The Prague Bulletin of Mathematical Linguistics
NUMBER 100 OCTOBER 2013 51–62

morphogen: Translation into Morphologically Rich Languages with Synthetic Phrases

Eva Schlinger, Victor Chahuneau, Chris Dyer

Language Technologies Institute, Carnegie Mellon University

Abstract

We present morphogen, a tool for improving translation into morphologically rich languages with synthetic phrases. We approach the problem of translating into morphologically rich languages in two phases. First, an inflection model is learned to predict target word inflections from source side context. Then this model is used to create additional sentence specific translation phrases. These "synthetic phrases" augment the standard translation grammars and decoding proceeds normally with a standard translation model. We present an open source Python implementation of our method, as well as a method of obtaining an unsupervised morphological analysis of the target language when no supervised analyzer is available.

1. Introduction

Machine translation into morphologically rich languages is challenging, due to lexical sparsity on account of grammatical features being expressed with morphology. In this paper, we present an open-source Python tool, morphogen, that leverages target language morphological grammars (either hand-crafted or learned unsupervisedly) to enable prediction of highly inflected word forms from rich, source language syntactic information.[1]

Unlike previous approaches to translation into morphologically rich languages, our tool constructs sentence-specific translation grammars (i.e., phrase tables) for each sentence that is to be translated, but then uses a standard decoder to generate the final

[1] https://github.com/eschling/morphogen

Cite as: Eva Schlinger, Victor Chahuneau, Chris Dyer. morphogen: Translation into Morphologically Rich Languages with Synthetic Phrases. The Prague Bulletin of Mathematical Linguistics No. 100, 2013, pp. 51–62. doi: 10.2478/pralin-2013-0011.

translation with no post-processing. The advantages of our approach are: (i) newly synthesized forms are highly targeted to a specific translation context; (ii) multiple alternatives can be generated with the final choice among rules left to a standard sentence-level translation model; (iii) our technique requires virtually no language-specific engineering; and (iv) we can generate forms that were not observed in the bilingual training data.

This paper is structured as follows. We first describe our "translate-and-inflect" model that is used to synthesize the target side of lexical translations rule given its source and its source context (§2). This model discriminates between inflectional options for predicted stems, and the set of inflectional possibilities is determined by a morphological grammar. To obtain this morphological grammar, the user may either provide a morphologically analyzed version of their target language training data, or a simple unsupervised morphology learner can be used instead (§3). With the morphologically analyzed parallel data, the parameters of the discriminative model are trained from the complete parallel training data using an efficient optimization procedure that does not require a decoder.

At test time, our tool creates **synthetic phrases** representing likely inflections of likely stem translations for each sentence (§4). We briefly present the results of our system on English–Russian, –Hebrew, and –Swahili translation tasks (§5), and then describe our open source implementation, and discuss how to use it with both user-provided morphological analyses and those of our unsupervised morphological analyzer[2] (§6).

2. Translate-and-Inflect Model

The task of the translate-and-inflect model is illustrated in Figure 1 for an English–Russian sentence pair. The input is a sentence $\mathbf{e}$ in the source language[3] together with any available linguistic analysis of $\mathbf{e}$ (e.g., its dependency parse). The output $\mathbf{f}$ consists of (i) a sequence of stems, each denoted σ, and (ii) one morphological inflection pattern for each stem, denoted μ.[4] Throughout, we use Ω_σ to denote the set of possible morphological inflection patterns for a given stem σ. Ω_σ might be

[2]Further documentation is available in the `morphogen` repository.

[3]In this paper, the source language is always English. We use $\mathbf{e}$ to denote the source language (rather than the target language), to emphasize the fact that we are translating *from* a morphologically impoverished language *to* a morphologically rich one.

[4]When the information is available from the morphological analyzer, a stem σ is represented as a tuple of a lemma and its inflectional class.

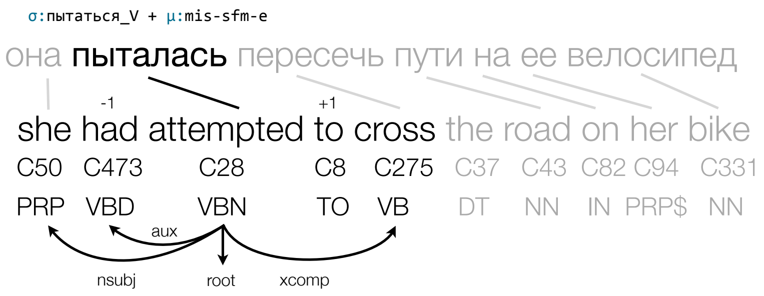
σ:пытаться_V + μ:mis-sfm-e
она пыталась пересечь пути на ее велосипед
-1
+1
she had attempted to cross the road on her bike
C50 C473 C28 C8 C275 C37 C43 C82 C94 C331
PRP VBD VBN TO VB DT NN IN PRP$ NN
aux
nsubj
root
xcomp

Figure 1. The inflection model predicts a form for the target verb stem based on its source attempted *and the linear and syntactic source context. The inflection pattern* `mis-sfm-e` *(*main+indicative+past+singular+feminine+medial+perfective*) is that of a supervised analyzer.*

defined by a grammar; our models restrict Ω_σ to be the set of inflections observed anywhere in our monolingual or bilingual training data as a realization of σ.[5]

We define a probabilistic model over target words $\mathbf{f}$. The model assumes independence between each target word f conditioned on the source sentence $\mathbf{e}$ and its aligned position i in this sentence.[6] This assumption is further relaxed in §4 when the model is integrated in the translation system. The probability of generating each target word f is decomposed as follows:

$$p(f \mid \mathbf{e}, i) = \sum_{\sigma\star\mu=f} \underbrace{p(\sigma \mid e_i)}_{\text{gen. stem}} \times \underbrace{p(\mu \mid \sigma, \mathbf{e}, i)}_{\text{gen. inflection}}.$$

Here, each stem is generated independently from a single aligned source word e_i, but in practice we use a standard phrase-based model to generate sequences of stems and only the inflection model operates word-by-word.

2.1. Modeling Inflection

In morphologically rich languages, each stem may be combined with one or more inflectional morphemes to express different grammatical features (e.g., case, definiteness, etc.). Since the inflectional morphology of a word generally expresses multiple features, we use a model that uses overlapping features in its representation of both

[5]This is a practical decision that prevents the model from generating words that would be difficult for a closed-vocabulary language model to reliably score. When open-vocabulary language models are available, this restriction can easily be relaxed.

[6]This is the same assumption that Brown et al. (1993) make in, for example, IBM Model 1.

$$\left\{\begin{array}{c}\text{source aligned word } e_i\\ \text{parent word } e_{\pi_i} \text{ with its dependency } \pi_i \to i\\ \text{all children } e_j \mid \pi_j = i \text{ with their dependency } i \to j\\ \text{source words } e_{i-1} \text{ and } e_{i+1}\end{array}\right\} \left\{\begin{array}{c}\text{token}\\ \text{part-of-speech tag}\\ \text{word cluster}\end{array}\right\}$$

$$\begin{array}{c}\text{– are } e_i, e_{\pi_i} \text{ at the root of the dependency tree?}\\ \text{– number of children, siblings of } e_i\end{array}$$

Table 1. Source features $\varphi(\boldsymbol{e}, i)$ extracted from $\boldsymbol{e}$ and its linguistic analysis. π_i denotes the parent of the token in position i in the dependency tree and $\pi_i \to i$ the typed dependency link.

the input (i.e., conditioning context) and output (i.e., the inflection pattern):

$$p(\mu \mid \sigma, \boldsymbol{e}, i) = \frac{\exp\left[\boldsymbol{\varphi}(\boldsymbol{e}, i)^\top \boldsymbol{W} \boldsymbol{\psi}(\mu) + \boldsymbol{\psi}(\mu)^\top \boldsymbol{V} \boldsymbol{\psi}(\mu)\right]}{\sum_{\mu' \in \Omega_\sigma} \exp\left[\boldsymbol{\varphi}(\boldsymbol{e}, i)^\top \boldsymbol{W} \boldsymbol{\psi}(\mu') + \boldsymbol{\psi}(\mu')^\top \boldsymbol{V} \boldsymbol{\psi}(\mu')\right]}. \qquad (1)$$

Here, $\boldsymbol{\varphi}$ is an m-dimensional *source context* feature vector function, $\boldsymbol{\psi}$ is an n-dimensional *morphology* feature vector function, $\boldsymbol{W}$ is an $m \times n$ parameter matrix, and $\boldsymbol{V}$ is an $n \times n$ parameter matrix. In our implementation, $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$ return sparse vectors of binary indicator features, but other features can easily be incorporated.

2.2. Source Contextual Features: $\varphi(\boldsymbol{e}, i)$

In order to select the best inflection of a target-language word, given the source word it translates from and the *context* of that source word, we seek to leverage numerous features of the context to capture the diversity of possible grammatical relations that might be encoded in the target language morphology. Consider the example shown in Figure 1, where most of the inflection features of the Russian word (past tense, singular number, and feminine gender) can be inferred from the context of the source word it is aligned to. To access this information, our tool uses parsers and other linguistic analyzers.

By default, we assume that English is the source language and provide wrappers for external tools to generate the following linguistic analyses of each input sentence:

- Part-of-speech tagging with a CRF tagger trained on sections 02–21 of the Penn Treebank,
- Dependency parsing with TurboParser (Martins et al., 2010), and
- Mapping of the tokens to one of 600 Brown clusters trained from 8B words of English text.[7]

[7]The entire monolingual data available for the translation task of the 8th ACL Workshop on Statistical Machine Translation was used. These clusters are available at `http://www.ark.cs.cmu.edu/cdyer/en-c600.gz`

From these analyses we then extract features from $\mathbf{e}$ by considering the aligned source word e_i, its preceding and following words, and its dependency neighbors. These are detailed in Table 1 and can be easily modified to include different features or for different source languages.

3. Morphological Grammars and Features

The discriminative model in the previous section selects an inflectional pattern for each candidate stem. In this section, we discuss where the inventory of possible inflectional patterns it will consider come from.

3.1. Supervised Morphology

If a target language morphological analyzer is available that analyses each word in the target of the bitext and monolingual training data into a stem and vector of grammatical features, the inflectional vector may be used directly to define $\boldsymbol{\psi}(\mu)$ by defining a binary feature for each key-value pair (e.g., `Tense=past`) composing the tag. Prior to running `morphogen`, the full monolingual and target side bilingual training data should be analyzed.

3.2. Unsupervised Morphology

Supervised morphological analyzers that map between inflected word forms and abstract grammatical feature representations (e.g., PAST+SING) are not available for every language into which we might seek to translate. We therefore provide an *unsupervised* model of morphology that segments words into sequences of morphemes, assuming a concatenative generation process and a single analysis per type. To do so, we assume that each word can be decomposed into any number of prefixes, a stem, and any number of suffixes. Formally, we let M represent the set of all possible morphemes and define a regular grammar M^*MM^* (i.e., zero or more prefixes, a stem, and zero or more suffixes). We learn weights for this grammar by assuming that the probability of each prefix, stem, and suffix is given by a draw from a Dirichlet distribution over all morphemes and then inferring the most likely analysis.

Hyperparemeters. To run the unsupervised analyzer, it is necessary to specify the Dirichlet hyperparameters $(\alpha_p, \alpha_s, \alpha_t)$ which control the sparsity of the inferred prefix, stem, and suffix lexicons, respectively. The learned morphological grammar is (rather unfortunately) very sensitive to these settings, and some exploration is necessary. As a rule of thumb, we observe that $\alpha_p, \alpha_s \ll \alpha_t \ll 1$ is necessary to recover useful segmentations, as this encodes that there are many more possible stems than inflectional affixes; however the absolute magnitude will depend on a variety of factors. Default values are $\alpha_p = \alpha_s = 10^{-6}, \alpha_t = 10^{-4}$; these may be adjusted by factors of 10 (larger to increase sparsity; smaller to decrease it).

Unsupervised morphology features: $\psi(\mu)$ For the unsupervised analyzer, we do not have a mapping from morphemes to grammatical features (e.g., `+past`); however, we can create features from the affix sequences obtained after morphological segmentation. We produce binary features corresponding to the content of each potential affixation position relative to the stem. For example, the unsupervised analysis `wa+ki+wa+STEM` of the Swahili word *wakiwapiga* will produce the following features:

```
Prefix[-3][wa] Prefix[-2][ki] Prefix[-1][wa].
```

3.3. Inflection Model Parameter Estimation

From the analyzed parallel corpus (source side syntax and target side morphological analysis), `morphogen` sets the parameters $\boldsymbol{W}$ and $\boldsymbol{V}$ of the inflection prediction model (Eq. 1) using stochastic gradient descent to maximize the conditional log-likelihood of a training set consisting of pairs of source sentence contextual features ($\boldsymbol{\varphi}$) and target word inflectional features ($\boldsymbol{\psi}$). The training instances are word alignment pairs from the full training corpus. When morphological category information is available, an independent model may be trained for each open-class category (e.g., nouns, verbs); but, by default a single model is used for all words (excluding words shorter than a minimum length).

It is important to note here that our richly parameterized model is trained on the *full parallel training corpus*, not just on the small number of development sentences. This is feasible because, in contrast to standard discriminative translation models which seek to discriminate good complete translations from bad complete translations, `morphogen`'s model must only predict how good each possible *inflection* of an independently generated stem is. All experiments reported in this paper used models trained on a single processor using a Cython implementation of the SGD optimizer.[8]

4. Synthetic Phrases

How is `morphogen` used to improve translation? Rather than using the translate-and-inflect model directly to perform translation, we use it just to augment the set of rules available to a conventional hierarchical phrase-based translation model (Chiang, 2007; Dyer et al., 2010). We refer to the phrases it produces as **synthetic phrases**. The aggregate grammar consists of both synthetic and "default" phrases and is used by an unmodified decoder.

The process works as follows. We use the suffix-array grammar extractor of Lopez (2007) to generate sentence-specific grammars from the fully inflected version of the training data (the default grammar) and also from the stemmed variant of the training

[8]For our largest model, trained on 3.3M Russian words, $n = 231\text{K} * m = 336$ feature were produced, and 10 SGD iterations at a rate of 0.01 were performed in less than 16 hours.

Russian supervised	**Hebrew**	**Swahili**
Verb: 1st Person	Suffix ים (masculine plural)	Prefix *li* (past)
`child(nsubj)=I child(nsubj)=we`	`parent=NNS after=NNS`	`source=VBD source=VBN`
Verb: Future tense	Prefix א (first person sing. + future)	Prefix *nita* (1st person sing. + future)
`child(aux)=MD child(aux)=will`	`child(nsubj)=I child(aux)='ll`	`child(aux) child(nsubj)=I`
Noun: Animate	Prefix כ (preposition like/as)	Prefix *ana* (3rd person sing. + present)
`source=animals/victims/...`	`child(prep)=IN parent=as`	`source=VBZ`
Noun: Feminine gender	Suffix י (possesive mark)	Prefix *wa* (3rd person plural)
`source=obama/economy/...`	`before=my child(poss)=my`	`before=they child(nsubj)=NNS`
Noun: Dative case	Suffix ה (feminine mark)	Suffix *tu* (1st person plural)
`parent(iobj)`	`child(nsubj)=she before=she`	`child(nsubj)=she before=she`
Adjective: Genitive case	Prefix כש (when)	Prefix *ha* (negative tense)
`grandparent(poss)`	`before=when before=WRB`	`source=no after=not`

Figure 2. Examples of highly weighted features learned by the inflection model. We selected a few frequent morphological features and show their top corresponding source context features.

data (the stemmed grammar). We then extract a set of translation rules that only contain terminal symbols (sometimes called "lexical rules") from the stemmed grammar. The (stemmed) target side of each such phrase is then *re-inflected* using the inflection model described above (§2), conditioned on the source sentence and its context. Each stem is given its most likely inflection. The resulting rules are added to the default grammar for the sentence to produce the aggregate grammar.

The standard translation rule features present on the stemmed grammar rules are preserved, and `morphogen` adds the following features to help the decoder select good synthetic phrases: (i) a binary feature indicating that the phrase is synthetic; (ii) the log probability of the inflected form according to the inflection model; and (iii) if available, counts of the morphological categories inflected.

5. Experiments

We briefly report in this section on some experimental results obtained with our tool. We ran experiments on a 150k sentence Russian–English task (WMT2013; news-commentary), a 134k sentence English–Hebrew task (WIT^3 TED talks corpus), and a 15k sentence English–Swahili Task. Space precludes a full discussion of the performance of the classifier,[9] but we can also inspect the weights learned by the model to assess the effectiveness of the features in relating source-context structure with target-side morphology. Such an analysis is presented in Figure 2.

[9]We present our approach and the results of both the intrinsic and extrinsic evaluations in much more depth in Chahuneau et al. (in review)

	EN→RU	EN→HE	EN→SW
Baseline	14.7±0.1	15.8±0.3	18.3±0.1
+Class LM	15.7±0.1	16.8±0.4	18.7±0.2
+Synthetic			
unsupervised	16.2±0.1	17.6±0.1	19.0±0.1
supervised	16.7±0.1	—	—

Table 2. Translation quality (measured by bleu*) averaged over 3 MIRA runs.*

5.1. Translation

We evaluate our approach in the standard discriminative MT framework. We use cdec (Dyer et al., 2010) as our decoder and perform MIRA training (Chiang, 2012) to learn feature weights. We compare the following configurations:

- A baseline system, using a 4-gram language model trained on the entire monolingual and bilingual data available.
- An enriched system with a class-based n-gram language model[10] trained on the monolingual data mapped to 600 Brown clusters. Class-based language modeling is a strong baseline for scenarios with high out-of-vocabulary rates but in which large amounts of monolingual target-language data are available.
- The enriched system further augmented with our inflected synthetic phrases. We expect the class-based language model to be especially helpful here and capture some basic agreement patterns that can be learned more easily on dense clusters than from plain word sequences.

We evaluate translation quality by translating and measuring BLEU on a held-out evaluation corpus, averaging the results over 3 MIRA runs (Table 2). For all languages, using class language models improves over the baseline. When synthetic phrases are added, significant additional improvements are obtained. For the English–Russian language pair, where both supervised and unsupervised analyses can be obtained, we notice that expert-crafted morphological analyzers are more efficient at improving translation quality.

6. Morphogen Implementation Discussion and User's Guide

This section describes the open-source Python implementation of this work, `morphogen`.[11] Our decision to use Python means the code—from feature extraction to grammar processing—is generally readable and simple to modify for research purposes. For example, with few changes to the code, it is easy to expand the number of

[10]For Swahili and Hebrew, $n = 6$; for Russian, $n = 7$.

[11]`https://github.com/eschling/morphogen`

synthetic phrases created by generating k-best inflections (rather than just the most probable inflection), or to restrict the phrases created based on some source side criterion such as type frequency, POS type, or the like.

Since there are many processing steps that must be coordinated to run `morphogen`, we provide reference workflows using `ducttape`[12] for both supervised and unsupervised morphological analyses (discussed below). While these workflows are set up to be used with `cdec`, `morphogen` generates grammars that could be used with any decoder that supports per-sentence grammars. The source language processing, which we do for English using TurboParser and TurboTagger, could be done with any tagger and any parser that can produce basic Stanford dependencies. The source language does not necessarily need to be English, although our approach depends on having detailed source side contextual information.[13]

We now review the steps that must be taken to run `morphogen` with either an external (generally supervised) morphological analyzer or the unsupervised morphological analyzer we described above. These steps are implemented in the provided `ducttape` workflows.

Running `morphogen` with an external morphological analyzer. If a supervised morphological analyzer is used, the parallel training data must be analyzed on the target side, with each line containing four fields (source sentence, target sentence, target stem sentence, target analysis sequence), where fields are separated with the triple pipe (`|||`) symbol. Target language monolingual data must likewise be analyzed and provided in a file where each line contains three fields (sentence, stem sentence, analysis sequence) and separated by triple pipes. For supervised morphological analyses, the user must also provide a python configuration file that contains a function `get_attributes`,[14] which parses the string representing the target morphological analysis into a set of features that will be exposed to the model as the target morphological feature vector $\boldsymbol{\psi}(\mu)$.

Running `morphogen` with the unsupervised morphological analyzer. To use unsupervised morphological analysis, two additional steps (in addition to those required for an external analyzer) are required:

[12]`ducttape` is an open-source workflow management system similar to `make`, but designed for research environments. It is available from `https://github.com/jhclark/ducttape`.

[13]It is also unclear how effective our model would be when translating between two morphologically rich languages, since we assume that the source language expresses syntactically many of the things which the target language expresses with morphology. This is a topic for future research, and one that will be facilitated by `morphogen`.

[14]See the `morphogen` documentation for more information on defining this function. The configuration for the Russian positional tagset used for the "supervised" Russian experiments is provided as an example.

Tokenized source:	`We 've heard that empty promise before . |||` ↩
Tokenized target (inflected):	`Но мы и раньше слышали эти пустые обещания . |||` ↩
Tokenized target (stemmed):	`но мы и раньше слышать этот пустой обещание . |||` ↩
POS + inflectional features:	`C P-1-pnn C R Vmis-p-a-e P---paa Afpmpaf Ncnpan .`

Figure 3. Example supervised input; arrows indicate that the text wraps around to the next line just for ease of reading (there should be no newline character in the input).

- use `fast_umorph`[15] to get unsupervised morphological analyses (see §3.2);
- use `seg_tags.py` with these segmentations to retrieve the lemmatized and tagged version of the target text. Tags for unsupervised morphological segmentations are a simple representation of the learned segmentation. Words less than four characters are tagged with an X and subsequently ignored.

Remaining training steps. Once the training data has been morphologically analyzed, the following steps are necessary:

- process the source side of the parallel data using TurboTagger, TurboParser, and Brown clusters.
- use `lex_align.py` to extract parallel source and target stems with category information. This lemmatized target side is used with cdec's `fast_align` to produce alignments.
- combine to get fully preprocessed parallel data, in the form (source sentence, source POS sequence, source dependency tree, source class sequence, target sentence, target stem sequence, target morphological tag sequence, word alignment), separated by the triple pipe.
- use `rev_map.py` to create a mapping from (stem, category) to sets of possible inflected forms and their tags. Optionally, monolingual data can be added to this mapping, to allow for the creation of inflected word forms that appear in the monolingual data but not in the parallel training data. If a (stem, category) pair maps to multiple inflections that have the same morphological analysis, the most frequent form is used.[16]
- train structured inflection models with SGD using `struct_train.py` A separate inflection model must be created for each word category that is to be inflected. There is only a single category when unsupervised segmentation is used.

[15] `https://github.com/vchahun/fast_umorph`

[16] This is only possible when a supervised morphological analyzer is used, as our unsupervised tags are just a representation of the segmentation (e.g. wa+ku+STEM).

Using `morphogen` for tuning and testing. At tuning and testing time, the following steps are run:

- extract two sets of per-sentence grammars, one with the original target side and the other with the lemmatized target side
- use the extracted grammars, the trained inflection models, and the reverse inflection map with `synthetic_grammar.py` to create an augmented grammar that consists of both the original grammar rules and any inflected synthetic rules (§4). By default, only the single best inflection is used to create a synthetic rule, but this can be modified easily.
- add target language model and optionally a target class based language model. Proceed with decoding as normal (we tune with MIRA and then evaluate on our test set)

Using the `ducttape` workflows. The provided `ducttape` workflows implement the above pipelines, including downloading all of the necessary tool dependencies so as to make the process as simple as possible. The user simply needs to replace the global variables for the dev, test, and training sets with the correct information, point it at their version of morphogen, and decide which options they would like to use. Sample workflow paths are already created (e.g. path with/without Monolingual training data, with/without class based target language model). These can be modified as needed.

Analysis tools. We also provide the scripts `predict.py` and `show_model.py`. The former is used to perform an intrinsic evaluation of the inflection model on held out development data. The latter provides a detailed view of the top features for various inflections, allowing for manual inspection of the model as in Figure 2. An example workflow script for the intrinsic evaluation is also provided.

7. Conclusion

We have presented an efficient technique which exploits morphologically analyzed corpora to produce new inflections possibly unseen in the bilingual training data and described a simple, open source tool that implements it. Our method decomposes into two simple independent steps involving well-understood discriminative models.

By relying on source-side context to generate additional local translation options and by leaving the choice of the global sentence translation to the decoder, we sidestep the issue of inflecting imperfect translations and we are able to exploit rich annotations to select appropriate inflections without modifying the decoding process or even requiring that a specific decoder or translation model type be used.

We also achieve language independence by exploiting unsupervised morphological segmentations in the absence of linguistically informed morphological analyses, making this tool appropriate for low-resource scenarios.

Acknowledgments

This work was supported by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533. We would like to thank Kim Spasaro for curating the Swahili development and test sets.

Bibliography

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Chahuneau, Victor, Eva Schlinger, Chris Dyer, and Noah A. Smith. Translating into morphologically rich languages with synthetic phrases, in review.

Chiang, David. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, 2007.

Chiang, David. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13:1159–1187, 2012.

Dyer, Chris, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proc. of ACL*, 2010.

Lopez, Adam. Hierarchical phrase-based translation with suffix arrays. In *Proc. of EMNLP*, 2007.

Martins, André F.T., Noah A. Smith, Eric P. Xing, Pedro M.Q. Aguiar, and Mário A.T. Figueiredo. Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. of EMNLP*, 2010.

Address for correspondence:
Eva Schlinger
eva@cmu.edu
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA