NLP 2 - Project 2

May 6, 2015

In general terms your task is to

- choose one of the projects below
- implement the models (and improvements)
- evaluate the approach
- write a report

1 Translation table smoothing and out-of-vocabulary translation

Instructor: Philip Schulz.

This project asks you to come up with a way of getting better phrase table estimates. The two major problems of phrase table estimates as they are currently done are 1) that for rare words these estimates may be unreliable as they are based on relative frequencies, and 2) for certain phrases the table may not even contain an entry, effectively rendering that phrase untranslatable.

While not much attention has been paid to these problems in the 1990s and early 2000s, there has recently been a surge of interest in them. Methods for phrase table enhancement mostly come from related areas such as paraphrasing and semantics. Mikolov et al. (2013) propose to learn a linear mapping between vector spaces formed from vocabularies in two languages. Such a mapping then allows to assign a similarity score to words from the two languages. Such a similarity score might be used to smooth the translation table (compare Songyot and Chiang (2014)). Mikolov et al. (2013) also suggest that one might use the linear mapping to find additional translation candidates not stored in the phrase table.

1.1 Task

Implement the method described in Mikolov et al. (2013). In order to estimate reasonable word vectors, use gensim (`https://radimrehurek.com/gensim/`) or another software of your choice. The word vectors should be estimated on the two sides of the Europarl corpus (the choice of language pair is up to you, however, you want to make sure that the data for this language pair is big so as to get good estimates). Then implement the regression approach as described in the paper. Your own contribution will be one of the two below:

1. Smooth lexical translation probabilities: As we have already seen, word-to-word translation probabilities may not always be reliably estimated. Your task would be to evaluate whether the vector similarity scores obtained from your implementation can be used to solve this problem. While there are many possible ways of doing this, the most straightforward is to simply add the similarity score as a feature in the decoder.

2. Find translations for unknown words: The beauty of Mikolov's approach lies in the fact that both vector spaces are derived from monolingual data. Thus, if their assumption that languages are indeed isometric to each other is correct, this should enable us to translate previously unseen words. In this project you would need to train your vectors on bigger monolingual data sets (e.g. Europarl + some of the monolingual corpora from WMT 2015; see `http://www.statmt.org/wmt15/translation-task.html`). Then you would train translation matrix on the words whose translation you are most sure about. This may be the words whose lexical translation probability exceeds a certain threshold or simply the most frequent words in one language and their dictionary translations. Once you have trained the matrix you can apply it to ANY word, also the unknown ones. You are thus guaranteed to find some translation for a word, given that it has occurred in your bigger monolingual corpora. Those translations can then be used in the decoder (it will be your task to define an appropriate translation score for them). **Note:** In order to check the efficacy of this approach it may be worthwhile deliberately exclude some words that are in the test set from the translation model training set. This way you ensure a certain amount of "unknown" words in the test data.

1.2 Evaluation

You will have to evaluate your implementation by translation a test set that has been determined at the start of your project. You will then compare then performance of a run-of-the-mill Moses system against the performance of your system in terms of BLEU score. If you go for the second approach, you might also check word coverage (i.e. quantify what fraction of all words received some translation).

2 Multilingual Part-of-Speech Tagging

Instructor: Stella Frank.

In this project you will experiment with projecting part-of-speech taggers across languages. The main part of the project will be to reimplement Fossum and Abney (2005) (based on Yarowsky and Ngai (2001)) on newer datasets that use (or can be mapped to) the Universal Part of Speech Tagset (Petrov et al., 2012). This will involve creating a part of speech tagger for multiple source languages and then projecting them, in various combinations, onto two different target languages.

This model is fairly simple, so it is easy to extend with new ideas. Implementing and testing such an extension will be worth 1–2 points. Extensions could be either based on other previous work (see e.g. the bibliography at the end of the cross-lingual learning lecture) or on your own ideas. Some suggestions are given below.

The idea of creating a part-of-speech tagger for a target language based on projected tags from a source language is introduced in Yarowsky and Ngai (2001). They estimate a HMM-type bigram tagger for the source tagger in two steps:[1]

1. estimate the lexical tag probabilities $P(t|w)$ by heavily smoothing the projected tags from the alignments
2. estimate tag sequence probabilities $P(t_i|t_{i-1})$ from a subset of the aligned data, namely sentences with projected tags of high confidence.

Fossum and Abney (2005) extend this method to use multiple source languages by combining multiple projected taggers into a consensus tagger, in which the taggers either combine probabilities or do a simple majority vote. They show that multiple taggers are always better than a single source language. You will not be replicating their results exactly (due to different corpora and tagsets), but this improvement should still hold.

Some ideas for extending this model could include:

- try different methods for combining taggers
- combine information from source languages at an earlier stage (e.g. create a combined lexical prior distribution)
- project tags by adding source POS to the alignment model (i.e. estimate $P(f|e, \text{tag})$. Sparsity could be an issue here!
- use dictionary information (i.e. a list of known tags for a target word) — what is a good way of incorporating such constraints?
- etc.

[1]Some of the details are unclear in the paper, so you will have to come up with something sensible. Be sure to document what you do!

Be sure to compare all of your results to the appropriate baselines, i.e. single-source projection and smaller combinations. If possible, compute an upper limit based on a tagger trained on the target language.

2.1 Alignment

If possible, use your aligner from Project 1; otherwise you can use GIZA++[2], the Berkeley Aligner[3], or Chris Dyer's fast_align[4].

2.2 Languages

You must run your experiments with at least two target languages that are typologically different (i.e., not French and Spanish).

Likewise, experimenting with different kinds of source languages (of which you should have at least 3) is encouraged: does language relatedness to the target help? Is a wide variety of typologically different languages better than very similar languages?

2.3 Data and Taggers

You can use whatever datasets you like, as long as they are appropriate for the task. Here are some initial pointers to datasets freely available online.

For parallel data, you could use the Europarl[5] and the WMT tasks datasets[6].

For training and testing the POS taggers, you will need annotated data.

The Universal Dependency Treebank project [7] is a source of data that has been tagged (and parsed) with a unified tagset, making the task much easier. You can find the datasets here:
`https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1464`

The CoNNL-X shared dependency parsing task [8] is also a potential source, though you may have to map the tagsets to UPOS.

The `nltk` package also has support for the UPOS tagset.

[2]`https://code.google.com/p/giza-pp/`
[3]`https://code.google.com/p/berkeleyaligner/`
[4]`https://github.com/clab/fast_align`
[5]`http://www.statmt.org/europarl/`
[6]`http://www.statmt.org/wmt15/translation-task.html`
[7]`http://universaldependencies.github.io`
[8]`http://ilk.uvt.nl/conll/post_task_data.html`

3 Reranking PETs

Instructor: Miloš Stanojević.

PETs (Permutation Trees) Zhang and Gildea (2007) are a compact way to represent reordering patterns between two languages. The recognition of these trees has many parallels to the recognition of standard phrase structure parse trees so applying techniques that already exist in parsing world comes very natural.

State of the art phrase structure parsers usually consist of two components:

1. Usage of simpler model that uses dynamic programming (CKY) to find n-best parse trees
2. These n-best trees are then rescored using some more powerful discriminative model with global features

The goal of this project will be to implement a second (reranking) component for reranking PETs. As input students will be given n-best PETs in Penn treebank format for each sentence in the training set and the test set (English-Japanese language pair). They will learn reranking model on the training set to maximize some reordering metric (for example Kendall τ) and then apply that model to rerank n-best lists in test set.

Learning-to-rank is a machine learning framework that is used for these kinds of ranking tasks. Students can take any learning algorithm that exists in that framework and use any library they find suitable for solving that task. Some options are:

- RankLib `http://people.cs.umass.edu/~vdang/ranklib.html` – contains lots of different learning algorithms from simple and fast (e.g. linear regression) to powerful but slow (e.g. neural networks for ranking),
- Weka `http://www.cs.waikato.ac.nz/ml/weka/` which does not support ranking in general but can easily be extended for that using method described in Stanojević and Sima'an (2014).

The core part of this project is in engineering the right features for reranking permutation trees. Since original PETs are extracted in unsupervised way they do not contain much of linguistic intuition so having some features of that sort might be helpful. An example of potentially useful feature could be an indicator of whether there is an *inversion node* when the span is a verb phrase (unlike English, Japanese usually puts head (verb) after its complement (object)).

There will be also cases which are harder to solve. For example, what to do when a constituent in PET tree is not a constituent in syntactic tree? Previous work used more complex labels similar to CCG, but maybe better solutions could be found.

Any kind of feature is allowed. Students can use dependency trees, constituency trees, semantic roles or whatever they consider useful for finding the right permutation tree.

4 Learning linear ordering problems

Instructor: Wilker Aziz.

There is a lot of evidence in the literature that we may be able to separate reordering decisions from lexical substitution in translation (Tromble and Eisner, 2009; Dyer and Resnik, 2010; Khalilov and Sima'an, 2011; DeNero and Uszkoreit, 2011; Neubig et al., 2012). In this project you will experiment with learning a model of *preordering*, that is, a model that tries to predict a permutation of the input sentence which is close to target language word-order. In other words, you would like to preprocess input sentences so that they are as suitable as possible to monotone (or near monotone) translation.[9] Reasoning about arbitrary permutations is an NP-complete problem, thus you will focus on binarizable permutations (Wu, 1997).

You will reimplement the model proposed by Tromble and Eisner (2009). The authors of this model propose to learn a pairwise preference matrix whose entries are a linear combination of rich contextual features associated with each possible pair of input words in a sentence. All binary bracketings of a sequence (and in fact, all binarizable permutations) can be efficiently packed in a CKY chart. The authors introduce a recursive formula similar to the INSIDE algorithm which decomposes the benefit of swapping any pair of adjacent subsequences of the input in terms of the entries of the preference matrix. At training, "gold-standard" permutations are extracted from an automatically word-aligned parallel corpus (they make your "labels"). The parameters of the linear model are estimated using an average perceptron algorithm. For each training instance, a perceptron update is the result of comparing the feature representation of that instance's gold-standard permutation with the feature representation of the best binarizable permutation under the current linear model.

Implementing and testing an extension to this model will be worth an extra point. There are several opportunities for improvements depending on where your interests lie. A few suggestions are:

1. replace optimisation by sampling and introduce a permutation distance metric in decoding (e.g. Kendall's tau);
2. consider a larger space of permutations (i.e. including permutations that are not binarizable – note that the authors propose to iterate their algorithm in order to achieve that

Finally, you will evaluate your model in terms of the permutations it produces (the authors compare predictions to gold-standard using BLEU). Again, you can earn an extra point if you evaluate the performance of a phrase-based MT system (e.g. Moses) using preordered text as produced by your model (and variants).

[9]Note how appealing this is to recent approaches based on sequence-to-sequence learning (Sutskever et al., 2014) (particularly those based on neural networks).

This project is an opportunity to get your hands dirty with important algorithms (e.g. CKY, Inside, Viterbi, perceptron) and, depending on how far you choose to go with the evaluation, it may become an opportunity to experiment with a complete MT pipeline.[10]

References

DeNero, J. and Uszkoreit, J. (2011). Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Dyer, C. and Resnik, P. (2010). Context-free reordering, finite-state translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 858–866, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fossum, V. and Abney, S. (2005). Automatically inducing a part-of-speech tagger by projecting from multiple source languages across aligned corpora. In *Proceedings of IJCNLP*.

Khalilov, M. and Sima'an, K. (2011). Context-sensitive syntactic source-reordering by statistical transduction. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 38–46, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013). Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Neubig, G., Watanabe, T., and Mori, S. (2012). Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 843–853, Jeju Island, Korea. Association for Computational Linguistics.

Petrov, S., Das, D., and McDonald, R. (2012). A universal part-of-speech tagset. In *LREC*.

Songyot, T. and Chiang, D. (2014). Improving word alignment using word similarity. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1840–1845, Doha, Qatar. Association for Computational Linguistics.

[10]Because this project is technically very demanding, unless you would like to implement it yourself, we can provide you with a friendly python implementation of chart parsing. The idea here is for you to focus on aspects of learning and evaluation.

Stanojević, M. and Sima'an, K. (2014). BEER: BEtter Evaluation as Ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Tromble, R. and Eisner, J. (2009). Learning linear ordering problems for better translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 1007–1016, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Yarowsky, D. and Ngai, G. (2001). Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of NAACL*.

Zhang, H. and Gildea, D. (2007). Factorization of synchronous context-free grammars in linear time. In *In NAACL Workshop on Syntax and Structure in Statistical Translation (SSST*.