UZH at TAC KBP 2017: Event Nugget Detection via Joint Learning with Softmax-Margin Objective

Peter Makarov **Simon Clematide**
Institute of Computational Linguistics,
University of Zurich, Switzerland
makarov@cl.uzh.ch simon.clematide@cl.uzh.ch

Abstract

This paper presents the CLUZH systems for the TAC KBP 2017 Event Nugget Detection Task. We submit three runs in all three languages: English, Spanish, and Chinese. Each run is an ensemble of five recurrent neural network (RNN) systems. Run 2 ensembles over pipelines of an RNN Span/Subtype classifier followed by an RNN Realis classifier. Each system of the Run 3 ensemble is a jointly learned model that predicts all properties of the Event Nugget: its Span, Subtype, and Realis. Unlike the other runs, Run 1 is trained without early stopping on all labeled data. For each language, it uses the architecture that results in the best development set performance: joint models for English and Spanish and pipelines for Chinese. All single models are trained with a softmax-margin objective that heavily penalizes recall and entity errors. We achieve the highest overall scores for all three languages in this year's evaluation, largely with our Run 1. Our post-submission experiments indicate that joint models are consistently better than pipelines; pre-trained embeddings, cost-sensitive learning, and bidirectional RNN encoding are crucial to the models' strong performance; ensembling always helps; but trading early stopping for training with more data is error-prone.

1 Introduction

This paper presents the submissions of the CLUZH team to the Event Nugget Detection (EN) Task of the TAC KBP 2017 evaluation. The highlights of our work are i) the joint extraction of all the Event Nugget properties (Span, Subtype, Realis) as opposed to the classical pipeline solution, ii) the application of cost-sensitive learning to boost recall as the task is vastly dominated by non-Event tokens, and iii) ensembling models to fight high variance. For feature encoding, we choose a standard RNN architecture for sequence labeling: a single-layer bidirectional Long Short-Term Memory (LSTM) encoder. As input, we use off-the-shelf word vectors and simple morpho-syntactic features derived from the Stanford CoreNLP toolkit.

BROADCAST,ACTUAL DEMONSTRATE,ACTUAL
Novosti's coverage of Ukrainian protests may well have hastened its demise.

Figure 1: A sentence with gold Event Nuggets: "coverage" triggers one BROADCAST Event, and "protests" triggers one DEMONSTRATE Event. The Realis value of both Events is ACTUAL.

We submit three runs in all three languages (English, Spanish, and Chinese), and each run is an ensemble of five systems. Run 2 is an ensemble of more common pipelines of Span/Subtype and Realis classifiers. Run 3 ensembles over jointly learned models predicting all Nugget properties at once. Run 1 does not feature any new model design; instead, it uses all available labeled data for training and no early stopping. Other than this, Run 1 is just like Run 2 or 3, depending on which ensemble has the highest development set performance for that language.

In the following, we first present the task and our simplified reformulation of it. The gold EN data feature a number of complex phenomena, e.g. Event Nuggets triggering multiple Events of different Subtypes. Fortunately, most of these difficult cases are infrequent, and so one typically simplifies the task to avoid them altogether. Next, we list the datasets that we use for training. We

Type	Subtypes
CONFLICT	ATTACK, DEMONSTRATE
CONTACT	BROADCAST, CONTACT, CORRESPONDENCE, MEET
JUSTICE	ARREST-JAIL
LIFE	DIE, INJURE
MANUFACTURE	ARTIFACT
MOVEMENT	TRANSPORT-ARTIFACT, TRANSPORT-PERSON
PERSONNEL	ELECT, END-POSITION, START-POSITION
TRANSACTION	TRANSACTION, TRANSFER-MONEY, TRANSFER-OWNERSHIP

Table 1: Eight Event Types and eighteen Event Subtypes of the TAC KBP 2017 EN Task.

then discuss feature encoding, which is shared by all single models across all languages. We then present our classifiers, training and ensembling methodologies. We conclude with a discussion of our official results and post-submission experiments.

2 Task Description and Reformulation

An Event Nugget is a span of text that most concisely and directly refers to an Event, or *triggers* it (Figure 1). One Event Nugget can trigger multiple Events. In the 2017 version of the EN Task, the Event ontology comprises eighteen Subtypes, each of which also uniquely determines the Type of the Event (Table 1). The factuality of the Event is captured by the Realis attribute, which has three values: ACTUAL, GENERIC, and OTHER.

EN is, thus, the task of identifying

1. character spans of all Event Nuggets in a document,
2. the Types and Subtypes of the Events triggered by the Event Nuggets, and
3. the Realis values of these Events.

We simplify this task in a number of ways, often suggested by other researchers.

Token As Event Nuggets are predominantly single-token expressions, following Reimers and Gurevych (2015, 2017), we do not use any label encoding schemas like IOB. We also identify sub-token Event Nuggets with full tokens. Thus, if the Event Nugget is "ex", we choose to predict the full token instead e.g. "ex-military". We do not attempt to correct such cases in a post-processing step.

Span Following Nguyen et al. (2016b); Lu and Ng (2016), we directly predict the Subtype of each token, including the NULL-Subtype that corresponds to a non-Event. Similarly, our joint model directly predicts the structured Subtype–Realis label of each token, and the structured (NULL-Subtype)–(NULL-Realis) label corresponds to a non-Event token. The Span is then deterministically derived from the predicted label.

Subtype If an Event Nugget triggers multiple Events of one Subtype, we learn to predict only one Event of this Subtype. Further, we extend the set of eighteen atomic Subtypes with pairs of Subtypes commonly triggered together by one Nugget (seen at least one hundred times in the labeled data), e.g. ATTACK|DIE as in

$$\overbrace{\text{Rabin's assassination}}^{\text{ATTACK|DIE,ACTUAL}} \text{ in 1995}$$

For all languages, we obtain a total of twenty-two Subtypes: the atomic Subtypes, the NULL-Subtype, and three composite Subtypes. For all languages, two of the retained composite Subtypes are ATTACK|DIE and TRANSFER-MONEY|TRANSFER-OWNERSHIP. The third one is ATTACK|INJURE for English and Spanish and DIE|INJURE for Chinese.

Realis For each Event Nugget, we only learn to predict one Realis value that is the most common in the data. Like with Subtypes, we only predict one Realis value even if a Nugget triggers multiple Events with that Realis value. This leaves us with four Realis labels: ACTUAL, GENERIC, OTHER, and NULL.

3 Datasets

In our choice of labeled data, we follow closely last year's participants. For simplicity, we use only Rich ERE annotations. Thus, one source is LDC2017E02, which is a compilation of TAC KBP EN training and evaluation datasets for years 2014, 2015, and 2016. From this compilation, we only take 2015 and 2016 data. The other source is DEFT Rich ERE annotations: for English, LDC2015E29 and LDC2015E68; for Spanish, LDC2015E107, LDC2016E34, and LDC2017E51; for Chinese, LDC2015E78,

Lang	Train set		Dev set
	DEFT Rich ERE	LDC2017E02	LDC2017E02
eng	LDC2015E29, LDC2015E68	2015: all, 2016: 69 docs	2016: 100 docs
spa	LDC2015E107, LDC2016E34, LDC2017E51	2015: all, 2016: 119 docs	2016: 50 docs
cmn	LDC2015E78, LDC2015E105, LDC2015E112	2015: all, 2016: 97 docs	2016: 70 docs

Table 2: Overview of the datasets and the training / development set splits used in this paper.

Lang	# tokens	# Event tokens, %	
eng	553.0K	18.1K	3.3%
spa	238.9K	6.5K	2.7%
cmn	331.0K	6.9K	2.1%

Table 3: Labeled data statistics: Overall number of tokens; number and percentage of tokens forming Event Nugget Spans (*Event tokens*).

LDC2015E105, and LDC2015E112. From LDC2015E78, we only use the original Chinese data and ignore English translations. For each language, non-Event tokens account for over 97% of all the text (Table 3).

For development, we draw samples of around 12% from LDC2017E02's 2016 evaluation data (Table 2) such that half of the development documents are newswire text and the other half—discussion forms.

We only use Event Nugget annotations, i.e. Trigger spans and Event Mention Type/Subtype and Realis annotations. Thus, unlike other teams, e.g. Lu and Ng (2016), our models do not rely on event argument annotations in any way. We also leave out Events of Subtypes other than the eighteen Subtypes of the 2017 version of the Task, unlike Nguyen et al. (2016b).

4 Feature Encoding

We preprocess all documents with the Stanford CoreNLP toolkit 3.8.0 (Manning et al., 2014). For each token, we extract the following features:

1. the token itself,
2. its Penn TreeBank part-of-speech (POS) tag,
3. the Universal Dependency relation (UD) label of this token,
4. the set of UD labels that govern the dependents of this token—following Nguyen et al. (2016b),
5. the binary text type feature—"newswire" or "discussion forum"—following Reimers and Gurevych (2015), and
6. the binary feature that signals whether the token is in a quote region.

Let $\mathbf{x} = x_1, \ldots, x_n$ be a sentence of n tokens and $\mathbf{y} = y_1, \ldots, y_n$ the sequence of their labels, where each label y_t is from some set of labels $\mathcal{Y}$. We shall now define our feature function $g(t, \mathbf{x}, \Phi)$ that produces a token representation $\mathbf{s}_t \in \mathbb{R}^{2H}$ for each x_t. This function g first embeds the features extracted for each x_t and concatenates the resulting embeddings, as well as some n-hot feature representations, into $\mathbf{i}_t$. Next, the sequence $\mathbf{i}_1, \ldots, \mathbf{i}_n$ of such vectors is used as input to a bidirectional RNN. Thus, s_t is a global-context–aware representation of x_t (hence, the dependence of g on the entire $\mathbf{x}$), enriched with information from all preceding and subsequent tokens. Φ are the parameters of g (as well as parameters of the model), which comprise all trainable embedding parameters and the parameters of the RNN.

For each token x_t, let $\mathbf{i}_t$ be the concatenation of the following vectors:

1. the pre-trained token embedding $\mathbf{t}_t \in \mathbb{R}^T$, that we do not update during training—following Marcheggiani et al. (2017),
2. the trainable token embedding $\mathbf{e}_t \in \mathbb{R}^E$,
3. the trainable POS embedding $\mathbf{o}_t \in \mathbb{R}^O$,
4. the trainable UD label embedding $\mathbf{u}_t \in \mathbb{R}^U$,
5. the n-hot vector $\mathbf{d}_t \in \{0, 1\}^D$ representing the set of UD labels (Item 4 in the feature list), and
6. vector $\mathbf{a}_t \in \{0, 1\}^2$ of text type and quote region features (Features 5 and 6).

To produce $\mathbf{s}_1, \ldots, \mathbf{s}_n$ for each of the tokens in the sentence, we encode $\mathbf{i}_1, \ldots, \mathbf{i}_n$ with a bidirectional LSTM (Graves and Schmidhuber, 2005):

$$\mathbf{s}_1, \ldots, \mathbf{s}_n = \text{BiLSTM}(\mathbf{i}_1, \ldots, \mathbf{i}_n) \quad (1)$$

The way we represent tokens—our choice of features and how we encode them—is, therefore, fairly standard and is influenced by semantic role labeling literature and successful work on EN.

5 Classifiers and Loss Functions

Our classifiers predict the label of a token independently of the labels of other tokens in the sentence. This is also the case for joint models of Run 3, which, for each x_t, predict a structured label that is independent from all other labels in the sentence.

In the kind of multi-class classifier that appears in our pipeline systems, the score of label $y \in \mathcal{Y}$ for input token x_t with representation $\mathbf{s}_t$ is given by

$$f_{\text{plain}}(x_t, y) = \mathbf{w}_y^\top \mathbf{s}_t \tag{2}$$

$\mathbf{w}_y$ is a parameter vector for y and is in the model parameters Θ, which also include the parameters Φ of the feature function g.

Given x_t, the classifier predicts

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} f_{\text{plain}}(x_t, y) \tag{3}$$

Instead of the more common log loss, we train all our classifiers with the softmax-margin loss (Gimpel and Smith, 2010a), which allows for incorporating extra cost for specific types of error.

Softmax-margin loss Softmax-margin augments the score of label y for input token x_t (Eq. 2) with a cost of choosing y when the true label is y_t:

$$f(x_t, y) = \mathbf{w}_y^\top \mathbf{s}_t + \text{cost}(y_t, y) \tag{4}$$

We use the following cost function to assign extra cost to y when the truth is y_t:

$$\text{cost}(y_t, y) = \begin{cases} 0 \text{ if } y_t = y, \\ P \text{ if } y_t = \text{NULL} \wedge y \neq \text{NULL}, \\ R \text{ if } y_t \neq \text{NULL} \wedge y = \text{NULL}, \\ T \text{ if } y_t \neq \text{NULL} \neq y \wedge y_t \neq y. \end{cases} \tag{5}$$

and $P, R, T \in \mathbb{R}_{\geq 0}$. Thus, we add cost P whenever choosing y would result in a false-positive error, cost R—a false-negative error, and cost T—a wrong non-NULL label prediction (=entity error) (Gimpel and Smith, 2010b), which occurs when the classifier correctly identifies the token as triggering an Event but assigns it a wrong Subtype or Realis label.

The loss function that we maximize has the following familiar form:

$$\log \mathcal{L}(x_t, y_t; \Theta) = f(x_t, y_t) - \log \sum_{y \in \mathcal{Y}} \exp f(x_t, y) \tag{6}$$

Our classifiers typically incur a large additional loss for false negative errors. In this way, we compensate for the infrequency of Event tokens in the data (Table 3).

Joint Subtype-Realis learning Run 3 ensembles single classifiers that learn to jointly predict Subtype and Realis labels. In such a classifier, Subtype labels $y \in \mathcal{Y}$ and Realis labels $v \in \mathcal{V}$ are scored separately but from the same token representation $\mathbf{s}_t$. Additionally, a set of label compatibility parameters $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{Y}|}$ is learned:

$$f_{\text{plain}}(x_t, y, v) = \mathbf{w}_y^\top \mathbf{s}_t + \mathbf{u}_v^\top \mathbf{s}_t + A_{v,y} \tag{7}$$

and $\mathbf{w}_y$, $\mathbf{u}_v$, and $\mathbf{A}$ are in Θ. Given an input token x_t, the classifier predicts

$$\hat{y}, \hat{v} = \arg\max_{y \in \mathcal{Y}, v \in \mathcal{V}} f_{\text{plain}}(x_t, y, v) \tag{8}$$

At training, the score is augmented with cost:

$$\begin{aligned} f(x_t, y, v) = &\left(\mathbf{w}_y^\top \mathbf{s}_t + \text{cost}^{Su}(y_t, y)\right) \\ &+ \left(\mathbf{u}_v^\top \mathbf{s}_t + \text{cost}^{Re}(v_t, v)\right) + A_{v,y} \end{aligned} \tag{9}$$

And the loss that we maximize is given by

$$\begin{aligned} \log \mathcal{L}(x_t, y_t, v_t; \Theta) = &f(x_t, y_t, v_t) \\ &- \log \sum_{y \in \mathcal{Y}} \sum_{v \in \mathcal{V}} \exp f(x_t, y, v) \end{aligned} \tag{10}$$

The model lets the choices of the Realis and Subtype attributes for token x_t influence each other. First, the token representation $\mathbf{s}_t$ is learned to score both Subtype and Realis labels for x_t. Second, the label dependency is explicitly modeled with the label compatibility parameters $\mathbf{A}$, and the model normalizes over all Subtype and Realis label pairs.

The cost functions cost^{Su} (for Subtype) and cost^{Re} (for Realis) are like cost in Eq. 5 but with possibly different P, R, and T values.

Pipeline of classifiers Run 2 ensembles over pipelines of two classifiers: a Subtype classifier followed by a Realis classifier. Each classifier learns its own token representation $\mathbf{s}_t$. The Subtype classifier is like in Eq. 3 trained with the loss in Eq. 6. Somewhat unconventionally, instead of a simple Realis classifier that conditions on (i.e. gets as part of input) the Subtype label $\hat{y}$ produced by the Subtype classifier, we train a joint Subtype-Realis model and output the Realis label of the

Hyperparameter	Value
dim. T of trainable token embeddings	100
dim. E of fixed token embeddings	300
dim. O of POS tag embeddings	20
dim. U of UD label embeddings	20
dim. H of LSTM hidden layer	200
LSTM depth	1
word dropout	.25
precision cost P	0
recall cost R	10
entity cost T	5

Table 4: Hyperparameter values of all single models in this paper.

highest scoring non-NULL structured label for that x_t. This strategy appears to work better than conditioning on the Subtype label $\hat{y}$ in Eq. 8 and finding the argmax over $v \in \mathcal{V}$.

6 Post-processing

We apply some post-processing to the output of our models. If a contiguous sequence of tokens are assigned the same label, we take the whole sequence to be the Span of one Nugget with that label. In case a sequence of tokens forming an Event Nugget and a following token have the same Subtype but different Realis labels, we merge them into one Event Nugget and assign it the Realis of the sequence unless the token is a verb, in which case the Nugget gets the Realis of the token.

7 Ensembling

To fight high variance, we ensemble five single systems: five joint models for Run 3 and five pipelines for Run 2. For each Event Nugget detected by any of the single systems, we calculate its support—the number of systems that predict it. We treat minimal support as a hyperparameter that we tune on the development set: For the ensemble to predict an Event Nugget, its support should be at least as high as the required minimum (e.g. two systems out of five).

8 Training and Hyperparameters

We apply the same set of hyperparameter values to all single models and all languages that we find to work reasonably well for English (Table 4). Due to time restrictions, we could only conduct a very limited, heuristic-driven search for satisfactory hyperparameter values and rely a lot on the literature.

Lang	Run 1		Run 2	Run 3
	# ep.	like	support	
eng	7	Run 3	3	2
spa	8	Run 3	3	3
cmn	5	Run 2	3	2

Table 5: Run- and language-specific configurations. Run 1: number of epochs before training termination, the run whose configuration (=type of single system and minimal support) it inherits. Run 2 and 3: minimal support (§7).

We use LSTMs with peephole connections (Greff et al., 2017) and a forget state set to $\mathbf{1} - \mathbf{i}_t$, where $\mathbf{i}_t$ is the input gate. All the models in our submission contain a bug in the implementation of the bidirectional LSTM, which has a mixed effect on test performance (see Figure 2).

We opt for the off-the-shelf word2vec embeddings: the GoogleNews embeddings of Mikolov et al. (2013) for English, Spanish embeddings trained on an assortment of corpora (Cardellino, 2016),[1] and Chinese embeddings trained on Wikipedia.[2] We do not experiment with the choice of embeddings, although more syntax-aware embeddings are observed to positively impact performance on this task (Reimers and Gurevych, 2015).

Model parameters Θ are initialized as recommended in Glorot and Bengio (2010). All single models are trained with Adadelta (Zeiler, 2012). We train with minibatches of ten sentences. In Runs 2 and 3, all single models are trained for thirty epochs with early stopping after five epochs of patience. Models of Run 1 are trained without early stopping for the number of epochs that we heuristically compute as follows. For each language, we take the average number of epochs that the single models of the run with the best development set performance take to stop, add to this one population standard deviation, and round. In this way, we train joint models for English and Spanish for seven and eight epochs respectively, and pipelines for Chinese for five epochs.

The minimal support hyperparameter, which enters the ensemble computations, is set to maximize the performance on the development set for

[1] http://crscardellino.me/SBWCE/

[2] https://github.com/Kyubyong/wordvectors

Span	Sub	Realis	Sub+Real
English			
67.27	56.19	**48.33**	**39.73**
59.95	50.37	47.48	39.28
59.63	50.14	47.42	39.24
59.16	**48.60**	42.47	36.81
Spanish			
50.17*	**42.81***	**36.81**	**30.95**
31.10	23.25	17.60	12.69
Chinese			
54.74	**50.64**	**43.37**†	**39.97**†
52.85	46.76	35.35	32.43
49.76	46.45	35.19	31.51
47.12	42.14	34.58	31.17

Table 6: Official results: Top four results for English and Chinese, all EN Track results for Spanish. Our results are marked in bold. Star ($\star$) marks results by Run 2, dagger ($\dagger$) results by Run 3, all other scores are from Run 1.

Lang	Run	Span	Sub	Real	Sub+Real
eng	1	**59.16**	**48.60**	**48.33**	**39.73**
	2	58.39	48.29	44.31	36.69
	3	57.27	47.66	46.67	38.96
spa	1	49.00	41.17	**36.81**	**30.95**
	2	**50.17**	**42.81**	7.68†	6.46†
	3	47.89	39.80	34.21	28.79
cmn	1	**54.74**	**50.64**	42.52	39.66
	2	53.09	49.26	40.58	37.88
	3	53.71	48.97	**43.37**	**39.97**

Table 7: Test set performance of our runs. Due to a bug, Spanish Run 2 Realis labels could not be mapped correctly, hence the low scores ($\dagger$).

Runs 2 and 3. It is set to two for English and Chinese joint-model ensembles and three for all the rest. For the Run 1 ensembles, we choose the same minimal support as in the run that they inherit their configuration from (Table 5).

All the models are implemented in DyNet (Neubig et al., 2017).

9 Results and Discussion

Official results All in all, ten teams submit English runs, three—Chinese, and two—Spanish to the EN Task this year. One of the Cold Start submissions, projected to the EN evaluation dataset, achieves competitive scores in the EN Track for Chinese. Table 6 shows how our models rank compared to the other participants.

Our models are overall best in all three languages and are best in every sub-part of the task except English Span and Subtype (called "Plain" and "Type" respectively in the official results). These high scores are due to Run 1 with the exception of Chinese Realis and Subtype+Realis (Run 3 is better) and Spanish Span and Subtype (Run 2 is better).

The risky Run 1 strategy pays off well (Table 7). We examine this risk more closely in a post-submission experiment.

The official results for English show that we are particularly strong in predicting Realis and combining Realis labels with Subtype labels. We attribute the latter to joint modeling, which compensates for our relatively weak performance on Subtype. We observe this effect when comparing our Runs 2 and 3 (Table 7, English and Chinese): The specialized Span/Subtype classifier performs strongly on Subtype, but adding to it Realis labels from a strong joint model results in worse performance on Realis and overall. We relate this effect to a mismatch between the models in a pipeline (including the effect of performing well, but on slightly different subsets of the data, which would be possible given the rather low inter-annotator agreement on this task (Reimers and Gurevych, 2015)). Whether this is indeed the case and whether conditioning on the Subtype label would help is left to future work. An alternative solution to this problem could also be a different pipelining strategy—by first ensembling and then pipelining.

Post-submission experiments We run post-submission experiments to gain a better insight into (i) how our choices of features, architecture, and the objective function affect model performance, (ii) how much improvement we gain with ensembling, and (iii) how reliable the Run 1 strategy is.[3]

Surprisingly, many of the additions do not improve performance on average (Figure 2). Except for Spanish pipeline systems, all morpho-syntactic features at best have no positive impact (-POS, -DEP, -GOV). Even switching off all of them together (-SYN) does not harm performance. Precomputed fixed word embeddings (-FE) are cru-

[3] Somewhat unfortunately, the experiments feature LSTMs without peephole connections and with the standardly defined forget state.

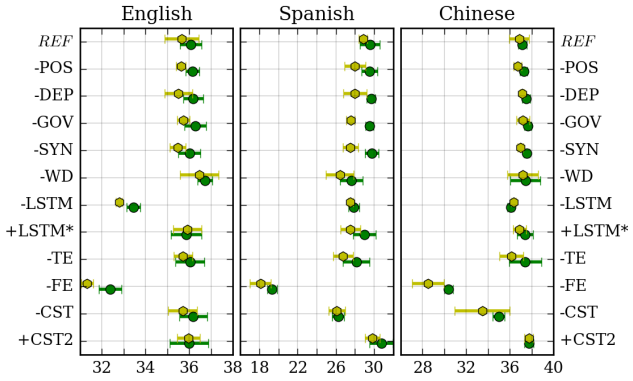

Figure 2: Ablation experiments: Effects of modifying the model on Subtype+Realis test set F1-score. Averages over five single models. Joint models are green circles, pipelines are yellow hexagons. Error bars show population standard deviation. REF=configuration from submission (Table 4) with corrected BiLSTM, −POS=no POS embeddings, −DEP=no UD label embeddings, −GOV=no set of dependents' UD labels, −SYN=no morpho-syntactic features, −WD=no word-dropout, −LSTM=no BiLSTM, +LSTM*=buggy BiLSTM, −TE=no trainable token embeddings, −FE=no pre-trained fixed embeddings, −CST = no costs, +CST2=costs 5 for recall errors and 2 for entity errors.

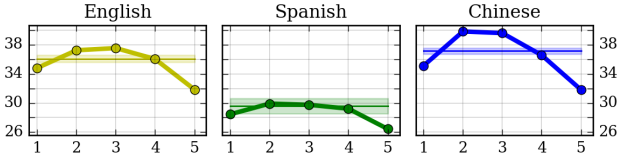

Figure 3: Effect of minimal support parameter (x-axis) on the ensemble's Subtype+Realis test set F1-score. F1-score for joint-model ensembles with different support (thick); mean F1-score (thin) and population standard deviation (shaded region) for the same joint models.

cial for all three languages. Interestingly, proper BiLSTM encoding of the sentence adds a lot to English models, but less so for Spanish and Chinese. Although we employ word dropout in our submission (Iyyer et al., 2015; Marcheggiani et al., 2017), the ablation experiments suggest that it rather harms performance for English, but helps quite a lot for Spanish. Despite the gains from placing high costs on recall and entity errors, we now observe that lower costs achieve even better results for all languages.

We also note that, in our initial experiments

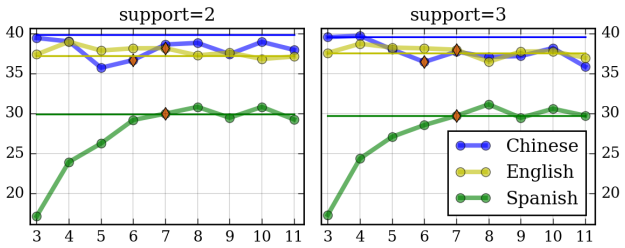

Figure 4: Training with all data (Run 1) vs early stopping on the example of joint-model ensembles with different minimal support. Subtype+Realis test set F1-score for the ensembles after each training epoch (thick); the score after the last estimated training epoch (orange diamond); the same ensemble trained with early stopping (thin).

with English Span and Span/Subtype classifiers, a higher number of LSTM layers and more hidden units did not seem to influence performance much, contrary to our expectations.

We take a closer look at the effects of ensembling. We compute all ensembles over five joint models and plot their F1-scores as the function of minimal support (Figure 3). As expected, ensembles with support values two and three are consistently better than the average single model, but gains are small for Spanish, whose models vary a lot in performance.

Next, we consider how well we estimate the number of epochs for single models in Run 1 (Figure 4). Whereas Spanish shows an expected learning curve, English and Chinese quickly reach a high test set performance and then fluctuate strongly within some constant range. Therefore, estimating an optimal fixed number of learning epochs from the development set is difficult. Our motivation for the Run 1 strategy was that Spanish and Chinese training data are scarce and so not losing data on a development set would likely help. Both the Chinese results in Figure 4 and our official results for Chinese indicate that using all labeled data does not compensate for a suboptimal choice of training termination, which is very likely with the fluctuation that we observe. We leave it to future work to establish whether this problem could be resolved by changing the training.

10 Related Work

Detecting and correctly classifying expressions that trigger Events is central to event extraction. The traditional solution to the EN Task

is a pipeline of classifiers, one for each of the Event Nugget properties (Span, Subtype, Realis) trained on ground truth and propagating their labels down the pipeline at prediction time (Reimers and Gurevych, 2015; Ahn, 2006). A common variation to this is to directly predict Subtypes plus the NULL-Subtype and then deterministically derive Spans from the predicted Subtype labels (Lu and Ng, 2016; Nguyen et al., 2016b). As for features, pre-trained word vectors and RNNs have long been successfully applied to this task (Reimers and Gurevych, 2015).

There has been a lot of interest in applying joint learning to event extraction (Li et al., 2013; Araki and Mitamura, 2015; Nguyen et al., 2016a; Yang and Mitchell, 2016). The detection of Event Nugget Spans and Subtypes benefits from the joint event extraction and argument identification and classification. To the best of our knowledge, we are the first to propose a joint model for the whole of the EN task.

11 Conclusion

We present our structured prediction RNN approach to the trilingual TAC KBP 2017 Event Nugget detection task. Our systems produce the best micro-average F1-scores for the overall (Subtype+Realis) task in all the languages: English (39.7), Spanish (31.0), and Chinese (40.0). Our contribution is a model that jointly predicts the structured Subtype–Realis label for each token, but independently of the labels of other tokens. Global context enters our classifier mainly via the bidirectional LSTM encoding of the sentence. We show that the softmax-margin loss with extra costs for recall errors is an effective objective, given that Event tokens are highly infrequent. We gain consistent increases in performance by model ensembling. Although we employ morpho-syntactic features, our post-submission experiments indicate that they largely do not help.

Acknowledgements

We would like to thank Luzia Roth and Tilia Ellendorff for their work. Peter Makarov has been supported by European Research Council Grant No. 338875.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*. ACL.

Jun Araki and Teruko Mitamura. 2015. Joint event trigger identification and event coreference resolution with structured perceptron. In *EMNLP*.

Cristian Cardellino. 2016. Spanish Billion Words Corpus and Embeddings. http://crscardellino.me/SBWCE/.

Kevin Gimpel and Noah A Smith. 2010a. Softmax-margin CRFs: Training log-linear models with cost functions. In *NAACL-HLT*.

Kevin Gimpel and Noah A Smith. 2010b. Softmax-margin training for structured log-linear models. Technical report.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18(5).

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2017. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems* .

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL-JCNLP*.

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

Jing Lu and Vincent Ng. 2016. UTD's event nugget detection and coreference system at KBP 2016. In *Proceedings of the Ninth Text Analysis Conference*.

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*.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *CoNLL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016a. Joint event extraction via recurrent neural networks. In *HLT-NAACL*.

Thien Huu Nguyen, Adam Meyers, and Ralph Grishman. 2016b. New York University 2016 system for KBP event nugget: A deep learning approach. In *Proceedings of Ninth Text Analysis Conference*.

Nils Reimers and Iryna Gurevych. 2015. Event nugget detection, classification and coreference resolution using deep neural networks and gradient boosted decision trees. In *Proceedings of the Eighth Text Analysis Conference*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *EMNLP*.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *NAACL-HLT*.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .