HITS at TAC KBP 2015: Entity Discovery and Linking, and Event Nugget Detection

Benjamin Heinzerling[†]
AIPHES[*], HITS

Alex Judea[‡]
HITS

Michael Strube
HITS

Heidelberg Institute for
Theoretical Studies gGmbH (HITS)
Schloss-Wolfsbrunnenweg 35
69118 Heidelberg, Germany
{benjamin.heinzerling|alex.judea|michael.strube}@h-its.org

Abstract

HITS participated in the English Entity Linking and Discovery (EDL) and Event Nugget Detection (EN task 1) tracks at TAC KBP 2015. Our EDL system introduces a novel, interleaved multitasking approach, which allows interaction between interdependent entity linking subtasks while avoiding the structural and algorithmic complexity of joint models. Out of the eight systems that participated in the monolingual English EDL evaluation, our system is ranked first in linking and second in clustering.

HITS also participated in the Event Nugget Detection task. Our approach is based on a structured perceptron which predicts entity mentions, event triggers, and realis jointly. We generalize to unknown triggers with hidden units, a concept used by the frame-semantic parser SEMAFOR.

1 EDL

1.1 Introduction

HITS' entity discovery and linking (EDL) system at TAC KBP 2015 (Ji et al., 2015) introduces a new architecture that performs all EDL subtasks via interleaved multitasking. Switching freely between interdependent EDL subtasks, our system captures interactions that are missed by common pipeline approaches, while avoiding the structural and algorithmic complexity of joint models.

Interleaved multitasking is implemented in the form of sieves, or unsupervised rules, that take local or pairwise decisions in order of precision. The modular nature of this approach allows us to easily integrate the system we submitted at previous TAC KBP EDL tracks (Fahrni et al., 2014; Judea et al., 2014), creating a hybrid system that combines local, unsupervised sieves with global, supervised, joint disambiguation and NIL clustering.

1.2 Interleaved Multitasking

The core of HITS' EDL system is a pipeline of sieves that perform the interdependent subtasks necessary for EDL, i.e., mention detection, entity linking, NIL classification, and NIL clustering. The sieves are ordered in descending order of precision and according to dependency on results of previous sieves. This architecture follows similar approaches in coreference resolution (Lee et al., 2011; Raghunathan et al., 2010). However, unlike coreference sieves, which are all designed for a single task, our approach freely interleaves sieves for multiple tasks. Sieves vary in scope: Some may only perform a very specific part of a certain subtask, e.g., split a multiword mention in two, while others may perform several subtasks jointly, such as detecting and clustering mentions of NIL persons.

Sieve-based approaches in coreference first detect mentions in a separate step, and then resolve coreference relations between the detected mentions. Since mention detection and disambiguation are interde-

[†]Corresponding author for EDL participation.
[*]Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) funded by DFG under grant No. GRK 1994/1.
[‡]Corresponding author for EN task 1 participation.

pendent EDL subtasks that benefit from joint inference (Sil and Yates, 2013; Luo et al., 2015), our approach integrates mention detection into the interleaved multitasking framework. Because we need to first detect mentions before they can be disambiguated, but also aim to base mention detection decisions on as much contextual information – including already disambiguated mentions – as possible, we are faced with a chicken-and-egg problem. To solve this problem, we employ bootstrapping by first performing high-precision mention detection and disambiguation, whose results will serve as seeds for the iterative application of subsequent sieves.

1.2.1 High-Precision Seeds

The purpose of the first group of sieves in our pipeline is to find high-precision mentions and KB links, which will serve as seeds for decisions by subsequent sieves. These seeds are found by identifying (almost) unambiguous mentions , which are then verified through entity typing and information in the KB.

Unambiguous CrossWiki mention: Having run NER, annotate each recognized named entity (NE) mention with the sense found in the CrossWiki dictionary, if that sense has a conditional probability > 0.95 that was obtained from at least 10 non-Wikipedia links.

Sense label mismatch filter: Undo any sense annotation for which none of the known surface forms, i.e. rdfs:label and basekb:common.topic.alias values found in the KB, matches the corresponding mention text. This effectively removes erroneous links introduced by previous sieves, at the cost of also removing some correct links.

Entity type filter: Undo any sense annotation whose NE type, as determined by CoreNLP, does not agree with the rdf:type values found in the reference KB. This filtering is applied to linked named entities of type PERSON, LOCATION, and ORGANIZATION.

The combination of these three sieves achieved a `strong_link_match` precision > 0.9 on our development set.

1.2.2 Person Names

Simple visual error analysis (Heinzerling and Strube, 2015) of our previous years' global disambiguation system showed that it does not deal well with ambiguous intra-document coreference of partial person names, e.g., a surname occurring after two or more persons with the same surname were mentioned. The following sieves aim at disambiguating such cases:

Possible genders: Annotate each person mention with its compatible semantic genders using the following method:

1. For non-NIL person mentions already linked, there is only one possible gender, which can easily be queried from the KB.

2. For all remaining person mentions, assign semantic gender according to preceding gender markers such as *Mr*, *Ms*, or *Lady*.

3. For all remaining person mentions, mark gender using CoreNLP's gender annotator. Do not override gender annotations from the previous step, as gender-specific salutations and other gender markers are not taken into account by CoreNLP's gender annotator.

4. All remaining person mentions not covered by any of the previous methods are marked as compatible with both female and male semantic gender.

Person name matching: Match first- or surname-only person name mentions with their unambiguous full name antecedent, taking gender compatibility into account.

Generic name mention removal: Remove plural forms of known surnames, e.g. *the Steenkamps*, and family references such as *the Steenkamp family*, as these are not annotated in the gold data.

1.2.3 Salient Semantic Paths

Global disambiguation methods employ measures of semantic relatedness for collectively re-ranking candidate senses of given mentions. While linguistically appealing, these methods suffers from high complexity: The combinatorial explosion resulting from considering the possible combinations of all candidate senses for all mentions renders exact

global inference infeasible. Prior work has tackled this problem through approximative inference by: factorizing the set of all mentions into small "connected components" that model relevant context (Sil and Yates, 2013); identifying "collaborators" that are most likely to help disambiguate a given mention (Chen and Ji, 2011; Pan et al., 2015); or using graph-based methods (Han et al., 2011; Hoffart et al., 2011; Moro et al., 2014).

In contrast to these approaches, which, somewhat simplifying, fix two candidate entities and then compute their semantic relatedness, we fix one argument and a relation type, and then look for a matching second argument in the input document. More concretely, we first compile a set of paths in the KB graph that connect two entity types of interest. These paths generally depend on the corpus genre, but are independent of the current input document. Following these paths for each linked mention of matching entity type, we query related entities in the KB, and then check if known surface forms of any of the related entities occur in the text. For example, having already disambiguated a non-NIL person mention, we check if any of this person's children are mentioned in the text and then assign the corresponding KB IDs.

Since the TAC 2015 EDL corpus is relatively small and limited to the news genre, we manually compiled a list of salient paths. A more principled method would be to obtain paths from annotated data, which we leave to future work. We find entities that are semantically related to entities already identified along the following paths:

- **Geographical containment:** Transitive closure of basekb:location.location.containedby (Figure 1a). We found that YAGO offers a richer inventory of geographical information,[1] and hence also take the transitive closure of the corresponding YAGO entry's <isLocatedIn> predicate.[2]
- **Children:** basekb:people.person.children (Figure 1b).
- **Aliases or nicknames:** basekb:common.topic.alias (Figure 1c).
- **Party affiliation:** For all known politicians, i.e., all persons with a corresponding basekb:people.person.profession value, query the path basekb:government.politician.party → basekb:government.political_party_tenure.party, which connects politicians to political parties via a compound value type (CVT).

1.2.4 Joint global disambiguation and NIL clustering

At this point, we run our last years' EDL system, which jointly performs global disambiguation and NIL clustering. The system is based on a Markov Logic Network (MLN) whose weights were trained on 500 Wikipedia articles. For a detailed description see (Fahrni et al., 2014). Mentions linked by upstream sieves are provided to the MLN as ground truth, which brings the advantages of reducing the problem space, and, more importantly, providing additional contextual information.

1.2.5 General post-processing

After global disambiguation and NIL clustering, we apply several post-processing heuristics that are mainly designed to increase linking recall and improve NIL clustering performance.

Most frequent sense fallback: For all low-confidence NIL mentions,[3] assign the most frequent sense according to intra-Wikipedia article links, if that sense is dominant, i.e., its mention-sense prior probability exceeds a threshold tuned on our development set. While this is a crude method, the resulting loss in linking precision is outweighed by increased recall, leading to an increase of linking F1.

First token unification: For all multi-token mentions whose first token is not a frequent word, unify, i.e., set to the same sense, all matching tokens that have not been handled by an upstream sieve.

Abbreviations: For all multi-token mentions, generate abbreviation strings and unify matching tokens.

[1] Presumably due to its inclusion of the `https://www.geonames.org` database.

[2] The transitive closure for a given predicate in the reference KB can be queried via SPARQL property paths. For YAGO, a recursive SQL query is required.

[3] Low-confidence NIL mentions are all recognized NEs that have not been linked or explicitly classified as NIL by an upstream component.

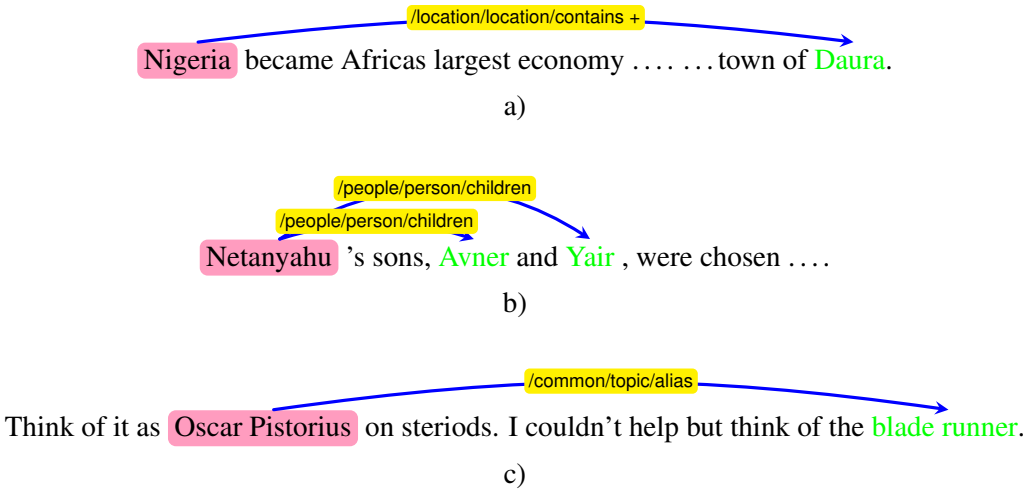

Figure 1: Using salient semantic paths for mention detection and linking. Pink highlights indicate mentions that are already linked to the KB, yellow arrow labels are salient semantic paths in Freebase predicate notation, and green text shows string matches for known surface forms of related entities.

Country adjectivals: Map any occurrence of country adjectivals to their corresponding country.[4]

1.2.6 Post-processing specific to TAC KBP 2015

Post authors: Apply a regular expression match for post author XML attributes in discussion forum text and perform string match for known post authors in the document text, if the post author's name is not one of the most frequent English words.

Media and news organization removal: Remove any linked mention whose basekb:common.topic.notable_types include any of /tv/tv_network, /broadcast/radio_network, or /book/newspaper. Since the notable_types available in the KB do not cover all media and news organizations, this sieve is also applied if the first sentence of the corresponding basekb:common.topic.description contains a noun phrase such as *news website* or *television network*.

1.3 Implementation and Resources

HITS' EDL system is implemented as a UIMA pipeline (Ferrucci and Lally, 2004), using the Stanford CoreNLP (Manning et al., 2014) UIMA components provided by DKPro (Gurevych et al., 2007) for text segmentation, POS tagging, and named entity recognition, and DKPro WSD (Miller et al., 2013) for modeling entity mentions and links. In addition to the reference knowledge base (KB) provided by the organizers, our system makes use of the following resources:

- CrossWiki (Spitkovsky and Chang, 2012), as an inventory of unambiguous surface forms;
- YAGO (Suchanek et al., 2007), as a gazetteer;
- Wikipedia (2013 dump), for most-frequent-sense statistics collected from inter-article links; and
- Word lists such as demonyms and gender-specific salutations.

1.4 Results

HITS submitted four runs, whose settings are described in Table 1. Table 2 shows each run's performance for the three main metrics. Since we only processed the English subset of the trilingual queries, only monolingual English results are reported. We did not resolve nominal coreference, i.e. NOM queries, as any attempt to do so degraded overall performance. Due to time constraints, we only submitted the output of CoreNLP's 3-class NER and 3-class type information from the KB as entity type, which explains our low ranking in terms of the `strong_typed_mention_match` metric.

[4]This mapping is derived from: `https://en.wikipedia.org/wiki/List_of_adjectival_and_demonymic_forms_for_countries_and_nations`

Run	Setting
HITS1	Sieves, remove low-conf. NILs
HITS2	Sieves, MLN, remove low-conf. NILs
HITS3	Sieves, retain low-conf. NILs
HITS4	Sieves, MLN, retain low-conf. NILs

Table 1: Settings for the four runs we submitted. *Sieves* means that all sieves were applied, *MLN* that our previous years' MLN-based system was run, and removal or retention of *low-conf. NILs* specifies whether low-confidence NILs were retained in an attempt to boost recall, or removed in order to increase precision.

For linking, our system is ranked first, and for clustering second, with only 0.002 F1 points difference to the first-ranked system.

1.5 Discussion

Our original intention in devising the sieves was to let them deal with some of the easy decisions, thereby reducing the remaining search space and providing more contextual information to HITS' more sophisticated, MLN-based global joint disambiguation and clustering system. As such, we were surprised by the fact that the sieves account for almost all of the linking and clustering performance. A possible explanation is the fact that the MLN-based system was trained on a different corpus, while the sieves are tailored to this years' training data, namely newswire texts and discussions of news-related topics. Furthermore, there is a certain amount of overlap between the information used by the sieves, and the features collected by the MLN-based system. However, even in comparison to the systems of other teams, the sieves-only setting (HITS1), achieves the same ranking in terms of `strong_link_match` and `mention_ceaf` as our best run (HITS2), which combines sieves and the MLN-based system.

The sieve-only approach is, given NER and gender annotations, essentially a deterministic sequence of database queries and string matching operations, It would be interesting to examine the mistakes this approach commits in comparison to other, possibly more sophisticated systems. The top three `strong_link_match` and top two `mention_ceaf` systems achieve similar performance. Differing mistakes would suggest that other systems can benefit from incorporating sieves similar to ours, while finding largely the same mistakes would mean that state-of-the-art monolingual mention detection, entity linking, and NIL clustering performance can be achieved by database queries and string matching.

1.6 Conclusions

By using interleaved multitasking, HITS' EDL system is able to capture a high degree of interaction between the EDL subtasks, namely mention detection, entity linking, NIL classification, and NIL clustering. Because it is still a pipeline approach, interleaved multitasking avoids the structural and algorithmic complexity of joint models. A modular implementation allowed us to easily integrate our previous years' system, thereby combining unsupervised local disambiguation stages with a supervised, joint global disambiguation and NIL clustering component. In the English EDL evaluation, HITS' system achieved the overall best linking performance and was ranked second in clustering.

2 EN Task 1: Event Nugget Detection

HITS' Event Nugget Detection system is based on a structured perceptron. The system performs segmentation, classification, and realis status prediction simultaniously. It segments a sentence into entity mentions and event triggers[5], assigns entity type and event type labels to those segments, and predicts the realis status of triggers.

The system is based on the structured perceptron. Li et al. (2014) proved the effectiveness of a structured perceptron approach for entity mention and event nugget detection. In addition to their features for nugget prediction, we treat nuggets seen during training as prototypes for the events they triggered. This way we try to further generalize to triggers never seen during training. This idea was introduced in SEMAFOR, a frame-semantic parser (Das et al., 2014).

We pool all documents from ACE 2005 and TAC 2015. The system learns on this document pool. From TAC documents it learns to predict event trig-

[5]In this paper, we will use the terms *trigger* and *nugget* interchangeably

Run	NER			Linking			Clustering		
	P	R	F1	P	R	F1	P	R	F1
HITS1	**0.629**	0.514	0.566	**0.707**	0.578	0.636	0.747	0.610	0.671
HITS2	0.627	0.525	0.571	0.703	0.588	**0.640**	**0.748**	0.626	**0.682**
HITS3	0.614	**0.540**	**0.574**	0.673	**0.592**	0.630	0.719	**0.633**	0.673
HITS4	0.617	0.536	**0.574**	0.670	0.582	0.623	0.709	0.616	0.659
Best	0.791	0.673	0.727	0.703	0.588	0.640	0.765	0.619	0.684

Table 2: HITS' EDL performance for English. *NER* shows `strong_typed_mention_match` results, *Linking* is `strong_mention_match`, and *Clustering* `mention_ceaf`. Bold font denotes the highest value achieved among our runs. *Best* shows the overall best result for each metric.

gers, their types and the respective realis status. For ACE documents, learning is selective. We can choose to learn the prediction of entity mentions only, or we can use the event annotations to have more training data for nugget detection.

We now describe the system in more details. See also Algorithm 1.

2.1 Approach

Our system consists of two parts. The first part is segmentation of a sentence into entity mentions and event triggers, and classification of those segments. The second part is realis prediction of each nugget. Both parts are carried out simultaniously. We now describe each part in detail.

2.1.1 Segmentation and Classification

The first part, *segmentation and classification*, operates intra-sentential. It is based on the idea of semi-Markov chains (Sarawagi and Cohen, 2004). In the standard segmentation setting (conditional random fields, hidden markov models) individual tokens are labeled. For a sentence $x = x_1 \dots x_n$, each token x_i is assigned a label, e.g. using the BILOU schema. Features of x_i usually cover sourrounding tokens in a limited (and often fix-sized) window, but they cannot incorporate features describing the entire segment of interest.

In contrast, our segmenter operates on segments of arbitrary size. It is not limited to single tokens. Instead of labeling each t_i, we look for the best segmentation $u = u_1 \dots u_n$ of x, consisting of segments $u_e = (b, e, t)$, where b and e are begin and end indices, and t is either an entity or event type from the allowed types $\mathbf{T}$, or none of the two.

We start at the first position with a segments of length 1. We generate segments $u_1 = (1, 1, t), \forall t \in \mathbf{T}$ and rank them according to their score. We move to second position. Here, we generate segments of different types up to length 2 and rank them. For the third position, we generate segments of different types up to length 3 and rank them. We do this until the end of the sentence. As the system moves to the next position, it keeps track of all segmentation histories and their scores[6].

To find the best sequence of segments, we compute a score for each segment, which is the weighted sum of its features. The score s for a specific segment $u_e$ending at position e is defined as follows.

In contrast, our segmenter is not limited to single tokens and fixed context windows. It operates on segments of arbitrary size. Instead of labeling each x_i, we look for the best segmentation $u = u_1 \dots u_n$ of x, consisting of segments $u_e = (b, e, t)$, where b and e are begin and end indices of tokens, and t is an entity or event type from the set of allowed types $\mathbf{T}$. Note that $\mathbf{T}$ includes a 'null class'.

We start at the first position with segments of length 1. We generate segments $u_1 = (1, 1, t), \forall t \in \mathbf{T}$ and rank them according to their score. We move to second position. Here, we generate segments of different types up to length 2 and rank them. For the third position, we generate segments of different types up to length 3 and rank them. We do this until the end of the sentence. As the system moves to the next position, it keeps track of all segmentation histories and their scores[7]. What the system effectively does is to pair all possible segments at each position

[6]The score of a history is the sum of its segment scores.

[7]The score of a history is the sum of its segment scores.

with all segmentation possibilities at previous positions in order to find the best sentence segmentation in the end.

To find the best segmentation, we compute a score for each segment, which is the weighted sum of its features. The score s for a specific segment u_e ending at position e is defined as

$$s(u_e) = \mathbf{f}(x, u) \cdot \mathbf{w}. \quad (1)$$

Note that the null class always has score 0.

In order to learn the weights $\mathbf{w}$ we use the structured perceptron framework. Formally, for each training instance $(x, \hat{u})$, where $\hat{u}$ is the gold segmentation, we compute the best segmentation z according to the model.

$$z = \arg\max_{u' \in \mathcal{U}(x)} \mathbf{f}(x, u') \cdot \mathbf{w} \quad (2)$$

If $z \neq \hat{u}$, we update $\mathbf{w}$.

$$\mathbf{w}^{\text{new}} = \mathbf{w} + \mathbf{f}(x, \hat{u}) - \mathbf{f}(x, z) \quad (3)$$

To cope with unknown triggers the model includes a latent variable iterating over triggers seen in training (called hidden units, Das et al. (2014)). At inference time, hidden units serve as prototypes for unknown words. If u' has an event type, the feature function factorizes as follows.

$$\mathbf{f}(x, u) = \sum_{h \in H_t} \mathbf{f}(x, u, h) + \mathbf{f}(x, u) \quad (4)$$

Here, h is a hidden unit, H_t are all hidden units for event type t, and $\mathbf{f}(x, u, h)$ are features for hidden units.

For TAC documents, we train on event segments. For ACE documents, we can choose if we want to consider only segments with an entity type or only segments with an event type, or both for training.

Finally, we make use of parameter averaging (Collins, 2002). We do not use the last version of $\mathbf{w}$ after examining n training instances, but the average of all versions of it, that is, the average of all $\mathbf{w}^0 \cdots \mathbf{w}^n$.

2.1.2 Approximating the Search

Explicitely going through all possible segmentations of a sentence would be costly. Furthermore, it makes no sense to generate very long segments, since most entity mentions and event nuggets are rather short. To account for the latter, we restrict the maximum size of segments. Each type has a maximum length[8], and we set a global maximum length of 3. To account for the former, we approximate the search.

Following (Li et al., 2014) we use beam search to approximate the search. At each position, we generate all valid segmentations and rank them. Instead of keeping the entire list, we only keep the k best segmentations. We set k to 3.

(Huang et al., 2012) proved that performing the standard perceptron update with inexact search leads to invalid updates of the weight vector which in turn decrease performance. If $z \neq \hat{u}$, it may be that $\hat{u}$ is not predictable by the model (because it is not in the beam) but has a higher score than z according to the model. In the standard case, we would update the weights even though they are correct. The error was introduced by the search procedure, not by the model. Performing such invalid updates often leads to decreased performance.

Huang et al. (2012) proved that performing the standard perceptron update with inexact search leads to invalid updates of the weight vector which in turn decrease performance. If $z \neq \hat{u}$, it may be that $\hat{u}$ is not predictable by the model (because it is not in the beam) but has a higher score than z according to the model. In the standard case, we would update the weights even though they are correct. The error we have was introduced by the search procedure, not by the model. Performing such invalid updates may push the weight vector into a less god direction.

One technique to avoid invalid updates is *early update*. For our system, early update is performed as follows. We predict a partial segmentation z' including all segments up to a certain position. If z' is not a prefix of the true segmentation $\hat{u}$, we perform the update immediately and continue with another sentence. We compute a partial segmentation for each position in the sentence.

2.1.3 Realis Status

The second part of our system is *realis prediction*, carried simultaneously to the other tasks. We iterate through all (partial) segmentations in our beam. For

[8] We picked the maximum length so that more than 95% of the cases in the training data are covered.

each trigger segment, we iterate through all realis values for it, score them using the same procedure as for scoring segments (weighted sum of features), and pick the one with the highest score.

2.2 Features

We define two types of features. Local features operate only on a segment and its context. Global features operate on other segments as well. Local features capture properties of segments, e.g., the text they cover, the part-of-speech of their head noun, or its context words. Global features capture properties of the sequence derived so far, e.g., they assign a weight to the decision of having two DIE event triggers in one sentence.

Table 3 reports event nugget detection features, including hidden unit features. For each segment u_e we extract two versions of features. In the first, features are extracted for event subtypes, e.g. CONFLICT.ATTACK. In the second, features are extracted for event supertypes, e.g., CONFLICT. The intuition for this is that certain properties are shared across supertypes, e.g., across JUSTICE events, captured by the second feature version. We always extract both versions, so each feature is concatenated with an event supertype and with an event subtype. Our feature set is similar to the set in (Li et al., 2013).

Table 4 reports realis features. Our feature set is fairly simple. It characterizes the trigger and its lexical and syntactic context, as well as the status of previous triggers in the sentence, if any.

Some features make use of word clusters in order to increase generalization. We learned word clusters first training word2vec (Mikolov et al., 2013) on a portion of Gigaword, and then applying k-means clustering to the vectors, with $k = 500$.

2.3 Runs

We submitted three runs to TAC 2015. Table 5 reports results. We now give details to the configuration of each run. We describe only the differences to the previous run.

HITS1. Beam size was 3, learning rate of the perceptron 0.5. We trained for 10 epochs on ACE and TAC documents, without using hidden unit features.

HITS2. Trained for 20 epochs, used hidden unit features and trained nugget detection on TAC documents only.

HITS3. Here, we skipped potential triggers we have never seen during training

The major variation in our runs is whether we use hidden unit features or not (HITS1 vs. HITS2 and HITS3). Evaluation suggests that using hidden unit features greatly increases recall, while decreasing precision, leading to a higher F_1 score. Skipping potential triggers we have never seen during training increases precision again a bit, but lowers recall too much, resulting in a lower F_1 compared to not restricting the set of potential triggers.

2.4 Conclusions

For most evaluation metrics, our system ranked 8th of 14. Regarding runs, HITS2 was our best, i.e., training on TAC documents only and using hidden units as additional features. This suggests that hidden units have the potential to capture triggers unseen during training. However, more experiments in this regard have to be carried out. It is necessary to test hidden units on the same training corpus, using the same beam size and the same number of training iterations.

We also adopted a fairly agressive update policy: We carry out the early update as soon as the top segmentation is not the correct one. This leads to decreased training time, but may impact results negatively. Further experiments have to explore the traditional early update policy: Update when the correct segmentation falls out of the beam.

Finally, the structured perceptron approach is capable of predicting event arguments jointly with mentions, triggers, and the realis status. Related work showed that joint trigger and argument prediction leads to increased performance for both substasks.

Acknowledgments

Benjamin Heinzerling has been supported by the German Research Foundation as part of the Research Training Group “Adaptive Preparation of Information from Heterogeneous Sources” (AIPHES) under grant No. GRK 1994/1, and partially funded by the Klaus Tschira Foundation, Heidelberg, Germany. Alex Judea has been supported by a HITS Ph.D. scholarship.

Lexical	unigrams/bigrams in u_e, as well as in a context window of size 2
	unigrams/bigrams of part-of-speech tags in u_e, as well as in a context window of size 2
	lemmas of words in u_e
	word cluster of current word
Syntactic	dependent and governor words, and depdency types of words in u_e
	for every word in u_e: is it a modifier of a job title?
Entities	unigrams/bigrams in a context window of size 2, concatenated by the overlapping entity type
	dependency features, concatenated by the overlapping entity type
	nearest entity type and string in the sentence and clause
Hidden units	trigger and h have WordNet relation: indicator feature, relation, lemmas of h
	lemma-POS-sequence of h equals that of trigger: indicator feature, lemma-POS-sequence
	h equals trigger: indicator feature, token sequence

Table 3: Nugget detection features. u_e is a segment ending at position e in the sentence, h is a hidden unit, i.e. a trigger for a specific event type seen during training.

Lexical	unigrams/bigrams in a window of size 2 around the nugget
	subtype/supertype, text, part-of-speech sequence of the nugget
	string, word cluster of nugget head
Syntactic	dependency parent and dependency children strings
Nuggets	realis status and string of last nugget, if any

Table 4: Realis features.

Run	Attributes	precision	recall	F_1
HITS1	plain	89.0	35.9	51.2
	mention type	82.9	33.5	47.7
	realis status	62.6	25.3	36.0
	mention type + realis status	57.8	23.4	33.3
HITS2	plain	66.0	50.7	57.4
	mention type	55.4	42.6	48.2
	realis status	42.6	32.8	37.0
	mention type + realis status	35.2	27.0	30.6
HITS3	plain	68.4	44.4	53.8
	mention type	59.6	38.6	46.9
	realis status	45.1	29.3	35.5
	mention type + realis status	38.8	25.1	30.5

Table 5: Evaluation results for the three HITS runs.

Algorithm 1 Algorithm to process a sentence and update the weight vector.

```
function SEGMENT(sentence x = x_1 ... x_n,gold segmentation û of x)
    init beam[1 ... n][beamsize]
    for i = 1 ... n do
        initialize beam_i
        for d = 1 ... global maximum length do
            k = i − d + 1
            beam_k = beam[k]
            for type t ∈ T do
                create new segment u_i = (k, i, t)
                compute features for u_i
                beam_i ← copy(beam_k ∪ u_i)
            end for
        end for
        sort and prune beam_i
        if training then                                        ▷ Early update
            if z not prefix of û then
                w^new ← w + f(x, û) − f(x, z)
                exit procedure
            end if
        end if
        beam ∪ beam_i
    end for
    return top(beam)
end function
```

References

Zheng Chen and Heng Ji. 2011. Collaborative ranking: A case study on entity linking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing,* Edinburgh, Scotland, U.K., 27–29 July 2011, pages 771–781.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing,* Philadelphia, Penn., 6–7 July 2002, pages 1–8.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Angela Fahrni, Benjamin Heinzerling, Thierry Göckel, and Michael Strube. 2014. HITS' monolingual and cross-lingual entity linking system at TAC 2013. In *Proceedings of the Text Analysis Conference,* National Institute of Standards and Technology, Gaithersburg, Maryland, USA, 18–19 November 2013.

David A. Ferrucci and Adam Lally. 2004. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3):327–348.

Iryna Gurevych, Max Mühlhäuser, Christof Müller, Jürgen Steimle, Markus Weimer, and Torsten Zesch. 2007. Darmstadt knowledge processing repository based on uima. In *Proceedings of the First Workshop on Unstructured Information Management Architecture at Biannual Conference of the Society for Computational Linguistics and Language Technology, Tübingen, Germany*, page 89.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: A graph-based method. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Beijing, China, 25–29 July 2011, pages 765–774.

Benjamin Heinzerling and Michael Strube. 2015. Visual error analysis for entity linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics: System Demonstrations,* Beijing, China, 26–31 July 2015, pages 37–42.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing,* Edinburgh, Scotland, U.K., 27–29 July 2011, pages 782–792.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Montréal, Québec, Canada, 3–8 June 2012, pages 142–151.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of tac-kbp2015 tri-lingual entity discovery and linking.

Alex Judea, Benjamin Heinzerling, Angela Fahrni, and Michael Strube. 2014. HITS' monolingual and cross-lingual entity linking system at TAC 2014. In *Proceedings of the Text Analysis Conference*.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Shared Task of the 15th Conference on Computational Natural Language Learning,* Portland, Oreg., 23–24 June 2011, pages 28–34.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Sofia, Bulgaria, 4–9 August 2013, pages 73–82.

Qi Li, Heng Ji, Yu Heng, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 1846–1851.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint named entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* Lisbon, Portugal, 17–21 September 2015, pages 857–867.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

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

Tristan Miller, Nicolai Erbs, Hans-Peter Zorn, Torsten Zesch, and Iryna Gurevych. 2013. Dkpro wsd: A generalized uima-based framework for word sense disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations,* Sofia, Bulgaria, 4–9 August 2013, pages 37–42.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: A unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Denver, Col., 31 May – 5 June 2015.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing,* Cambridge, Mass., 9–11 October 2010, pages 492–501.

Sunita Sarawagi and William W. Cohen. 2004. Semi-Markov conditional random fields for information extraction. In *Proceedings of Advances in Neural Information Processing Systems 17.* Vancouver, B.C., Canada, 13–18 December 2004, pages 1185–1192.

Avirup Sil and Alexander Yates. 2013. Re-ranking for joint named-entity recognition and linking. In *Proceedings of the ACM 22nd Conference on Information and Knowledge Management,* San Francisco, Cal., 27 October – 1 November 2010, pages 2369–2374. ACM.

Valentin I Spitkovsky and Angel X. Chang. 2012. A cross-lingual dictionary for English Wikipedia concepts. In *Proceedings of the 8th International Conference on Language Resources and Evaluation,* Istanbul, Turkey, 21–27 May 2012, pages 3168–3175.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A core of semantic knowledge unifying WordNet and Wikipedia. In *Proceedings of the 16th World Wide Web Conference,* Banff, Canada, 8–12 May, 2007, pages 697–706.