IUCL: Combining Information Sources for SemEval Task 5

Alex Rudnick, Levi King, Can Liu, Markus Dickinson, Sandra Kübler
Indiana University
Bloomington, IN, USA
{alexr,leviking,liucan,md7,skuebler}@indiana.edu

Abstract

We describe the Indiana University system for SemEval Task 5, the L2 writing assistant task, as well as some extensions to the system that were completed after the main evaluation. Our team submitted translations for all four language pairs in the evaluation, yielding the top scores for English-German. The system is based on combining several information sources to arrive at a final L2 translation for a given L1 text fragment, incorporating phrase tables extracted from bitexts, an L2 language model, a multilingual dictionary, and dependency-based collocational models derived from large samples of target-language text.

1 Introduction

In the L2 writing assistant task, we must translate an L1 fragment in the midst of an existing, nearly complete, L2 sentence. With the presence of this rich target-language context, the task is rather different from a standard machine translation setting, and our goal with our design was to make effective use of the L2 context, exploiting collocational relationships between tokens anywhere in the L2 context and the proposed fragment translations.

Our system proceeds in several stages: (1) looking up or constructing candidate translations for the L1 fragment, (2) scoring candidate translations via a language model of the L2, (3) scoring candidate translations with a dependency-driven word similarity measure (Lin, 1998) (which we call *SIM*), and (4) combining the previous scores in a log-linear model to arrive at a final n-best list. Step 1 models transfer knowledge between the L1 and L2; step 2 models facts about the L2 syntax, *i.e.*, which translations fit well into the local context; step 3 models collocational and semantic tendencies of the L2; and step 4 gives different weights to each of the three sources of information. Although we did not finish step 3 in time for the official results, we discuss it here, as it represents the most novel aspect of the system – namely, steps towards the exploitation of the rich L2 context. In general, our approach is language-independent, with accuracy varying due to the size of data sources and quality of input technology (*e.g.*, syntactic parse accuracy). More features could easily be added to the log-linear model, and further explorations of ways to make use of target-language knowledge could be promising.

2 Data Sources

The data sources serve two major purposes for our system: For L2 candidate generation, we use Europarl and BabelNet; and for candidate ranking based on L2 context, we use Wikipedia and the Google Books Syntactic N-grams.

Europarl The Europarl Parallel Corpus (Europarl, v7) (Koehn, 2005) is a corpus of proceedings of the European Parliament, containing 21 European languages with sentence alignments. From this corpus, we build phrase tables for English-Spanish, English-German, French-English, Dutch-English.

BabelNet In the cases where the constructed phrase tables do not contain a translation for a source phrase, we need to back off to smaller phrases and find candidate translations for these components. To better handle sparsity, we extend look-up using the multilingual dictionary BabelNet, v2.0 (Navigli and Ponzetto, 2012) as a way to find translation candidates.

Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 356–360,
Dublin, Ireland, August 23-24, 2014.

Wikipedia For German and Spanish, we use recent Wikipedia dumps, which were converted to plain text with the Wikipedia Extractor tool.[1] To save time during parsing, sentences longer than 25 words are removed. The remaining sentences are POS-tagged and dependency parsed using Mate Parser with its pre-trained models (Bohnet, 2010; Bohnet and Kuhn, 2012; Seeker and Kuhn, 2013). To keep our English Wikipedia dataset to a manageable size, we choose an older (2006), smaller dump. Long sentences are removed, and the remaining sentences are POS-tagged and dependency parsed using the pre-trained Stanford Parser (Klein and Manning, 2003; de Marneffe et al., 2006). The resulting sizes of the datasets are (roughly): German: 389M words, 28M sentences; Spanish: 147M words, 12M sentences; English: 253M words, 15M sentences. Dependencies extracted from these parsed datasets serve as training for the SIM system described in section 3.3.

Google Books Syntactic N-grams For English, we also obtained dependency relationships for our word similarity statistics using the arcs dataset of the Google Books Syntactic N-Grams (Goldberg and Orwant, 2013), which has 919M items, each of which is a small "syntactic n-gram", a term Goldberg and Orwant use to describe short dependency chains, each of which may contain several tokens. This data set does not contain the actual parses of books from the Google Books corpus, but counts of these dependency chains. We converted the longer chains into their component $(head, dependent, label)$ triples and then collated these triples into counts, also for use in the SIM system.

3 System Design

As previously mentioned, at run-time, our system decomposes the fragment translation task into two parts: generating many possible candidate translations, then scoring and ranking them in the target-language context.

3.1 Constructing Candidate Translations

As a starting point, we use phrase tables constructed in typical SMT fashion, built with the training scripts packaged with Moses (Koehn et al., 2007). These scripts preprocess the bitext, estimate word alignments with GIZA++ (Och and Ney, 2000) and then extract phrases with the `grow-diag-final-and` heuristic.

At translation time, we look for the given source-language phrase in the phrase table, and if it is found, we take all translations of that phrase as our candidates.

When translating a phrase that is not found in the phrase table, we try to construct a "synthetic phrase" out of the available components. This is done by listing, combinatorially, all ways to decompose the L1 phrase into sub-phrases of at least one token long. Then for each decomposition of the input phrase, such that all of its components can be found in the phrase table, we generate a translation by concatenating their target-language sides. This approach naively assumes that generating valid L2 text requires no reordering of the components. Also, since there are 2^{n-1} possible ways to split an n-token phrase into subsequences (*i.e.*, each token is either the first token in a new sub-sequence, or it is not), we perform some heuristic pruning at this step, taking only the first 100 decompositions, preferring those built from longer phrase-table entries. Every phrase in the phrase table, including these synthetic phrases, has both a "direct" and "inverse" probability score; for synthetic phrases, we estimate these scores by taking the product of the corresponding probabilities for the individual components.

In the case that an individual word cannot be found in the phrase table, the system attempts to look up the word in BabelNet, estimating the probabilities as uniformly distributed over the available BabelNet entries. Thus, synthetic phrase table entries can be constructed by combining phrases found in the training data and words available in BabelNet.

For the evaluation, in cases where an L1 phrase contained words that were neither in our training data nor BabelNet (and thus were simply out-of-vocabulary for our system), we took the first translation for that phrase, without regard to context, from Google Translate, through the semi-automated Google Docs interface. This approach is not particularly scalable or reproducible, but simulates what a user might do in such a situation.

3.2 Scoring Candidate Translations via a L2 Language Model

To model how well a phrase fits into the L2 context, we score candidates with an n-gram lan-

[1] `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`

guage model (LM) trained on a large sample of target-language text. Constructing and querying a large language model is potentially computationally expensive, so here we use the KenLM Language Model Toolkit and its Python interface (Heafield, 2011). Here our models were trained on the Wikipedia text mentioned previously (without filtering long sentences), with KenLM set to 5-grams and the default settings.

3.3 Scoring Candidate Translations via Dependency-Based Word Similarity

The candidate ranking based on the n-gram language model – while quite useful – is based on very shallow information. We can also rank the candidate phrases based on how well each of the components fits into the L2 context using syntactic information. In this case, the fitness is measured in terms of dependency-based word similarity computed from dependency triples consisting of the the head, the dependent, and the dependency label. We slightly adapted the word similarity measure by Lin (1998):

$$SIM(w_1, w_2) = \frac{2 * c(h, d, l)}{c(h, -, l) + c(-, d, l)} \quad (1)$$

where $h = w_1$ and $d = w_2$ and $c(h, d, l)$ is the frequency with which a particular $(head, dependent, label)$ dependency triple occurs in the L2 corpus. $c(h, -, l)$ is the frequency with which a word occurs as a head in a dependency labeled l with any dependent. $c(-, d, l)$ is the frequency with which a word occurs as a dependent in a dependency labeled l with any head. In the measure by Lin (1998), the numerator is defined as the information of all dependency features that w_1 and w_2 share, computed as the negative sum of the log probability of each dependency feature. Similarly, the denominator is computed as the sum of information of dependency features for w_1 and w_2.

To compute the fitness of a word w_i for its context, we consider a set D of all words that are directly dependency-related to w_i. The fitness of w_i is thus computed as:

$$FIT(w_i) = \frac{\sum_{w_j}^{D} SIM(w_i, w_j)}{|D|} \quad (2)$$

The fitness of a phrase is the average word similarity over all its components. For example, the fitness of the phrase "eat with chopsticks" would be computed as:

$$FIT(\text{eat with chopsticks}) = \frac{FIT(\text{eat}) + FIT(\text{with}) + FIT(\text{chopsticks})}{3} \quad (3)$$

Since we consider the heads and dependents of a target phrase component, these may be situated inside or outside the phrase. Both cases are included in our calculation, thus enabling us to consider a broader, syntactically determined local context of the phrase. By basing the calculation on a single word's head and dependents, we attempt to avoid data sparseness issues that we might get from rare n-gram contexts.

Back-Off Lexical-based dependency triples suffer from data sparsity, so in addition to computing the lexical fitness of a phrase, we also calculate the POS fitness. For example, the POS fitness of "eat with chopsticks" would be computed as follows:

$$FIT(\text{eat/VBG with/IN chopsticks/NNS}) = \frac{FIT(\text{VBG}) + FIT(\text{IN}) + FIT(\text{NNS})}{3} \quad (4)$$

Storing and Caching The large vocabulary and huge number of combinations of our $(head, dependent, label)$ triples poses an efficiency problem when querying the dependency-based word similarity values. Thus, we stored the dependency triples in a database with a Python programming interface (SQLite3) and built database indices on the frequent query types. However, for frequently searched dependency triples, re-querying the database is still inefficient. Thus, we built a query cache to store the recently-queried triples. Using the database and cache significantly speeds up our system.

This database only stores dependency triples and their corresponding counts; the dependency-based similarity value is calculated as needed, for each particular context. Then, these FIT scores are combined with the scores from the phrase table and language model, using weights tuned by MERT.

system	acc	wordacc	oofacc	oofwordacc
run2	0.665	0.722	0.806	0.857
SIM	0.647	0.706	0.800	0.852
nb	0.657	0.717	0.834	0.868

Figure 1: Scores on the test set for English-German; here next-best is CNRC-run1.

system	acc	wordacc	oofacc	oofwordacc
run2	0.633	0.72	0.781	0.847
SIM	0.359	0.482	0.462	0.607
best	0.755	0.827	0.920	0.944

Figure 2: Scores on the test set for English-Spanish; here best is UEdin-run2.

system	acc	wordacc	oofacc	oofwordacc
run2	0.545	0.682	0.691	0.800
SIM	0.549	0.687	0.693	0.800
best	0.733	0.824	0.905	0.938

Figure 3: Scores on the test set for French-English; here best is UEdin-run1.

system	acc	wordacc	oofacc	oofwordacc
run2	0.544	0.679	0.634	0.753
SIM	0.540	0.676	0.635	0.753
best	0.575	0.692	0.733	0.811

Figure 4: Scores on the test set for Dutch-English; here best is UEdin-run1.

3.4 Tuning Weights with MERT

In order to rank the various candidate translations, we must combine the different sources of information in some way. Here we use a familiar log-linear model, taking the $\log$ of each score – the direct and inverse translation probabilities, the LM probability, and the surface and POS SIM scores – and producing a weighted sum. Since the original scores are either probabilities or probability-like (in the range $[0, 1]$), their $\log$s are negative numbers, and at translation time we return the translation (or n-best) with the highest (least negative) score.

This leaves us with the question of how to set the weights for the log-linear model; in this work, we use the ZMERT package (Zaidan, 2009), which implements the MERT optimization algorithm (Och, 2003), iteratively tuning the feature weights by repeatedly requesting n-best lists from the system. We used ZMERT with its default settings, optimizing our system's BLEU scores on the provided development set. We chose, for convenience, BLEU as a stand-in for the word-level accuracy score, as BLEU scores are maximized when the system output matches the reference translations.

4 Experiments

In figures 1-4, we show the scores on this year's test set for running the two variations of our system: *run2*, the version without the SIM extensions, which we submitted for the evaluation, and *SIM*, with the extensions enabled. For comparison, we also include the best (or for English-German, next-best) submitted system. We see here that the use of the SIM features did not improve the performance of the base system, and in the case of English-Spanish caused significant degradation, which is as of yet unexplained, though we suspect difficulties parsing the Spanish test set, as for all of the other language pairs, the effects of adding SIM features were small.

5 Conclusion

We have described our entry for the initial running of the "L2 Writing Assistant" task and explained some possible extensions to our base log-linear model system.

In developing the SIM extensions, we faced some interesting software engineering challenges, and we can now produce large databases of dependency relationship counts for various languages. Unfortunately, these extensions have not yet led to improvements in performance on this particular task. The databases themselves seem at least intuitively promising, capturing interesting information about common usage patterns of the target language. Finding a good way to make use of this information may involve computing some measure that we have not yet considered, or perhaps the insights captured by SIM are covered effectively by the language model.

We look forward to future developments around this task and associated applications in helping language learners communicate effectively.

References

Bernd Bohnet and Jonas Kuhn. 2012. The best of both worlds – A graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 77–87, Avignon, France.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 89–97, Beijing, China.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-06*, Genoa, Italy.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of English books. In *Second Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 241–247, Atlanta, GA.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL-2003*, pages 423–430, Sapporo, Japan.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, Prague, Czech Republic.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *International Conference on Machine Learning (ICML)*, volume 98, pages 296–304.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 440–447, Hong Kong.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July.

Wolfgang Seeker and Jonas Kuhn. 2013. Morphological and syntactic case in statistical dependency parsing. *Computational Linguistics*, 39(1):23–55.

Omar F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.