressources externes pour la reformulation des requêtes dans un système de recherche d'information. The Prague Bulletin of Mathematical Linguistics No. 99, 2013, pp. 87–99. doi : 10.2478/pralin-2013-0006.

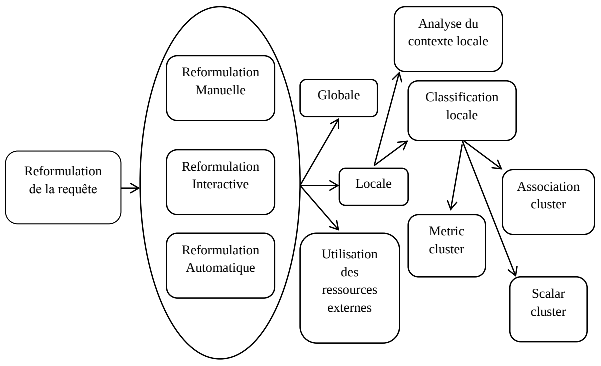
Reformulation de la requête
Reformulation Manuelle
Reformulation Interactive
Reformulation Automatique
Globale
Locale
Utilisation des ressources externes
Analyse du contexte locale
Classification locale
Association cluster
Metric cluster
Scalar cluster

FIG. 1 : Les approches pour la reformulation de la requête

quête initiale. Deux techniques sont alors développées (Baeza-Yates and Berthier, 1999) : le thesaurus de similarité (similarity thesaurus) et le thésaurus statistique (statistical thesaurus).

2. L'analyse locale : les documents retournés en réponse à une requête sont analysés pour extraire des termes pertinents qui serviront à étendre la requête. Les études effectuées dans (Baeza-Yates and Berthier, 1999 ; Carpineto and Romano, 2012 ; Aalbersberg, 1992 ; Xu and Croft, 2000 ; Lee et al., 2008 ; Salton and Buckley, 1990) montrent que contrairement à l'analyse globale, l'analyse locale est plus simple à réaliser et permet d'améliorer les performances d'un SRI. Deux techniques pour l'analyse locale sont proposées dans la littérature (Baeza-Yates and Berthier, 1999) :
   – La classification locale (local clustering) : consiste à construire une matrice d'association qui quantifie les relations de corrélation entre les termes issus de l'ensemble des documents retournés en réponse à la requête initiale. Selon la méthode de construction des relations de corrélation on distingue trois types de clusters : association clusters, metric clusters et scalar clusters.
   – L'analyse du contexte local : consiste à utiliser les concepts à la place de mot-clés pour représenter les documents (Xu and Croft, 2000).
3. L'utilisation des ressources externes : consiste à utiliser les ressources externes comme les ontologies ou les thésaurus pour trouver des termes similaires à la requête initiale (Efthimiadis, 1996 ; Harb et al., 2011). Nous nous intéressons dans la suite de cet article à cette technique de reformulation.

En plus de la reformulation manuelle, nous distinguons deux manières pour la RQ, la première est basée sur un processus automatique, elle se déroule sans l'intervention de l'utilisateur, on parle de reformulation automatique de la requête, alors que la seconde est basée sur un processus interactif entre le SRI et l'utilisateur, on parle de reformulation interactive de la requête. Les expérimentations effectuées dans le cadre de cette dernière ont montré qu'elle permet d'améliorer la précision des résultats, néanmoins son efficacité reste fortement lié à la disposition des utilisateurs et leurs aptitudes à juger la pertinence des documents (Baeza-Yates and Berthier, 1999 ; Bodo, 2005 ; Hlaoua, 2007 ; Bruande and Chevallet, 2003 ; Salton and Buckley, 1990 ; Black et al., 2006).

2. La reformulation de la requête par utilisation d'une ressource externe

La RQ par utilisation d'une ressource externe consiste à analyser premièrement la requête pour détecter les termes qui renvoient à des concepts de l'ontologie ou du thésaurus. Ces termes seront donc remplacés par des concepts proches en utilisant les relations sémantiques de l'ontologie.

Dans notre cas nous avons exploité le contenu de WordNet Arabe (AWN)[1] ou du Dictionnaire (thésaurus) des Sens Arabe (DSA)[2] pour reformuler et étendre des requêtes (par expansion) de manière à retrouver plus précisément les bons documents (voir figure 2). Dans le cadre de notre expérimentation nous avons testé la relation de synonymie.

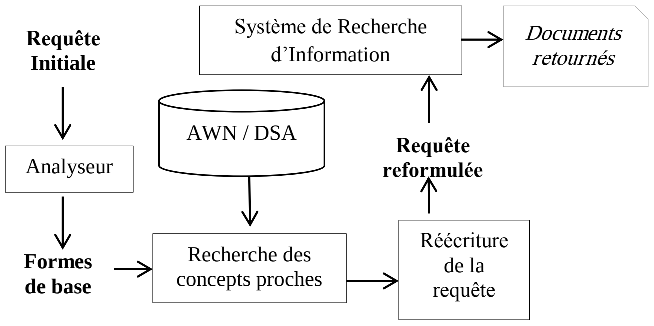

FIG. 2 : Les étapes de la reformulation de la requête

Il faut remarquer que le processus de modification de requêtes est indépendant du SRI, autrement dit l'enrichissement ne modifie pas la représentation interne des documents et des requêtes du SRI. Le but de l'enrichissement étant la formulation d'une

1. `http://www.globalwordnet.org/AWN/AWNBrowser.html`, voir aussi (Elkateb et al., 2006)

2. `http://www.almaany.com`

requête plus riche et donc plus précise. Une conséquence directe étant l'amélioration des performances du SRI en renvoyant des résultats plus pertinents.

Les travaux sur l'évaluation des techniques de la RQ pour les textes en arabe ne sont pas nombreux. Nous trouvons à titre d'exemples :

- Les travaux effectués dans Kanaan et al. (2005) ont montré que la reformulation manuelle manuelle par repondération des termes de la requête permet une amélioration des performances (rappel et précision) du SRI Arabe. Pour leur expérimentation, Kanaan et al. (2005) ont utilisé un corpus de 242 documents et un jeu de neuf (9) requêtes.
- Les travaux de Hammo et al. (2007) portant sur l'expansion de la requête par des termes issus d'un thésaurus montrent une amélioration dans le rappel du SRI arabe. Nous notons que le corpus utilisé était le coran.
- Les travaux de Xu et al. (2002) montrent que l'utilisation d'un thésaurus améliore considérablement (18%) les performances d'un SRI arabe. Xu et al. (2002) ont montré aussi que l'utilisation d'une indexation basée sur les racines est plus performante que l'utilisation des schèmes pour les textes arabes.
- Les travaux de Zaidi and Laskri (2007) sur l'expansion de la requête en utilisant une ontologie du domaine juridique et WordNet ont permet d'obtenir des améliorations considérables dans les performances du SRI.
- Le système de Ahmed and Nürnberger (2008) propose d'assister l'utilisateur dans la reformulation de sa requête, par l'ajout des formes proches morphologiquement des formes de la requête initiale, en se basant sur le calcul de similarité des n-grams entre les mots de la requête initiale et ceux enregistrés dans un lexique. Etant basée sur la similarité des chaines de caractères, cette approche ne peut résoudre le problème des variations lexicales ou sémantiques. Pour les opérations d'indexation et de recherche, Ahmed and Nürnberger (2008) ont utilisé les services du moteur de recherche Google.
- Les travaux de Abderrahim and A (2010) qui se résument à l'utilisation d'une ressource lexicale (WordNet Arabe) et un analyseur morphologique pour reformuler, par expansion, la requête de l'utilisateur permettent d'améliorer le rappel, mais pas la précision du SRI.
- Les travaux de Abderrahim and A (2012) sur l'évaluation de la stratégie de la classification locale pour les textes arabes montrent une amélioration dans les performances du SRI.

Cet article constitue une extension des travaux de Abderrahim and A (2010) en proposant d'évaluer l'apport réel de la reformulation de la requête guidée par une ontologie dans un SRI Arabe. Dans ce qui suit nous allons décrire notre expérimentation et discuter les résultats obtenus.

3. Expérimentation et discussion

Pour notre expérimentation nous avons utilisé un corpus de plus de 22 000 documents arabes (environ 180 Mo) de différents domaines (santé, sport, politique, science, religion, ...). Ce corpus compte environ 17 000 000 mots dont 612 650 mots différents. Un ensemble de 50 requêtes de type mots clés et de différents thèmes sont retenues pour l'expérimentation.

Pour les opérations d'indexation et de recherche nous avons utilisé l'API Lucene [3] qui est librement disponible sur le net. Par ailleurs, pour le processus de reformulation nous avons codé en java trois stratégies différentes que nous décrirons dans la suite de cette section. Nous avons aussi utilisé deux ressources externes différentes pour la recherche des concepts proches dans la phase de la reformulation de la requête. La première est WordNet arabe qui est une des rares ressources librement disponibles pour la langue arabe. Il compte actuellement 11 269 synsets et 23 481 mots (Alkhalifa, 2006 ; Elkateb et al., 2006 ; Black et al., 2006). Par ailleurs la seconde ressource est le DSA qui est aussi librement consultable en ligne. Dans sa version actuelle, il compte 20 500 synonymes et 35 000 mots. Du point de vue du contenu, le DSA est plus riche en informations comparé à WordNet arabe.

Dans un premier temps nous avons indexé l'ensemble des documents de notre corpus. Le résultat de cette opération nous a permis d'obtenir un index d'une taille d'environ 37 Mo (environ 1/5 de la taille du corpus utilisé). Cet index sera utilisé pour faire la recherche des informations en réponse aux 50 requêtes reformulées selon différents protocoles. La stratégie de reformulation est basée sur une expansion aveugle, contrôlée (sélective) et pondérée des concepts présents dans les requêtes. Pour chaque stratégie de reformulation nous avons fait correspondre un type de recherche que nous allons étudier séparément pour évaluer son apport dans l'amélioration des performances de la recherche. Il faut noter toutefois que chaque stratégie a été testée deux fois : une fois en utilisant WordNet arabe, et une autre fois en utilisant DSA.

Nous avons examinés quatre différents types de recherche :

1. Recherche simple ou recherche avant enrichissement (RS) : nous avons utilisé la liste des 50 requêtes simples de type mots clés, par exemple les requêtes :

   صبغ الشيب, صداقة جادة

2. Recherche Aveugle (RA) : nous avons utilisé une liste de 50 requêtes déduites des requêtes simples de départ par un enrichissement aveugle (nous ajoutons à la requête initiale l'ensemble des synonymes trouvé dans WordNet arabe ou DSA). Le tableau 1 présente des exemples de ce type de recherche selon la ressource externe utilisée.

3. `http://lucene.apache.org/`

Num	Requête simple (RS)	Nouvelle requête enrichie (RP) A partir de WordNet Arabe	Nouvelle requête enrichie (RP) A partir de DSA
1	صبغ الشيب	صبغ الشيب ، طلى ، دهن ، لون ، خضب ، تلوين ، صباغ ، صبغة	صبغ الشيب ، لون ، خضب ، تلوين ، طلى ، لون ، ضمخ
2	صداقة جادة	صداقة جادة ، ألفة ، عشرة ، صحبة ، خلة	صداقة جادة ، صحبة ، رفقة ، مخادنة ، ألفة ، مصاحبة ، إخاء ، ألفة ، مؤانسة

Tab. 1 : Exemples de requêtes en recherche Aveugle selon la ressource utilisée

3. Recherche Contrôlée (RC) : nous avons utilisé une liste de 50 requêtes déduites des requêtes simples par un enrichissement manuel (nous ajoutons à la requête seulement les synonymes sélectionnés manuellement). La construction de la requête enrichie se fait interactivement entre le système et l'utilisateur, le scénario est le suivant : Après analyse de la requête, le système recherche les synonymes des termes de la requête et ensuite, il propose à l'utilisateur de valider les termes suggérés (par suppression des termes non appropriés). Le tableau 2 présente des exemples de ce type de recherche selon la ressource externe utilisée.

Num	Requête simple (RS)	Nouvelle requête enrichie (RP) A partir de WordNet Arabe	Nouvelle requête enrichie (RP) A partir de DSA
1	صبغ الشيب	صبغ الشيب ، لون ، خضب ، تلوين ، صباغ ، صبغة	صبغ الشيب ، خضب ، تلوين ، طلى ، لون
2	صداقة جادة	صداقة جادة ، ألفة ، عشرة ، صحبة ، خلة	صداقة جادة ، صحبة ، رفقة ، ألفة ، مصاحبة ، إخاء ، مؤانسة

Tab. 2 : Exemples de requêtes en recherche contrôlée selon la ressource utilisée

4. Recherche Pondérée (RP) : nous avons utilisé une liste de 50 requêtes déduites des requêtes simples par un enrichissement automatique. Nous avons ajouté à la requête seulement un seul synonyme choisi automatiquement. La procédure de choix se base sur la pondération affectée à chaque terme. Le terme ayant la pondération la plus élevée est retenu, intuitivement, c'est le sens le plus utilisé.

Comme nous ne disposons pas de cette information de pondération des termes dans WordNet arabe ou DSA, nous avons calculé cette valeur à partir d'une analyse statistique de tous les documents de notre corpus. Dans la littérature il existe plusieurs méthodes de pondération, cependant, elles présentent toutes une variation de la formule « TF.IDF ». Dans cette expérimentation, nous avons choisi comme pondération la valeur TF (nombre d'occurrence d'un terme dans le corpus). Il est clair que la valeur de discrimination « IDF » ne présente aucun intérêt pour notre cas puisqu'elle ne permet de mettre en valeur que les termes qui apparaissent dans peu de documents. Le tableau 3 présente des exemples de ce type de recherche selon la ressource externe utilisée.

Num	Requête simple (RS)	Nouvelle requête enrichie (RP) A partir de WordNet Arabe	Nouvelle requête enrichie (RP) A partir de DSA
1	صبغ الشيب	صبغ الشيب ، لون	صبغ الشيب ، لون
2	صداقة جادة	صداقة جادة ، عشرة	صداقة جادة ، صحبة

Tab. 3 : Exemples de requêtes en recherche pondérée selon la ressource utilisée

Les résultats obtenus par ces quatre différents types de recherche sont consignés dans différents fichiers et pour chaque type de recherche et chaque requête nous avons calculé les différents rappels et précisions du système.

4. Analyse des résultats et discussion

Le tableau 4 présente les précisions à 11 points de rappels des différents systèmes associés à chaque type de recherche.

La figure 3 présente les différentes courbes rappel/précision obtenues à partir du tableau 4.

L'examen de la figure 3 nous permet de constater que :

– La reformulation testée avec ses différents types permet d'avoir une amélioration des performances du SRI dans l'intervalle de rappel [0; 0,1] de la RP utilisant DSA.
– Quelle que soit la ressource utilisée la RP donne les meilleures performances.
– La RA présente la technique la plus pauvre en performance.
– L'utilisation de DSA est plus bénéfique que WordNet Arabe.

Pour comprendre l'effet des différents types de recherche sur chaque requête, nous avons établi différentes mesures qui sont principalement basées sur la comparaison

		Utilisation de WordNet Arabe			Utilisation de DSA		
Rappel	RS	RA	RC	RP	RA	RC	RP
0	0,620	0,484	0,543	0,623	0,419	0,517	0,646
0,1	0,616	0,453	0,516	0,577	0,394	0,473	0,584
0,2	0,587	0,427	0,487	0,522	0,386	0,456	0,532
0,3	0,557	0,392	0,452	0,461	0,362	0,428	0,471
0,4	0,543	0,359	0,430	0,424	0,336	0,399	0,436
0,5	0,508	0,330	0,398	0,396	0,315	0,373	0,401
0,6	0,464	0,305	0,362	0,364	0,291	0,347	0,373
0,7	0,431	0,293	0,340	0,340	0,269	0,328	0,336
0,8	0,402	0,281	0,323	0,319	0,259	0,314	0,312
0,9	0,368	0,257	0,298	0,287	0,243	0,290	0,284
1	0,304	0,207	0,242	0,226	0,201	0,231	0,233

TAB. 4 : Les précisions à 11 points de rappels selon la ressource utilisée

des résultats avant et après enrichissement. Dès lors, pour une requête donnée, trois cas peuvent se présenter :

– Amélioration : toutes les précisions à 11 points de rappel avant sont inférieures à ceux d'après (i.e. : la courbe rappel / précision après est au-dessus de avant).
– Pas d'amélioration : c'est le cas inverse du précédent (i.e. : la courbe rappel / précision après est au-dessous de avant).
– Sans décision : pour certaines précisions à 11 points il y a amélioration et pour d'autres il n'y a pas d'amélioration (i.e. : intersection des deux courbes rappel / précision avant et après).

Le tableau 5 présente pour chaque requête utilisée dans l'expérimentation l'indicateur Amélioration(+), Pas d'amélioration(-) ou Sans décision(X) des différents types de recherche.

L'examen des résultats obtenus dans le tableau 5 nous permet de déduire les faits suivants :

1. Quelle que soit la méthode de reformulation testée, l'utilisation de WordNet Arabe permet d'avoir :
   – Une amélioration (+) dans 4 requêtes (les requêtes numéro : 12, 27, 28, 38) soit 8%.
   – Pas d'amélioration (-) dans 21 requêtes (42%).
   – Une indécision (X) dans 6 requêtes (12%).
   – Un ensemble de 19 requêtes (39%) pour lesquelles il existe au moins une amélioration dans 6 requêtes (12%) dans l'une des méthodes de reformulation

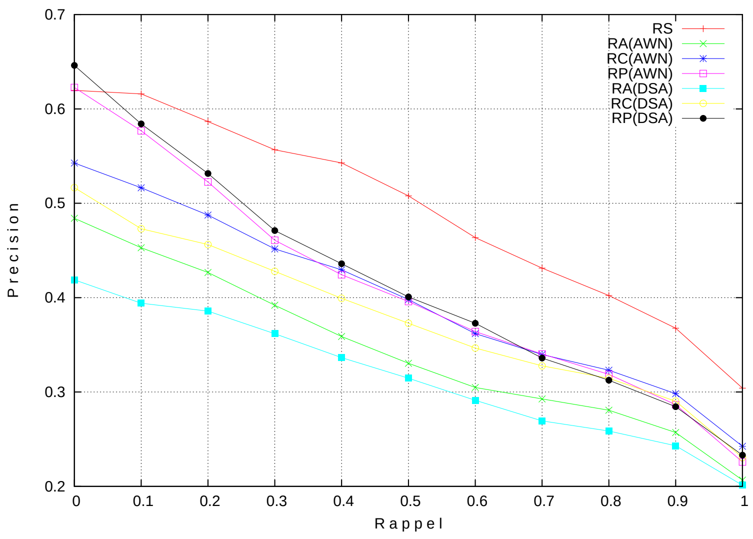

FIG. 3 : Les courbes rappel/précision à partir du tableau 4

testée. Pour les 13 requêtes restantes (26%) soit il n'y a pas d'amélioration soit une indécision.

2. Quelle que soit la méthode de reformulation testée, l'utilisation de DSA permet d'avoir :
   - Une amélioration (+) dans 2 requêtes (les requêtes numéro : 11, 36) soit 4%.
   - Pas d'amélioration (-) dans 16 requêtes (32%).
   - Une indécision (X) dans 6 requêtes (12%).
   - Un ensemble de 26 requêtes (52%) pour lesquelles il existe au moins une amélioration dans 6 requêtes (12%) dans l'une des méthodes de reformulation testée. Pour les 20 requêtes restantes (40%) soit il n'y a pas d'amélioration soit une indécision.

Du point de vue de l'amélioration, l'analyse des faits précédents (1 et 2) nous permet d'annoncer que la reformulation par utilisation d'une ressource externe permet d'améliorer les performances d'un SRI Arabe d'environ 6% ; Par ailleurs, quelle que soit la méthode préconisée pour la reformulation, l'utilisation de WordNet Arabe est bien meilleure que DSA. Pour déterminer la meilleure technique de recherche entre RA, RC et RP nous avons comptabilisé le nombre des requêtes pour chaque type de recherche dans le tableau 6.

Les résultats obtenus dans le tableau 6 ne permettent pas de faire un choix entre RA, RS et RP dans le cas de l'utilisation de WordNet Arabe, car nous avons le même

Num Requête	Utilisation de WordNet Arabe			Utilisation de DSA		
	RA	RC	RP	RA	RC	RP
1	-	-	-	-	-	-
2	X	X	X	-	-	-
3	-	X	X	-	-	X
4	X	-	X	-	-	-
5	X	X	X	-	X	X
6	-	-	-	-	-	-
7	-	-	X	-	X	-
8	-	-	-	-	X	-
9	+	X	+	X	-	X
10	-	-	X	-	-	-
11	X	-	-	+	+	+
12	+	+	+	+	+	X
13	X	X	-	-	X	X
14	X	X	X	X	X	X
15	-	-	-	-	-	X
16	-	-	-	-	-	-
17	-	-	-	-	-	-
18	-	-	-	-	-	-
19	X	X	X	X	X	X
20	-	-	-	-	-	-
21	X	X	X	X	X	X
22	X	+	X	+	X	X
23	X	X	+	X	X	X
24	X	X	X	X	X	-
25	-	-	-	-	-	-
26	-	X	-	-	X	X
27	+	+	+	X	X	X
28	+	+	+	-	-	+
29	-	-	-	-	-	-
30	-	-	X	-	-	-
31	-	-	-	-	-	X
32	+	+	X	X	+	X
33	-	X	X	-	-	X
34	-	-	-	-	-	X
35	-	-	-	X	X	X
36	X	X	-	+	+	+
37	-	-	-	-	-	-
38	+	+	+	X	X	+
39	-	-	-	-	X	-
40	-	-	-	-	-	-
41	-	-	-	-	-	X
42	-	-	-	X	X	+
43	X	-	-	-	-	-
44	X	X	-	X	X	-
45	-	-	X	-	X	X
46	+	+	X	-	X	X
47	-	-	-	-	X	
48	-	-	-	-	-	X
49	-	-	-	-	-	-
50	-	-	+	-	-	-

Tab. 5 : L'indicateur Amélioration(+), Pas d'amélioration(-) ou Sans décision(X) des différents types de recherche selon la ressource utilisée.

taux d'amélioration (14%) pour les trois cas. En revanche, il est clair que la technique RP (10%) présente le meilleur taux d'amélioration dans le cas de l'utilisation de DSA, d'ailleurs l'analyse des courbes rappel/précision de la figure 3 ne fait que confirmer ce résultat. Tout compte fait, nous pouvons conclure que l'apport de l'utilisation d'une ressource externe dans un SRI Arabe est d'environ 6%. Par ailleurs, il apparaît que WordNet Arabe est meilleur que DSA et que RP offre le taux d'amélioration le plus élevé.

	Nombre de requêtes					
	Utilisation de WordNet Arabe			Utilisation de DSA		
	RA	RC	RP	RA	RC	RP
Amélioration	7 (14%)	7 (14%)	7 (14%)	4 (8%)	4 (8%)	5 (10%)
Pas d'amélioration	29 (58%)	29 (58%)	27 (54%)	34 (68%)	26 (52%)	23 (46%)
Sans décision	14 (28%)	14 (28%)	16 (32%)	12 (24%)	20 (40%)	22 (44%)

Tab. 6 : Le nombre de requêtes vérifiant les conditions : Amélioration, Pas d'amélioration et Sans décision selon la ressource utilisée.

5. Conclusion

Dans cet article, nous avons examiné différentes manières pour faire la reformulation d'une requête dans un SRI Arabe. Cette reformulation étant basée sur une ressource externe, nous avons particulièrement expérimentée deux ressources à savoir WordNet Arabe et DSA. Les résultats obtenus nous ont permis de mesurer l'apport (6%) d'une telle approche dans l'amélioration des performances globales d'un SRI Arabe. Du point de vue du nombre de requêtes ayant effectivement conduits à une amélioration, les résultats de comparaison entre l'utilisation de WordNet Arabe et DSA sont en faveur du premier. En revanche, du point de vue "rappel/précision", l'utilisation de DSA est plus bénéfique que celle de WordNet Arabe. Par ailleurs, nous avons conclu que la technique de RP offre un meilleur taux d'amélioration des performances d'un SRI Arabe.

Cette étude nous a permis d'ouvrir la voie pour tester et comparer d'autres méthodes de reformulation avec les mêmes données de cette expérimentation afin de déterminer la technique la plus appropriée à intégrer dans un SRI Arabe.

6. Summary

In information retrieval systems (IRS), approaches to query reformulation are numerous. They can be classified according to the used resources in three main approaches: those using external resources, global analysis, and local analysis. In this context and as part of an IRS for Arabic texts, we are interested in evaluating performance of the first approach. For this purpose, two different resources are used, namely Arabic WordNet and the Arabic Dictionary (thesaurus) of Meaning. The ex-

periments conducted on a corpus of Arabic text allowed us to measure the contribution of this query reformulation approach applied to an Arabic IRS.

Références

Aalbersberg, IJsbrand Jan. Incremental Relevance Feedback. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 11–22. ACM, 1992.

Abderrahim, Med-El-Amine and Med-Alaeddine A. Using Arabic Wordnet for Query Expansion in Information Retrieval System. In *IEEE The Third International Conference on Web and Information Technologies*, Marrakech, Morocco, June 2010.

Abderrahim, Med-El-Amine and Med-Alaeddine A. Réinjection automatique de la pertinence pour la recherche d'informations dans les textes arabes. In *IEEE 4th International Conference on Arabic Language Processing (CITALA)*, pages 77–81, Rabat, Morocco, May 2012.

Ahmed, Farag and Andreas Nürnberger. Arasearch: Improving Arabic Text Retrieval via Detection of Word Form Variations. In *SIIE 2008*, pages 309–323, Hammamet, Tunisie, February 2008.

Alkhalifa, Musa. Arabic Wordnet and Arabic NLP. In *Journées d'Etudes sur le Traitement Automatique de la Langue Arabe (JETALA)*, Rabat, June 2006.

Baeza-Yates, Ricardo and Ribeiro-Neto Berthier. *Modern Information Retrieval*. Addison-Wesley, New York City, NY ACM Press, 1999.

Black, William, Sabri Elkateb, Horacio Rodriguez, Musa Alkhalifa, Piek Vossen, Adam Pease, and Christiane Fellbaum. Introducing the Arabic WordNet Project. In *Proceedings of the Third International WordNet Conference*, pages 295–300, 2006.

Bodo, Billerbeck. *Efficient Query Expansion*. PhD thesis, RMIT University, Melbourne, Australia, 2005.

Bruande, Marie-France and Jean-Pierre Chevallet. Assistance intelligente à la recherche d'information. *Lavoisier*, pages 99–129, 2003.

Carpineto, Claudio and Giovanni Romano. A survey of Automatic Query Expansion in Information Retrieval. *ACM Computing Surveys (CSUR)*, 44(1):1, 2012.

Efthimiadis, Efthimis N. Query Expansion. *Annual review of information science and technology*, 31:121–187, 1996.

Elkateb, Sabry, William Black, Piek Vossen, David Farwell, H Rodríguez, A Pease, and M Alkhalifa. Arabic WordNet and the Challenges of Arabic. In *Proceedings of Arabic NLP/MT Conference, London, UK*, 2006.

Hammo, Bassam, Azzam Sleit, and Mahmoud El-Haj. Effectiveness of Query Expansion in Searching the Holy Quran. In *Proceedings of L'institut organise le 2ème Colloque International sur le Traitement Automatique de la Langue Arabe (CITALA)*, volume 7, pages 18–19, Morroco, 2007.

Harb, Hany M, Khaled M Fouad, and Nagdy M Nagdy. Semantic Retrieval Approach for Web Documents. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 2 (9):11–75, 2011.

Hlaoua, Lobna. *Reformulation de requêtes par réinjection de pertinence dans les documents semi-structurés*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2007.

Kanaan, G, R Al-Shalabi, M Abu-Alrub, and M Rawashdeh. Relevance Feedback: Experimenting with a Simple Arabic Information Retrieval System with Evaluation. *International Journal of Applied Science and Computations*, 12(2), 2005.

Lee, Kyung Soon, W Bruce Croft, and James Allan. A Cluster-based Resampling Method for Pseudo-relevance Feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 235–242. ACM, 2008.

Salton, Gerard and Chris Buckley. Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science*, 41(4):288–97, 1990.

Xu, Jinxi and W Bruce Croft. Improving the Effectiveness of Information Retrieval with Local Context Analysis. *ACM Transactions on Information Systems (TOIS)*, 18(1):79–112, 2000.

Xu, Jinxi, Alexander Fraser, and Ralph Weischedel. Empirical Studies in Strategies for Arabic Retrieval. In *Annual ACM Conference on Research and Development in Information Retrieval: Proceedings of the 25 th annual international ACM SIGIR conference on Research and development in information retrieval*, volume 11, pages 269–274, 2002.

Zaidi, S and M Laskri. Expansion de la requête Arabe sur le réseau internet. In *Barmajiat (CSLA): Les applications logicielles en arabe: Pas vers le e-gouvernement*, Alger, December 2007.

Address for correspondence:
Mohammed El Amine Abderrahim
medamineabd@yahoo.fr
Université de Tlemcen
Faculté de Technologie
BP 230 Chetouane,
Tlemcen (13000) Algérie

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 99 APRIL 2013 101–105

REVIEWS

Review of the book "Morfologie českého slovesa a tvoření deverbativ jako problém strojové analýzy češtiny"

(The Morphology of the Czech Verb and Verb Derived Nouns and Adjectives as a Problem of the Formal Description and Automatic Analysis of the Czech Language)

Klára Osolsobě

Spisy Masarykovy univerzity v Brně, Filozofická fakulta, č. 401. MUNI Press 2011, 220 pp., ISBN 978-80-210-5565-0.

Reviewed by Jaroslava Hlaváčová

Czech deverbatives are usually derived from verbs by means of mostly very regular patterns. The rules for deverbative derivations have been described by many linguists in many previous works. Nonetheless, those descriptions are not possible to convert directly into a system usable in the field of natural language processing (NLP). They are not detailed enough to cover the whole diversity of morphological alternations in the particular verb forms taken as the basis for the derivations. For automatic processing, it is necessary to present the description down to the slightest detail. Moreover, the rules have to be expressed within a formalism that ensures their exactness and unambiguity.

The only widely used system utilizing the regularity of deverbative generation in NLP is the Prague morphological tool, in which the generation rules constitute an intrinsic part. However, the rules are very general, the result of which is a massive overgeneration. In morphological analysis, it does not matter as much, but in other fields of usage, it is naturally better to generate only those words that (may) really exist.

Thus, the presented treating deverbatives with respect to their application in NLP is a meritorious achievement.

All the linguistic data were taken from the Czech morphological dictionary which forms the basis for the automatic morphological analyzer "ajka" used for morphological tagging Czech corpora at Masaryk University in Brno. Until now, the dictionary has served mainly for analysis and synthesis of inflected word forms. The book brings an impulse for its usage in the domain of word formation. The first step was made with deverbatives, as they are one of the most productive parts of the Czech word-formation.

For automatic finding candidates of deverbatives, the software tool Deriv developed at the Faculty of informatics at MU Brno was used. It searches the dictionary for pairs of words, that meet special substitution rules transforming the first word into the second one. In that sense, the "derivation" is in reality no real derivation. It is "mere" checking, if there are pairs of words present in the dictionary, fulfilling certain (substitution) rules. The author prepared tens of such rules. The resulting pairs had to be checked manually because of great homonymy. The final lists are not presented in the book. They rest on the server where they were processed. The author gives (in Section IX) only a list of directories and files where the results are stored. The results are not accessible directly, it is necessary to ask an administrator for permission. I used an anonymous access, so I was able to use the tool for several trials, but did not manage to look into the author's results. I would expect that the author make her findings, or at least several representative examples, accessible.

The book is organized "procedurally". The Introduction (Section I) contains a short historical overview. The theoretical background and terminology is covered by the next three sections. Section V contains observations of the data with consequences in the form of rules that are later (in Section VII) specified more precisely for using in the automatic tool Deriv which is described in Section VI. Having a brief user introduction to Deriv, the Section VII brings the final rules used for searching the dictionary with the tool. Sections VIII and IX are devoted to the results. The Conclusion (Section X) summarizes the book, together with a short English résumé. The book ends with a detailed bibliography and three appendices: A – description of morphological tags, B – web interface of Deriv, C – examples of results.

Let us have a more detailed look at the essential parts of the book.

In the first three sections, there is a very brief overview of the morphological tools used in Brno. There is also stated the objective of the work – the formal description of "realized" derivations of selected types of Czech deverbatives. The word in quotes is important (and as such should not be explained in the book only as a footnote), because it restricts the linguistic material only to the morphological dictionary itself. The author admits that it would be more valuable to use a larger basis, for instance a big language corpus. However, the dictionary size is still quite large. Another aim of the work was to investigate the possibility of automatic language analysis with respect to word-formation and the dictionary served as a natural word reserve ready to use. Moreover, the dictionary offers additional pieces of information, namely the morphological tags. If a corpus was utilized for a similar research, there would be necessary to choose a slightly different approach, as there appear unknown words, not recognized by the morphological dictionary. As such, the research cannot rely on their morphological tags.

The fourth section presents the terminology used in word segmentation, with the special attention to deverbatives.

Section VI contains a short user manual of the software tool Deriv. There is also a very brief introduction into the syntax of regular expressions (table on p. 46). The tool itself was probably designed to fit the needs of the author and as such it works surely very well. However, its user interface is not very intuitive and it would demand certain time to get used to it. Nevertheless, it is not the subject of this review.

The heart of the book are the sections V, VII and VIII where the main fiddly work of the author is presented.

Section V summarizes morphological alternations occurring in verbs and deverbatives. They are presented in the form of "observations", from which the author derives "rules".[1] The section is divided into several subsections, first of them being introductory, others are sorted according to the place where the alternation occurs.

Section VII brings the list of all analyzed types of deverbatives, each of them having its own subsection. They have always the same structure: At first there is a description of the derivation, including alternations relevant to the given type. Then comes a formulation of substitution rules for the software tool Deriv. The rules are presented in the form of a table, together with an example and the number of resulting pairs [verb, deverbative]. The last two columns express the number of real pairs and over-generated ones.[2] Each subsection ends with a discussion about observed overgeneration illustrated by a bar graph. My notice concerning graphs in the whole book see below.

Section VIII presents a quantitative analysis of all the resulting pairs [verb, deverbative] with respect to the overgeneration. Derivation types are compared in groups that were created according to the basis of the generation. The tables given for every group repeat summed figures from the tables of Section VII, but some mistakes have crept in.[3] They are not serious mistakes, but their presence unfortunately decrease the credit of the whole, in reality very laborious and valuable work.

The main shortcoming of the book is lack of index. Also a list of all the abbreviations presented on a separate page at a special place of the book (at the beginning, or

[1]Their numbering, especially on pages 26 to 30, is a bit confusing.

[2]Here, a defect occurs in all the subsections – the tables presenting substitution rules do not have properly labeled the 3rd (no label) and 4th column (label "pair" is not instructive enough). Similar tables in Section VIII are labeled correctly.

[3]See for instance numbers at the top of page 76, where the percentage is calculated correctly, and the summarized table with a slightly different (and wrong) figure on page 171 for the suffix *-tel*. Another mistake occurs in the figures for the suffix *-dlo* on page 171 where right and over-generated pairs do not sum to the total given in the third column (wrong copy of figures from page 91).

at the end) would be appreciated. There are quite a lot of abbreviations used through the whole book, but their explanation is presented only inside the text, mainly on pages 20 and 24.[4] Also the system of references within the book is not adequate. The references "see above / below" are not possible to follow, if they point farther than several paragraphs.[5] I have also a critical remark on graphs. They are presented in Section VII for individual deverbative suffixes, and then in Section VIII, where they are summarized according to the types of their generation. All the graphs present the same – relation between number of correctly generated deverbatives and overgenerated ones, expressed in percents. Thus, there are always pairs of numbers summing up to 100%. The graphs are not designed very well. Firstly, they all are too wide for the presented data – the right (and bigger) side of every graph is always empty. Secondly, the marks of the x-axis do not have any value – in fact, they do not represent anything and therefore their presence is very confusing. And thirdly, as all the graphs present the same relationship, it would be more illustrative if the y-axis was always calibrated from 1 to 100, not sometimes to another amount, according to the concrete numbers.[6] In my opinion, the graphs would be more clear, if there were only one bar, always of the same length, divided in the given ratio, not two of them. A horizontal bar, always in the same size, with numbers expressed directly in the graph, not above, would probably fit the given purpose the best.

Throughout the book, there is quite a lot of minor inaccuracies, typos, sometimes even minor mistakes. Inconsistencies are present also in formalized expressions of strings. Sometimes, the author uses regular expressions, sometimes she chooses possibly a "more readable" slash (/) for alternatives, sometimes even within one paragraph,[7] or one expression.[8] The slash is actually overused as a whole, which sometimes may cause problems with understanding, especially when it is used within one sentence in more meanings, as in the 3rd paragraph from the bottom on page 47 where the slashes mean at first alternatives and at the end of the sentence it serves as a sign for a pair.

My final critical remark concerns the names of sections and subsections. The main sections do not have their names, they are only numbered. It is naturally the question of the author's attitude, but the subsections should perhaps be more structured, probably also numbered for a better cross referencing. I would also number the graphs and tables.

[4]Moreover, the abbreviation VSB introduced there under item *d*) is modified as SBV in item *h*) and later in the table summarizing these abbreviations (page 26 above). In the footnote 97 on the page 42, there is again VSB.

[5]See for instance footnotes 476 and 477 on page 171.

[6]Compare for instance graphs on pages 72, 83 and 87.

[7]Rule 2 on p. 42.

[8]At the end of the Rule 1 on the same page.

The book contains three essential results: the lists of pairs [verb, deverbative] that can be used in various domains of NLP, such as machine translation, data mining, summarization and many others. The second substantial achievement is the formulation of the set of formalized rules describing the relationship between verbs and their deverbatives. Those rules may be utilized for detection of other similar pairs, that have not been present in the morphological dictionary, in another source – for instance in a language corpus. The third contribution is the method itself. Formulation of additional rules may detect other relationships among words of different parts of speech and their derivatives.

The author has investigated an immense amount of linguistic material, categorized it and stated formal rules for derivation of deverbative nouns and adjectives from verbal stems or word forms. The book represents an important contribution to the domain of Czech word-formation. My main critical remarks concerned mostly technical aspects of the book. Such a laborious work would definitely deserve a more careful technical preparation. However, the factual content of the book presents an important bridge between descriptive and computational linguistics and as such constitutes an essential achievement in the modern linguistic research.

Address for correspondence:
Jaroslava Hlaváčová
hlavacova@ufal.mff.cuni.cz
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics,
Charles University in Prague
Malostranské náměstí 25
118 00 Praha 1, Czech Republic

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 99 APRIL 2013 107–108

NOTES

Second International Conference on Dependency Linguistics DepLing 2013

http://ufal.mff.cuni.cz/project/depling13/

DepLing 2013 will be held at the Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic on August 27–30, 2013. The conference is organized by the Institute of Formal and Applied Linguistics.

DepLing 2013 is the second conference in the DepLing series (the first was held in Barcelona, Spain in 2011). The series responds to the growing need for linguistic meetings dedicated to syntactic, semantic and lexicographic approaches that are centered around dependency as a central notion.

The local committee of the conference is chaired by Eva Hajičová (Charles University in Prague). The scientific committee is chaired by Kim Gerdes (Sorbonne Nouvelle, Paris), Leo Wanner (Universitat Pompeu Fabra) and Eva Hajičová. Invited speakers are: Aravind Joshi (IRCS and CIS, Philadelphia), Richard Hudson (UCL, London).

DepLing 2013 is followed by the international workshop on Meaning-Text Theory (MTT), (August 30–31).

DepLing 2013 is supported by The Vilem Mathesius Center and the LINDAT / CLARIN project.

Contacts: Eva Hajičová (hajicova@ufal.mff.cuni.cz) and Anna Kotěšovcová (organizational matters, kotesovcova@ufal.mff.cuni.cz).

MT Marathon 2013 Open Source Convention

http://www.statmt.org/mtm13

The Machine Translation Marathon 2013, MTM2013, will take place in Prague, Czech Republic in September 9–14th, 2013.

The MT Marathon will again host an Open Source Convention to advance the state of the art in machine translation. We invite developers of open source tools to present their work and submit a paper of up to 10 pages that describes the underlying methodology and includes instructions on how to use the tools.

We are looking for stand-alone tools and extensions of existing tools, such as the Moses open source system. Accepted papers will be presented during the MT Marathon and published in the Prague Bulletin of Mathematical Linguistics.

Possible Topics

- Training of Machine Translation models
- Machine Translation decoders
- Tuning of Machine Translation systems
- Evaluation of Machine Translation
- Visualisation, annotation or debugging tools
- Tools for human translators
- Interfaces for web-based services or APIs
- Extensions of existing tools
- Other tools for Machine Translation

Important Dates

Abstract submission:	July 5, 2013 (1 paragraph, to allocate reviewers)
Paper submission:	July 19, 2013
Notification of acceptance:	August 6, 2013
Camera-ready:	August 13, 2013
Presentations:	September 9–14, 2013 (MT Marathon in Prague)

Instructions for the authors and full details are available at the web page http://www.statmt.org/mtm13.

PBML

The Prague Bulletin of Mathematical Linguistics

NUMBER 99 APRIL 2013

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`.