Bilingual Lexicon Induction from Monolingual Sources for Low Resource Languages

Abstract

State of the art statistical machine translation methods rely on the availability of substantial amounts of human translated texts (*bitexts*) for parameter estimation. Such bilingual resources are available for few language pairs, and the expense associated with generating them is a fundamental obstacle to extending statistical MT for low-resource languages. In this work, we exploit a set of cues present in plentiful monolingual data for the task of bilingual dictionary induction. In exploting cross-lingual, phrasal distributional statistics as one such cue, we employ a novel, space-efficient extension to the *locality sensitive hashing* (LSH) scheme.

We give results for an extensive number of languages (paired with English), and show the efficacy of each cue in the context of a language pair where plentiful bilingual resources are available.

1 Introduction

Statistical methods for machine translation continue to push the state of the art in automatic translation. However, they crucially rely on the availability of large numbers of translations aligned across two languages. Generation of these parallel corpora require the efforts of bilingual speakers and are extremely expensive to produce in sufficient quantities to induce a high quality statistical translation system. As a result, these methods can not be successfully applied to the majority of word's languages and especially those less frequently taught. In terms of the community's evaluation of progress, most shared tasks involve european languages for which generous quantities of multilingual parliament proceedings are available.

At the same time we now have unprecedented access to vast and continually expanding monolingual resources. Moreover, they often contain additional metadata which can provide non-sentential cues for inducing bilingual resources; suggesting we might substantially reduce and eventually eliminate the requirement for explicitly aligned bilingual translations. Recent examples include exploiting temporal information to induce Named Entity lexicons ((**?**; **?**)), and topic information to generate translations ((**?**)). Moreover, resources such as these are likely to be extremely useful for numerous multilingual NLP tasks ().

Examples

2 Inducing Bilingual Lexicons

Our goal is to generate bilingual resources for pairs of languages for which we have sufficient monolingual data. These languages currently include .

List languages

This article consitutes the first report on this effort: since more data is continuously becoming available, we expect to add more languages to this list in the future.

Cues and similarity metrics:

- Context (including contextual NEs), using dependency contexts for the resources rich side (i.e. English).
- Time
- Topics (i.e. wiki categories)
- Edit distance

Combination strategies:

- cue scores as classification features: use seed dictionaries for supervised data.
- rank aggregation

3 Contextual Similarity

Grouping semantically related words through distributional statistics (e.g. (Pereira et al., 1993)) is a

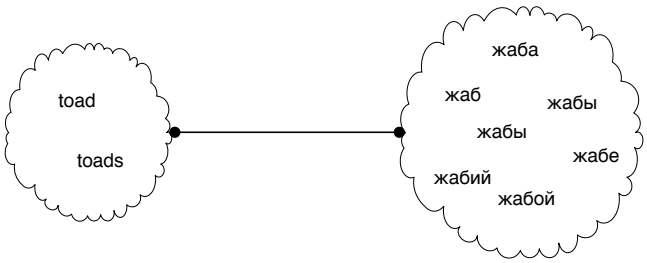

Figure 1: Inducing lexicons using contextual similarity. An example set of word forms for *toad* for English and Russian.

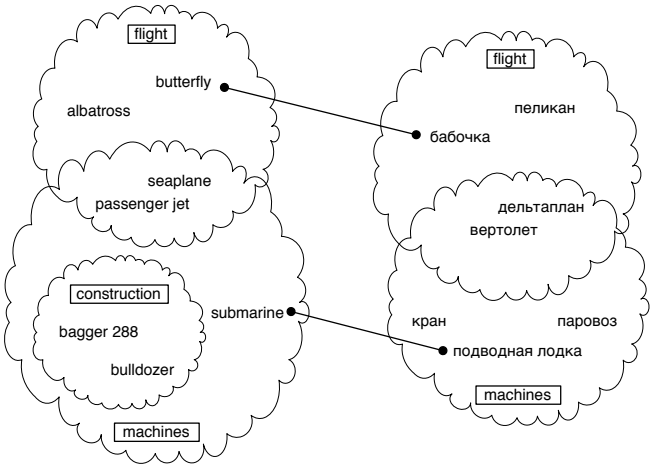

Figure 2: Hierarchical clustering: associations can be inferred between sets of semantically related words. Intersections between sets e.g. "flying machines") can also substantially limit the sets of potential translation candidates.

way to reduce both the search space and associate translations. We can find associations between clusters instead of individual lexemes or phrases, by intersecting them we can substantially reduce the space of possible translations (Fig. 2). .

Verbiage is very close to CCB's proposal

4 Experimental Evaluation

Evaluation: tokens for inducing translations. Evaluation metrics: why precision at top-k?

4.1 Data and Other Resources

Describe the data:

- Wiki
- News

Describe the resources:

- Dictionaries: generally, noisy

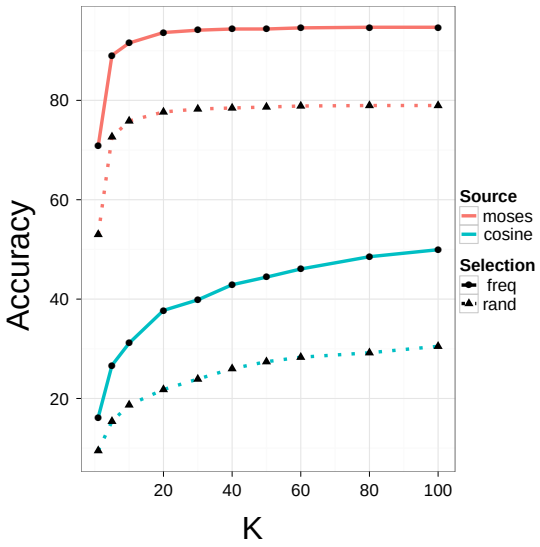

Figure 3: Moses as compared to cosine.

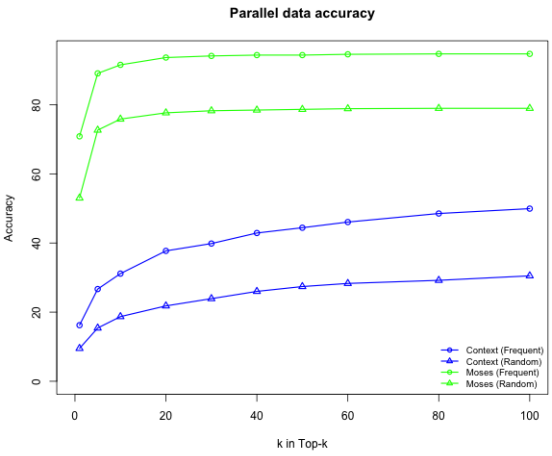

Figure 4: Bilingual lexicons from Moses lexical tables vs. using contextual cues.

- Parallel data: some languages may have small amounts of parallel data

4.2 Quality of Available Resources

Parallel data experiments:

- Moses lexical tables vs. monolingual cues (e.g., Fig. 4).
- Use Moses lexical tables as seed dictionaries.

4.3 Quality of Individual Cues

Performance of individual cues per language.

4.4 Combination strategies

Classification and rank aggregation.

Space Efficient Online LSH

Feature vectors based on lexical features are often of a high dimension. This leads to $O(d)$ operations to calculate cosine similarity. This is improved in practice through the use of data structures that exploit feature sparsity, leading to an expected $O(f)$ operations, where f is the typical number of unique features we expect to have non-zero entrues in any given vector.

Ravichandran et al. (2005) showed that the *locality sensitive hash* procedure of Charikar (2002) (following from Indyk and Motwani (1998) and Goemans and Williamson (1995)), could be used in the context of lexical feature vectors to acheive speed efficiencies in approximate cosine-similarity computation. Constructing such LSH signatures involves the following hash function, where $\vec{v} \in \mathbb{R}^{d'}$ is a vector in the original feature space, and $\vec{r}_i$ is a random vector drawn from $N(0,1)^{d'}$:

$$h_i(\vec{v}) = \begin{cases} 1 & \text{if } \vec{v}^{\mathrm{T}}\vec{r}_i \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

If we let $h^d(\vec{v})$ be the length d bit signature resulting from d such hash functions, then the cosine similarity between two vectors $\vec{u}$ and $\vec{v}$ is approximated by:

$$\cos(\vec{u},\vec{v}) = \frac{\vec{u}\cdot\vec{v}}{|\vec{u}||\vec{v}|} \approx \cos(\frac{H(h^d(\vec{u}),h^d(\vec{v}))}{d} * \pi),$$

where $H(\cdot,\cdot)$ is the *hamming distance* between two bit-vectors of equal length. This technique is used when $d << d'$, which leads to faster pairwise comparisons between vectors, and a lower memory footprint.

Van Durme and Lall (2010) recently observed[1] that if the feature values being considered are additive over a dataset (such as when collecting word co-occurrence frequencies), then these signatures may be maintained *online* by unrolling the dot-product into a series of local operations: $\vec{v}^{\mathrm{T}}\vec{r}_i = \Sigma_t \vec{v}_t{}^{\mathrm{T}}\vec{r}_i$, where $\vec{v}_t$ represents features observed locally, at time t in a data-stream.

Van Durme and Lall used this observation to construct LSH signatures in an online fashion, rather than waiting until fully collecting feature observations over an entire corpus. Since updates can be done locally, then feature vectors do not need to be stored explicitly, leading to potentially significant space savings, as only d scalar variables are needed to record d running sums.

[1] A preprint of this article was made available to the authors by request. This point was similarly made by Li et al. (2008) in discussing stable random projections.

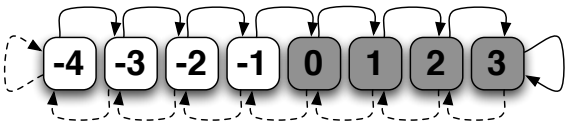

Figure 5: An 8-state SumSign Markov Chain, requiring $\lg(8) = 3$ bits to represent. Light and dark states correspond to whether a given bit of an LSH signature will be set to 0 or 1, respectively. States are numerically labelled in order to reflect the similarity to a small bit integer data type, one that never overflows.

We adopted this technique in our own experiments, while making an additional observation: the scalar variables used to store the running sums may themselves be an inefficient use of space. A d bit signature requires the online storage of $d * 32$ bits of memory (assuming a 32-bit floating point representation per counter), but since the only thing one cares about the resultant sum is its sign, then it might be wasteful to record the actual running total, as compared to maintaining some other data structure whose final state correlates with the sign.

SumSign Markov Chain

Figure 5 gives an example of what we call a SumSign Markov Chain: a linear chain that begins in its center state[2] and transitions to the left or the right in the presence of positive or negative unit updates. When in a left- or right-most state, further steps in the terminal direction will self-loop. In programmatic terms, this structure can be thought of as an integer valued counter with a finite representation, that rejects updates which would cause numerical *overflow* or *underflow*.

Under the assumption that n unit values, $x_t \in \{-1,+1\}$, are drawn *iid*, distributed according to a fixed probablity p, then $\text{sign}(\Sigma_t^n x_t)$ positively correlates with whether the current state of the chain is to the left or the right of its center, when these n values are treated as updates. While deferring analysis to future work, it should be clear that two factors impact the strength of correlation of the sum with the state of the chain: (1) the further p is from 0.5, the stronger the correlation; (2) the longer the chain, the stronger the correlation. Here we restrict ourselves to chains representable by bit

[2] In the case of an even length chain, initialization can be choosen randomly from the two center states.

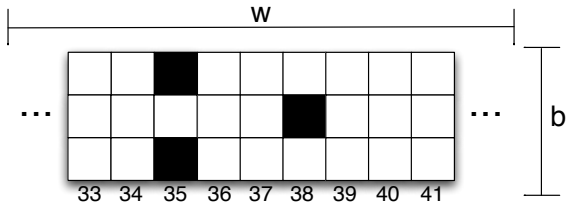

Figure 6: A Spectral Bloom Counter of width w, with $b = 3$ bits per cell. Locations 35 and 38 respectively contain the bit sequences 101 and 010.

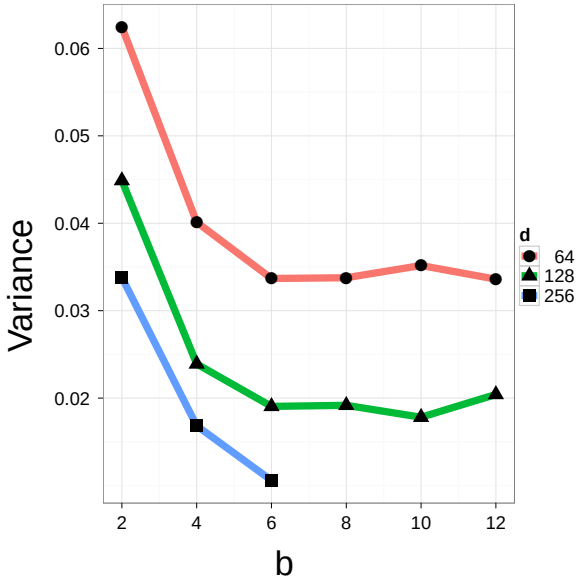

Figure 7: Observed variance in the approximated cosine similarity for 100,000 randomly sampled pairs, as a function of the number of bits used for each local counter.

sequences, and thus the number of available states grows exponentially with the number of available bits, b (length = 2^b).

As applied to LSH signature generation, here in practice we will convert each entry $y \in r_i$ into a sequence of $\lfloor |y| \rfloor$ unit updates of the same sign, plus one additional update conditional on whether a $[0, 1]$ uniformly distributed random draw is less than $|\text{remainder}(y)|$.

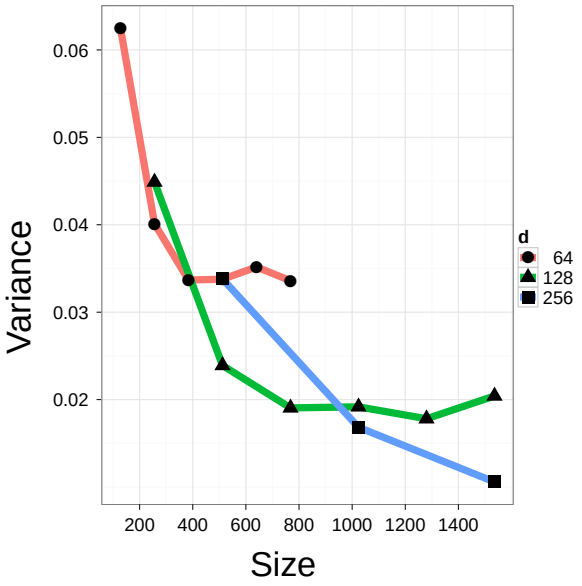

Figure 8: Observed variance in the approximated cosine similarity for 100,000 randomly sampled pairs, as a function of total number of bits ($d * b$) used to compute each individual signature.

5 Related Work

- Context: (**?**; **?**; **?**)
- Time: (**?**; Klementiev and Roth, 2006)
- Topics: (**?**; **?**)
- Multiple: (**?**; **?**; **?**)
- Dependecies: (**?**)
- Bridge languages: (**?**)
- Combination Strategies: (**?**; Klementiev and Roth, 2006; **?**)
- Mechanical Turk: Our NAACL workshop paper.
- Other: (**?**)

6 Conclusions and Future Work

Using cues for MT.

References

Moses Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of STOC*.

Michel X. Goemans and David P. Williamson. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *JACM*, 42:1115–1145.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of STOC*.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ping Li, Kenneth W. Church, and Trevor J. Hastie. 2008. One Sketch For All: Theory and Application of Conditional Random Sampling. In *Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS)*.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Benjamin Van Durme and Ashwin Lall. 2010. Online Generation of Locality Sensitive Hash Signatures. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.