Modeling Event Extraction via Multilingual Data Sources

Andrew Hsi
Carnegie Mellon University
Pittsburgh, PA 15213 USA
ahsi@cs.cmu.edu

Jaime Carbonell
Carnegie Mellon University
Pittsburgh, PA 15213 USA
jgc@cs.cmu.edu

Yiming Yang
Carnegie Mellon University
Pittsburgh, PA 15213 USA
yiming@cs.cmu.edu

Abstract

In this paper, we describe our system for the TAC KBP 2015 Event track. We focus in particular on development of multilingual event extraction through the combination of language-dependent and language-independent features. Our system specifically handles texts in both English and Chinese, but is designed in a manner to be extendable to new languages. Our experiments on the ACE2005 corpus show promising results for future development.

1 Introduction

Event extraction has been a well-studied field within the information extraction community. Most of the work on event extraction has focused on English texts (Grishman et al., 2005; Ji and Grishman, 2008; Gupta and Ji, 2009; Liao and Grishman, 2010; Liao and Grishman, 2011; Li et al., 2013; Bronstein et al., 2015), with a smaller set of literature focusing on other languages (Chen and Ji, 2009; Li et al., 2012; Chen and Ng, 2012; Chen and Ng, 2014). However, with few exceptions, most works have attempted to simply focus on a single language.

In our system, we attempt to make some initial headway into developing approaches for cross-lingual event extraction that utilize both language-dependent and language-independent features in order to make predictions on texts. Although we do not make any claims to be state-of-the-art on English event extraction, our hope is to make advances for event extraction on less well-studied languages, potentially offering insights on how to adapt event extraction toward low-resource settings.

The remainder of the paper is organized as follows. In Section 2, we introduce some necessary terminology used in the event extraction community. In Section 3, we first describe our overall system architecture, and subsequently describe each component of our system. In Section 4, we describe the specific differences between our various submitted system runs. In Section 5, we describe our experimental results with our system, and in Section 6 we describe our performance in the official TAC evaluations. Finally, in Section 7, we describe our conclusions and thoughts for future work.

2 Terminology

In this section, we describe some common terminology relevant to the remainder of this paper.

An **event mention** is a specific textual instance of an event. Documents may contain multiple event mentions that all point to the same event. For example, consider the following pair of sentences:

(1) The man gunned down two police officers last night. The shooting occurred outside a bank after an attempted robbery.

This example contains two different mentions, but both refer to the same event.

There are two common representations that event mentions can take. An **event trigger** is a single word that indicates the presence of an event in the text. This is a more traditional representation, and was used in the ACE program. In the above example, "shooting" is an event trigger.

More recently, **event nuggets** have been proposed as a generalization of event triggers. In particular, a key differentiating aspect of event nuggets is that they can be multi-word phrases, rather than just a single word. In the above example, this means that in addition to marking "shooting" as an event nugget, we can also consider the phrase "gunned down" as an additional event nugget.

An **event argument** is an entity that fulfills some role within the event. Typically, common roles include Time, Place, and Agent. The set of valid roles depends on the type of event in question – for instance, Conflict.Attack has an Attacker role, while Business.Declare-Bankruptcy does not.

An **event argument mention** is a specific textual instance of an event argument. As in the case with event mentions, there may be multiple event argument mentions that reflect the same event argument.

3 System Components

3.1 Overview

Our system is designed via a set of interconnected components, as shown in Figure 1. We begin with a preprocessing module that takes raw text as input, and runs a suite of tools to prepare the data. The processed data is then fed into our event trigger component, which outputs predictions reflecting which words trigger events. This output is then provided to both a post-processing module for extracting the Event Nugget Detection task output, and an argument prediction component. The output argument predictions are subsequently fed into argument merging and argument post-processing modules, which format the data into the Event Argument Detection and Linking task output.

The event argument verification module forms a separate component that relies on the output from our argument system, as well as the output from the other teams' output. The output of this module serves as the Event Argument Verification and Linking task output.

3.2 Preprocessing

We begin by running the Stanford CoreNLP toolkit on the raw text (Manning et al., 2014). This provides word segmentation, lemmatization, part-of-speech tagging, and other system produced annotations that are used downstream for feature extraction. Additionally, we run the Java Extraction Toolkit (JET) (Grishman et al., 2005), which we use for extracting entity mentions and their coreference links.

3.3 Event Trigger Classification

Given the preprocessed data, we train a logistic regression classifier to classify each word in each document as belonging to one of 33 event trigger subtypes in the ACE2005 ontology, or NONE if the word is not determined to be an event trigger. We train our model with the LIBLINEAR software (Fan et al., 2008).

We consider a variety of features for this module, including:

- the current word/lemma
- bigrams of the current word/lemma with words/lemmas within a fixed context window
- part-of-speech tag for the current word
- bigrams of part-of-speech tags for the current word and words within a fixed context window
- word embedding vector for the current word
- universal part-of-speech tags
- dependent/governor information from dependency parsing

For our word embedding features, we train word embeddings on the full text of English Wikipedia, using the word2vec software (Mikolov et al., 2013). We create our universal part-of-speech features by using available mappings from language-specific part-of-speech tags to the universal part-of-speech tagset (Petrov et al., 2012).

The final part of this component is an additional logistic regression classifier that is run on the output triggers. For each identified trigger word (i.e. those with labels other than NONE), we classify the trigger word into one of three realis labels: ACTUAL, GENERIC, or OTHER. This classifier uses the same base set of features as the previous classifier, with an additional feature added for the predicted trigger type.

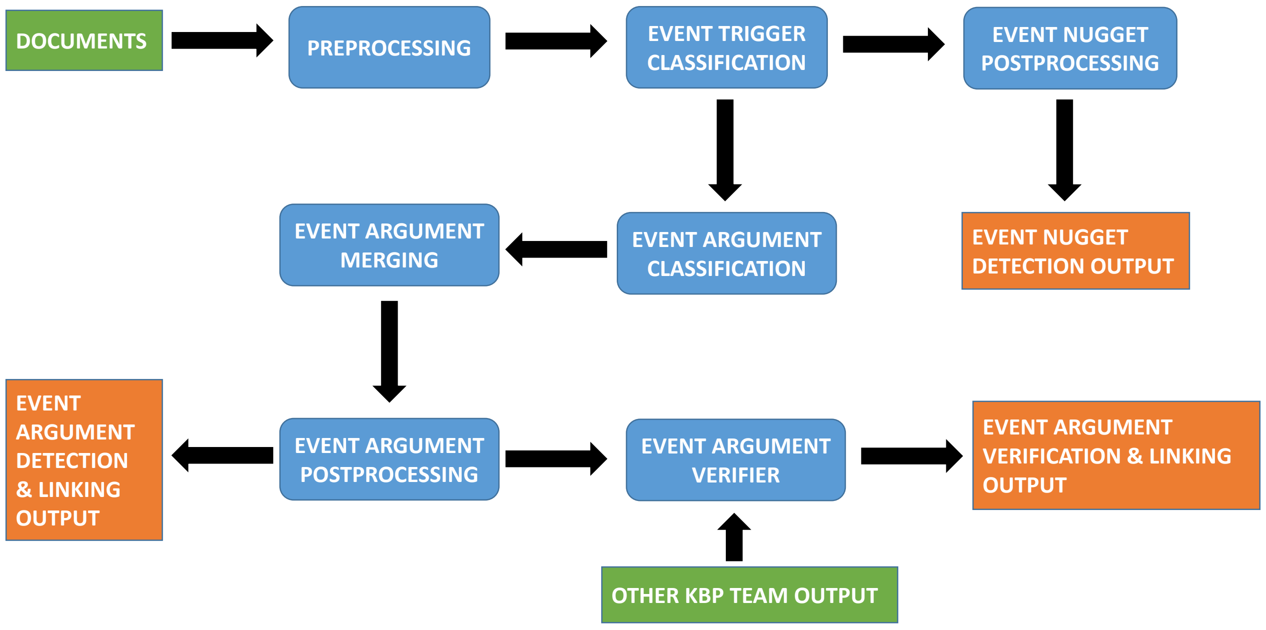
DOCUMENTS
PREPROCESSING
EVENT TRIGGER CLASSIFICATION
EVENT NUGGET POSTPROCESSING
EVENT ARGUMENT MERGING
EVENT ARGUMENT CLASSIFICATION
EVENT NUGGET DETECTION OUTPUT
EVENT ARGUMENT DETECTION & LINKING OUTPUT
EVENT ARGUMENT POSTPROCESSING
EVENT ARGUMENT VERIFIER
EVENT ARGUMENT VERIFICATION & LINKING OUTPUT
OTHER KBP TEAM OUTPUT

Figure 1: Event Extraction Pipeline

3.4 Event Nugget Postprocessing

Since the task at this year's evaluation is based on event nuggets rather than event triggers, our system needs to account for differences between nuggets and triggers. In particular, we need to handle the fact that nuggets may be multi-word phrases, whereas triggers are only ever a single word. For example, consider the following sentences:

(2) The boss fired an employee today.

(3) My friend is out of a job.

In the first sentence, the word "fired" is both an event trigger and an event nugget (specifically for the event type Personnel.End-Position). However, in the second sentence, the phrase "out of a job" can be identified as an event nugget, but would not be a valid event trigger.

We analyzed the RichERE English texts, and identified all multi-word event nuggets. This resulted in a list of 142 unique phrases, varying from lengths of two words to four words. We then considered our predicted event triggers, and for each trigger word, we analyzed a context window of words surrounding the identified trigger. If we found a set of consecutive words that matched any of the phrases in our list, we extended the trigger to contain the necessary context words. Using this process, we converted the event triggers into event nuggets.

3.5 Event Argument Classification

For our event argument classifier, we consider input pairs of the form (t_i, e_j), where t_i is a predicted trigger word (from the trigger classifier) and e_j is an entity extracted from the text with JET. We only consider pairs such that t_i and e_j occur in the same sentence.

For each such pair, we classify the relationship between the trigger word and the entity into one of the argument types from the ACE2005 ontology, or NONE if the entity is not determined to be an argument for the given event mention. We train an additional logistic regression classifier to make these predictions. Once again, we use LIBLINEAR to perform the actual training of our model.

We consider a variety of features for this module, including:

- full phrase of the entity

- unigrams for each word in the entity
- head word of the entity
- entity type, subtype
- trigger word, type, subtype
- distance between the trigger and entity
- dependency links between the trigger and entity

3.6 Argument Merging

In the development of our event argument system, we found difficulty in achieving high recall. As one potential workaround to this problem, we added an additional component that allows for merging argument predictions from other sources. For our purposes, we used the argument predictions from JET, though in principle any systems could be utilized in this module.

Given two (or more) sets of argument predictions, we create our final set of predicted arguments by including any argument that occurred at least once in any of the systems. Although this kind of approach is potentially harmful for precision, it will certainly not lower the recall, and in practice may very well increase it. Note that this is in some sense a simplified version of our Event Argument Verification module, which will be described in Section 3.8.

3.7 Event Argument Postprocessing

Once we have obtained our set of predicted arguments, we pass this output to the event argument postprocessing module to organize the data into the correct format. In addition to straightforward aspects such as providing the document ids and offsets, there are a few non-trivial aspects of this component:

- Verifying validity of argument roles – since our classifier does not explicitly take the validity of argument roles into account, there is the possibility of assigning an impossible role for a particular event type. For example, "Agent" is not a valid argument role for an event of type Conflict.Demonstrate. We pass over each extracted argument, and if we find any invalid roles, we remove the predicted argument from the output.
- Ontology difference – as the KBP ontology differs slightly from the ontology used by ACE2005, we provide a mapping between the two ontologies in order to account for this difference.
- Canonical Argument String (CAS) – for each argument, we were required to report its canonical argument string. To accomplish this, we considered all coreferent entities for each argument, and applied a set of heuristics to select the most applicable string.
- Time normalization – the KBP guidelines requested that the CAS for Time arguments be in the format "XXXX-YY-ZZ", where XXXX is the year, YY is the month, and ZZ is the day. As the default choice, we provided "XXXX-XX-XX" as the CAS, but we also applied a set of heuristics to refine this further based on the extracted strings. For instance, a string containing "november" or "nov" could have its month attribute set to "11".
- Realis label – due to time constraints, we did not develop a classifier to determine the realis label for arguments. As a default approach, we listed each realis label as "ACTUAL".

For the linking output, we linked together all arguments that were assigned to the same event trigger.

3.8 Event Argument Verifier

In this module, we take the output from our event argument postprocessing, as well as the output arguments from the other KBP teams. Our goal in this component is to aggregate and filter the output of different systems into a single output file, and link the arguments together accordingly.

We begin with the aggregation step. We merge together arguments from different systems that have the same argument role, same event type, overlapping CAS, and overlapping base filler. We allow for overlapping CAS and base fillers to merge instead of accepting only exact matches to account for minor system differences in output – intuitively, fillers like "30-year old businessman" and "the 30-year old businessman" should be considered identical. We keep track of the frequency for each merged CAS and base filler, and take the most frequently seen strings for our final output.

After aggregating arguments together, we then filter out arguments that we believe to be of low-confidence. Originally, one idea we had was to somehow incorporate the confidence values provided by system submissions. However, we found that in practice, a large number of these confidence values were uninformative (e.g. uniform confidence across all arguments). As an alternative, we settled upon a weighted voting scheme for filtering arguments. We set the weights of the BBN and OSU submissions to be 1.25, based on their performance from last year's Event track. For all other systems, we set the weights uniformly at 1. For each argument, we scored it by the number of systems that included the argument, and we set three different thresholds α for filtering low-scoring arguments:

- $\alpha = 1.25$ – the least restrictive threshold we tried, which requires that each argument be identified by at least two systems, or by one of the high-confidence systems
- $\alpha = 2$ – a middleground threshold, which requires that each argument be identified by at least two systems
- $\alpha = 4$ – the most restrictive threshold we considered, which requires that each argument be identified by at least four systems

For our linking strategy, we considered two separate approaches. In the first approach, we considered the provided predicate justification spans, and linked together any pair of arguments with overlapping spans and matching event types. We believe this to be a fairly safe linking strategy in that we will likely not make too many mistakes (higher precision). However, we may also miss a large number of links (lower recall).

To overcome this, we attempted a higher recall strategy for our second approach, in which we additionally attempt to estimate coreference links by considering the CAS. In this approach, for each pair of arguments with the same event type, we first consider whether two arguments have the same CAS. If they do, we assume them to be coreferent, and automatically link them together. If they do not, we fall back on our first approach.

4 Description of System Runs

In this section, we describe the specific differences between our submitted runs for the Event Track.

4.1 Event Nugget Detection

We submitted a set of three runs for the Event Nugget Detection task.

- CMU_CS_EVENT1: trained using English RichERE, English ACE2005, and Chinese ACE2005
- CMU_CS_EVENT2: trained using English RichERE and English ACE2005
- CMU_CS_EVENT3: trained using English ACE2005 and Chinese ACE2005

4.2 Event Argument Detection and Linking

We submitted a set of two runs for the Event Argument Detection and Linking task.

- CMU_CS_EVENT1: trained using both English and Chinese documents from ACE2005
- CMU_CS_EVENT2: trained using only English documents from ACE2005

4.3 Event Argument Verification and Linking

We submitted a set of five runs for the Event Argument Detection and Linking task.

- CMU_CS_EVENT1: $\alpha = 2$, estimates coreference for linking
- CMU_CS_EVENT2: $\alpha = 2$, links based solely on predicate justification spans
- CMU_CS_EVENT3: $\alpha = 4$, estimates coreference for linking
- CMU_CS_EVENT4: $\alpha = 4$, links based solely on predicate justification spans
- CMU_CS_EVENT5: $\alpha = 1.25$, links based solely on predicate justification spans

5 Experimental Results

For our development purposes, we utilized the ACE2005 English and Chinese texts. We held out 30 English documents for our test set, and an additional 40 English documents for parameter tuning.

We provide experimental results for two different systems. In system 1 (CMU-English), we use only the English data for training our model. In system 2 (CMU-Multilingual), we use both the English data and the Chinese data simultaneously to train a single model. Note that for these experiments, we skip the Argument Merging and Verification components.

Traditionally, the event extraction literature reports results on micro-averaged precision, recall, and F1. Following past work, we also report on micro-averaged results, but we additionally show macro-averaged results. Our motivation for doing so is due to the high class-imbalance within the ACE data – certain frequently seen event types (e.g. Conflict.Attack, Life.Die) tend to be much easier to learn good predictors for than the rare event types (e.g. Justice.Extradite, Life.Divorce). Macro-averaged results can provide some insight into how well we are doing over classes, rather than over instances.

Results for event trigger detection are provided in Table 1. We find that using only English provides superior performance for the micro-averaged results. However, when we switch to macro-averaged results, the multilingual version has higher performance. This would seem to suggest that while the frequent classes are being hurt by the Chinese data, it is in fact helping on the low-frequency classes.

We now move on to argument detection. Since the argument component relies heavily on both the input triggers and the input entities, we provide results under three different settings:

- Gold triggers, gold entities
- System-predicted triggers, gold entities
- System-predicted triggers, system predicted entities

We begin with results under the setting of perfect information, seen in Table 2. The results are similar to those of the trigger setting. Once again, we see better performance under the micro-averaged setting with the English-only baseline, but when considering the macro-averaged setting, find improved performance when factoring in the additional Chinese data.

We now show results for the setting of imperfect information. Table 3 shows results under the scenario of having system-produced triggers and gold entity information, while Table 4 show results under the setting of completely relying on system-produced input. When moving to this setting, all measures show drops in performance. This is to be expected, as the system-produced input will unavoidably introduce noise. Under this setting, the multilingual model lags behind the English-only model under both the macro-averaged and micro-averaged settings. This would seem to indicate that the current multilingual model is not as robust to noisy input. We hope to address this concern in future work.

6 Official Evaluations

Our official results for the TAC 2015 Event Argument and Linking track can be seen in Table 5. In addition to the standard precision, recall, and F1 metrics, we also report the EAArg and EALink subscores used in the official evaluation. The overall score is a weighted sum of the EAArg and EALink scores, and is used for determining the rankings. Our standalone system (CMU_CS_Event) ranked 5th among the submissions (excluding LDC's manual run), and our verification system ranked 2nd among the submitted systems. A clear weak point in our standalone submission is our low system recall, which results in a noticeable drop in overall performance despite having a fairly high score for precision. The results of our verification system achieve the highest recall among all the submissions, including that of LDC's manual run.

Our official results for the TAC 2015 Event Nugget Detection track can be seen in Table 6. Runs were ranked based on micro-average F1. Our system ranked 11th among the submitted runs. A more detailed breakdown of our system's performance over precision and recall (see Table 7) indicate that our results are being hurt primarily by low recall, which is consistent with our findings in the argument-focused track.

Training type	Micro-P	Micro-R	Micro-F1	Macro-P	Macro-R	Macro-F1
CMU-English	0.805	0.538	0.645	0.629	0.411	0.474
CMU-Multilingual	0.782	0.541	0.640	0.644	0.453	0.511

Table 1: Results for Event Trigger Classification on ACE2005

Training type	Micro-P	Micro-R	Micro-F1	Macro-P	Macro-R	Macro-F1
CMU-English	0.807	0.502	0.619	0.539	0.372	0.430
CMU-Multilingual	0.778	0.491	0.602	0.531	0.393	0.443

Table 2: Results for Event Argument Classification on ACE2005, using gold triggers and entities

Training type	Micro-P	Micro-R	Micro-F1	Macro-P	Macro-R	Macro-F1
CMU-English	0.720	0.300	0.424	0.481	0.206	0.278
CMU-Multilingual	0.679	0.288	0.404	0.465	0.195	0.259

Table 3: Results for Event Argument Classification on ACE2005, using system triggers and gold entities

Training type	Micro-P	Micro-R	Micro-F1	Macro-P	Macro-R	Macro-F1
CMU-English	0.561	0.169	0.261	0.286	0.098	0.136
CMU-Multilingual	0.525	0.164	0.250	0.285	0.090	0.131

Table 4: Results for Event Argument Classification on ACE2005, using system triggers and gold entities

Submission	P	R	F1	EAArg	EALink	Overall	Rank
LDC	75.5	40.0	52.3	36.7	33.7	35.2	N/A
Top-ranked system	36.8	39.2	38.0	23.6	23.3	23.5	1
ver-CMU_CS_event	15.0	47.8	22.8	5.6	21.1	13.4	2
CMU_CS_event	30.5	9.9	14.9	5.2	3.6	4.4	5
Lowest-ranked system	10.0	12.9	11.3	1.3	4.6	2.9	8

Table 5: TAC 2015 Event Argument and Linking Results

Submission	Micro-Average F1
Rank 1 system	44.24
Rank 2 system	41.77
CMU_CS_event	25.54
Bottom-ranked system	13.89

Table 6: TAC 2015 Event Nugget Detection Results: Overall performance

Submission	P	R	F1
CMU_CS_Event	57.88	16.39	25.54

Table 7: TAC 2015 Event Nugget Detection Results: Micro-averaged precision/recall/F1 breakdown

7 Conclusions and Future Work

In this paper, we have described our submitted system for the Event Track of TAC KBP 2015. We focus in particular on developing techniques for multilingual event extraction, via a system that uses available training data in both English and Chinese to make predictions on a single target language (English for the purposes of KBP). Our experimental results show that while an English-only model is still superior for achieving high scores on micro-averaged metrics, introducing additional multilingual training data can offer improvement on macro-averaged performance for event trigger detection, as well as for event argument detection under certain conditions.

We believe that there are several promising directions to pursue to improve our overall system performance. Firstly, we did not produce a realis classifier for the argument model, which currently uses a trivial approach of always predicting ACTUAL. Training such a classifier should certainly help with our performance in the future. Secondly, there is an important distinction between the ACE corpora and the KBP evaluation data, in that the KBP data contains a large amount of discussion forum data. To date, we have not focused on development of techniques for this kind of text, which is typically noisier than newswire and may require specialized techniques in order to achieve optimal performance. Thirdly, one common challenge across all our results is that high recall seems to be more difficult to achieve than high precision. We would like to investigate this behavior further and attempt to boost recall closer to our precision levels. Lastly, it may be beneficial to adapt our pipeline of classifiers approach to the structured approach of Li et al. (2013), which has been shown to be a more effective approach to learning the model parameters.

One final observation to make is that our current system focuses on two languages that can be considered fairly resource-rich for event extraction – compared to other languages, there is a large amount of available training data for event extraction on English and Chinese. In our experiments, we found that the multilingual data seemed to help primarily on the infrequent classes, where we have insufficient example datapoints. We hypothesize that this difference could become even more significant under a truly low-resource setting. We leave this exploration to future work.

Acknowledgments

This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program.

References

Ofer Bronstein, Ido Dagan, Qi Li, Heng Ji, and Anette Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? In *ACL*.

Zheng Chen and Heng Ji. 2009. Language specific issue and feature exploration in chinese event extraction. In *HLT-NAACL*.

Chen Chen and Vincent Ng. 2012. Joint modeling for chinese event extraction with rich linguistic features. In *COLING*.

Chen Chen and Vincent Ng. 2014. Sinocoreferencer: An end-to-end chinese event coreference resolver. In *LREC*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. In *JMLR*.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyus english ace 2005 system description. In *Proc. ACE 2005 Evaluation Workshop*.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *ACL-IJCNLP*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

Peifeng Li, Guodong Zhou, Qiaoming Zhu, and Libin Hou. 2012. Employing compositional semantics and discourse consistency in chinese event extraction. In *EMNLP*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Shasha Liao and Ralph Grishman. 2010. Filtered ranking for bootstrapping in event extraction. In *ACL*.

Shasha Liao and Ralph Grishman. 2011. Acquiring topic features to improve event extraction: in pre-selected and balanced collections. In *RANLP*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC*.