Ghent University
Faculty of Engineering and Architecture
Department of Information Technology

imec
Internet Technology and Data Science Lab

Neural Approaches to Sequence Labeling for Information Extraction

Examination Board:

prof. C. Develder (advisor)
dr. ir. T. Demeester (advisor)
prof. F. De Turck (chair)
prof. T. Dhaene (secretary)
prof. I. Augenstein
prof. K. Demuynck
prof. V. Hoste
prof. S. Van Hoecke

Dissertation for acquiring the degree of
Doctor of Computer Science Engineering

Table of Contents

List of Figures

List of Tables

List of Acronyms

ADE	Adverse Drug Events
AI	Artificial Intelligence
AT	Adversarial Training
AVG	Average
BiLSTM	Bidirectional LSTM
BIO	Beginning, Inside, Outside
CBOW	Continuous Bag-of-Words
CRF	Conditional Random Fields
CNN	Convolutional Neural Network
DREC	Dutch Real Estate Classifieds
EC	Entity Classification
HMM	Hidden Markov Model
LSTM	Long Short-Term Memory network
LTM	Locally Trained Model
MTT	Matrix-Tree Theorem
MLP	Multilayer Perceptron
NER	Named Entity Recognition
NLP	Natural Language Processing
ORG	Organization
PER	Person
POS	Part-of-Speech
RE	Relation Extraction

RNN	Recurrent Neural Network
SVM	Support Vector Machine
TL	Tweet-level LSTM

Samenvatting
– Summary in Dutch –

Artificiële intelligentie (AI) bracht een revolutie teweeg in verschillende aspecten van ons dagelijks leven, met toepassingen zoals spraakherkenning, automatische vertaling, beeldherkenning, zelfrijdende auto's.Voor sommige complexe taken, zoals het bordspel Go en het strategiespel StarCraft, overtroffen deze AI-systemen zelfs mensen. Het algemene doel van AI is om algoritmen en methoden te ontwikkelen waarmee machines taken uitvoeren die onze intelligentie imiteren. Dit omvat redeneren, leren, onthouden, problemen oplossen, etc. Er werden reeds vele methoden voorgesteld om deze intelligentie machines aan te leren. Ze kunnen worden gegroepeerd in statistische methoden, machinaal leren en optimalisatie problemen.

Een groot aantal toepassingen van AI richt zich op technieken om de menselijke taal te leren begrijpen. Tekstuele informatie is zeer nuttig voor veel toepassingen, maar heeft een zeer ongestructureerde vorm en de betekenis ervan is vaak dubbelzinnig en afhankelijk van de context. In dit proefschrift bestuderen we een reeks taken uit het domein van natuurlijke taalverwerking (in het Engels: *natural language processing*, NLP) vanuit een theoretische aanpak en gebruiken computationele technieken voor de automatische analyse en representatie van taal. NLP richt zich op een groot aantal taken waaronder automatisch samenvatten, classificatie van documenten, automatische vertaling, beantwoording van vragen, ... Eerder onderzochte benaderingen gebruiken machinaal leren, heuristische benaderingen en onlangs diepe neurale netwerken.

Het thema van mijn proefschrift is *het verrijken van ongestructureerde tekst met gestructureerde informatie*. Dit omvat de automatische extractie van entiteiten (bijvoorbeeld namen van mensen of organisaties, of soorten kamers in een huis) en relaties tussen deze entiteiten. Deze taken, ook wel *named entity recognition* en *relationship extraction* genoemd, zijn belangrijke NLP-taken en werden reeds vaak bestudeerd in eerder werk. Desalniettemin lieten tekortkomingen van die bestaande methoden nog voldoende ruimte voor verdere fundamentele onderzoeksstappen. Vooral via het gebruik van grote hoeveelheden data en nieuwe modellen zoals diepe neurale netwerken, welke de voorbije jaren een grote impact hebben gehad op

AI en NLP. Mijn onderzoek startte vanuit het stellen van volgende vraag: *kan men schematisch een huis reconstrueren, i.e., voorspellen hoeveel kamers een huis heeft, waar deze zich bevinden, etc. louter gebaseerd op een tekstuele beschrijving ervan?* Dit onderzoek was het resultaat van een samenwerking met immowebsite Realo en gaf aanleiding tot verschillende fundamentele onderzoekpistes, welke hebben geresulteerd in dit proefschrift. Na het vergelijken van klassieke methodes van machinaal leren (Hoofdstuk 2) en diepe neurale netwerken voor verrijking van tekst, heb ik me gericht op het verder ontwikkelen van diepe neurale netwerken (Hoofdstuk 3). Het toepassingsgebied van mijn onderzoek ontwikkelde zich daarna ook verder voor andere domeinen (Hoofdstukken 4A - 4B), en het extraheren van gebeurtenissen uit sociale media tekst (Hoofdstuk 5).

De hoofdstukken, samen met de belangrijkste bijdragen, in chronologische volgorde van mijn thesis zijn als volgt:

In Hoofdstuk 1 geven we een kort overzicht van de bestaande literatuur en introduceren we de technieken en applicaties om de lezer in staat te stellen de termen te begrijpen die in de volgende hoofdstukken worden beschreven.

In Hoofdstuk 2 definiëren we de eerder vermelde taak van structureren van tekst over vastgoed: de generatie van een gestructureerde weergave van het vastgoed, verder ook wel de *property tree* genoemd, enkel gebaseerd op de beschrijving in natuurlijke taal van het onroerend goed. Specifieker; (i) we beschrijven het verzamelen en annoteren van een grote hoeveelheid tekst voor de taak van informatie extractie, (ii) we introduceren drie methoden voor het implementeren van de toepassing, en (iii) we voeren een vergelijkende studie uit van de voorgestelde oplossingen op een nieuwe, geannoteerde dataset. We verdelen het probleem van het transformeren van tekst naar een hiërarchische structuur in drie eenvoudigere subtaken, namelijk (1) herkenning van entiteiten in de tekst (waaronder vloeren, kamers, gedeelten binnen kamers, etc.), (2) *dependency parsing* om relaties tussen de betrokken entiteiten te voorspellen, en (3) constructie van de *property tree*.

In Hoofdstuk 3 presenteren we een nieuw neuraal netwerkmodel dat *gezamenlijk* de twee kern-subtaken (*named entity recognition* en *dependency parsing*, zoals eerder besproken in Hoofdstuk 2) uitvoert voor de vastgoedtekst. Het doel van ons model is om de verschillende tekortkomingen van de traditionele pijplijnmethoden, gepresenteerd in Hoofdstuk 2, aan te pakken, zoals (i) propagatie van fouten, en (ii) onbenutte interacties tussen de subtaken. Tenslotte vergelijken we uitgebreid de prestaties van de pijplijnmethoden met het voorgestelde gezamenlijke model en rapporteren we een grote verbetering van de nieuw geïntroduceerde methode.

In Hoofdstuk 4A introduceren we een nieuw neuraal netwerkmodel dat (i) de twee taken van *named entity recognition* en relatie-extractie gelijktijdig uitvoert, en (ii) de complexiteit van het model gepresenteerd in

Hoofdstuk 3 reduceert. Ons model toont een verbeterde prestatie ten opzichte van de nieuwste *state-of-the-art* methoden voor een aantal verschillende domeinen (i.e., nieuws, biomedisch, onroerend goed) en talen (i.e., Engels, Nederlands) zonder het gebruik van handmatig ontworpen features of extra NLP-hulpmiddelen. In Hoofdstuk 4B leggen we uit hoe *adverserial training* kan worden toegepast op het gezamenlijke model (zoals beschreven in Hoofdstuk 4A) om de performantie van de *named entity recognition* en relatie-extractie verder te verbeteren. De belangrijkste bijdrage van dit hoofdstuk is het gebruik van adverserial training als een uitbreiding van de optimalisatie van de gezamenlijke extractietaak.

In tegenstelling tot eerdere hoofdstukken die zich concentreerden op de taak van named entity recognition en relatie-extractie, bespreekt Hoofdstuk 5 verbeterde methoden voor de detectie van gebeurtenissen in tekst op sociale media. We bestuderen deze taak als het labelen van sequenties (vergelijkbaar met named entity recognition). Ons onderzoek lost veel van de problemen die in eerdere studies werden geïdentificeerd, op. Meer concreet, ons model (i) brengt de sequentiële aard van tweets in rekening en (ii) gebruikt informatie uit eerdere tweets voor het voorspellen van het type van een gebeurtenis. We voeren ook hier een uitgebreide experimentele studie uit, die het voordeel van sequentie-modellering voor gebeurtenisdetectie in sport-Twitter-streams aantoont.

Tot slot, in Hoofdstuk 6, vatten we de bijdragen van het proefschrift samen en beschrijven we mogelijk pistes voor toekomstig onderzoek.

Summary

Artificial intelligence (AI) has revolutionized several aspects of our daily lives, with applications in speech recognition, machine translation, image recognition, autonomously driving cars, etc. It has even outperformed humans in complex tasks, such as the board game Go and the StarCraft strategy game. The general goal of AI is to develop algorithms and methods that make machines perform tasks imitating human intelligence. Human intelligence demonstrates specific characteristics such as reasoning, learning, memorization, problem solving, etc. Several methods have been proposed to tackle various AI tasks. These can be grouped into statistical methods, machine learning algorithms, and optimization approaches.

A large number of applications in AI focus on tasks that involve the understanding of human language. This is due to the fact that textual information is highly useful for many applications but it has a highly unstructured form and its meaning is often ambiguous and context-dependent. In this thesis, we study tasks from the Natural Language Processing (NLP) field (a sub-field of AI) which includes a theory-motivated range of computational techniques for the automatic analysis and representation of human language. NLP focuses on language comprehension in a number of tasks that include (but are not limited) to text summarization, document classification, machine translation, question answering, etc. The approaches proposed in literature include classical machine learning methods, heuristic approaches and neural network architectures.

The overall theme of my dissertation is in *enriching sequences of text*, in particular with indications of pre-defined entities (for example, names of people or organizations, or types of rooms in a house) and relations between these. These tasks, called *named entity recognition* and *relation extraction*, are core NLP tasks, and they have seen a lot of research in the past. However, a number of issues with existing methods left several opportunities for fundamental further research steps. Emerging techniques in deep neural networks have made that possible over the past few years. My story starts off with finding the best possible solution to the following research problem: *can we schematically reconstruct a house, in terms of which rooms it contains, or where these are located, merely based on a textual description?* This research question actually originates from a collaborative project with the Flemish company Realo. However, it opened up several interesting re-

search directions, which have resulted in the underlying dissertation. After comparing classical machine learning approaches (Chapter 2) and deep neural networks for the real-estate task, I focused on further developing the latter technology (Chapter 3). The application area of my research then opened up towards several other application domains (Chapters 4A-4B), and an extension of the entity detection task towards streaming social media data (Chapter 5).

The individual chapters follow the chronological evolution in my research. Here's a short overview, with the main contributions.

In Chapter 1, we provide a brief overview of the previous literature and we focus on techniques and tasks in order to allow the reader to understand terms described in subsequent chapters.

In Chapter 2, we define the aforementioned real-estate task: the prediction of a tree structured representation of a real-estate property, namely the *property tree*, based only on the natural language description of the property. Specifically, we (i) collect and annotate a large amount of data for the structured prediction problem, (ii) introduce three alternative methods for solving the newly defined problem, and (iii) perform a comparative study of the proposed solutions on a newly created and annotated real-world data set. We break down the problem of transforming flat text to hierarchical structures into three simpler subtasks, namely (1) entity recognition of real estate entities (where entities are floors, rooms, sub-spaces in rooms, etc.), (2) dependency parsing to predict the part-of relationships between the involved entities, and (3) construction of the *property tree*.

In Chapter 3 we present a new neural network model that *jointly* performs the two core subtasks (entity recognition and dependency parsing presented in Chapter 2) for the real-estate problem. The contribution of our joint model is to address several shortcomings of the pipeline methods presented in Chapter 2, such as (i) error propagation, and (ii) unexploited interactions between the subtasks. Finally, we extensively compare the performance of the pipeline methods with the proposed joint model, reporting a large improvement of the newly introduced method for our application.

In Chapter 4A, we introduce a new general purpose neural network model that (i) performs the two tasks of entity recognition and relation extraction simultaneously, and (ii) reduces the quadratic complexity of the sequence labeling model presented in Chapter 3. Our model showcases an improved performance over previous state-of-the-art methods in a number of different contexts (i.e., news, biomedical, real estate) and languages (i.e., English, Dutch) without the use of manually engineered features nor additional NLP tools. In Chapter 4B, we explain how adversarial training (AT) can be applied on top of our joint model (as described in Chapter 4A) to improve the performance of the named entity recognition (NER) and relation extraction tasks. The core contribution of this chapter is the use of AT as an extension in the training procedure for the joint extraction task.

Unlike previous chapters that focused on the task of entity recognition and relation extraction, Chapter 5 presents improved methods for the sub-event detection task in social media streams, which we frame as a sequence labeling problem (similar to named entity recognition). Our work overcomes limitations identified in previous studies. Specifically, our model is able to (i) take into account the chronological order of consecutive tweets, and to (ii) exploit information from previous tweets for predicting the presence and the type of a sub-event. We also perform an extensive experimental study, indicating the benefit of sequence labeling for sub-event detection in sports Twitter streams.

Finally, in Chapter 6, we summarize the contributions of the thesis and describe future research directions.

1 Introduction

"The essence of things lies in simplicity and we usually perceive it almost always slowly, absorbed in our complexity."

— Giannis Ritsos

Artificial intelligence (AI) has been applied recently in various application domains. Specifically, in our daily lives one can recognize several applications which involve the use of AI such as targeted advertisement [1], movie/friend recommendation [2], chatbots [3], speech recognition [4], self-driving cars [5], weather forecasting [6], etc. AI systems usually demonstrate behaviors that are related to "human intelligence" (i.e., reasoning, learning, memorization, problem solving). This type of "intelligence" commonly results from data-driven approaches that are based on statistical methods, machine learning algorithms and optimization approaches.

Machine learning is a sub-field of AI that involves methods which learn directly from the data. Specifically, based on the well-known quote defined in Mitchell (1997) [7], a learning algorithm can be described as follows: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." However, the type of task T, experience E, and performance measure P varies for each

application. Thus, the question is which are the appropriate T, E and P for each application domain.

In computer vision applications, such as image recognition, a task T is the classification of an image into a set of pre-defined categories, the experience E is a set of human-annotated images and the performance measure P can be the ratio of the correctly classified images out of total number of images. In Natural Language Processing (NLP) applications, such as in Named Entity Recognition (NER) where the goal is to identify the core entities of a given sentence, the task T is the application itself, the experience E is the set of human-labeled entities in the sentences and P is the performance measure, e.g., the percentage of the correctly identified entities inside the sentences.

In this thesis, we focus on tasks from the NLP area. NLP lies in the intersection of machine learning and computational linguistics (i.e., modeling of natural language). It is a theory-motivated range of computational techniques for the automatic analysis and representation of human language [8]. NLP enables computers to process unstructured text and extract useful (e.g., structured) information. Recent advances in the NLP area strive towards human-level understanding of language by achieving high performance in tasks such as text summarization [9], question answering [10], machine translation [11], etc.

The starting point of this thesis is a particular application aiming at structuring the natural language description (i.e., advertisement) of a real estate property into a tree-format (described in detail in Chapters 2 and 3). Specifically, the goal of this application is (i) the detection of the core parts of the property (e.g., rooms, floors), (ii) identification of the relations between them (e.g., which room is *part-of* a floor), and (iii) creation of a tree structure using the information extracted in (i) and (ii). To tackle this problem, we break down the tasks into a series of two sub-tasks, namely named entity recognition (i.e., identify the core entities of a sentence) and relation extraction (i.e., identify relations between pairs of entities). Our first solution (see Chapter 2) consists of a two-step pipeline method while afterwards we propose a joint model (see Chapter 3) that overcomes the limitations of the pipeline methods. Then, we realize that our proposed methodology is widely applicable and achieves state-of-the-art performance in various settings and languages. Thus, we improved the method presented in Chapter 3 in terms of complexity and robustness (see Chapters 4A and 4B). Finally, we applied the same techniques on sequences of micro-posts instead of applying them on word sequences (see Chapter 5).

In this chapter, we describe (i) the various modeling approaches used in NLP applications, (ii) learning approaches in the context of the NLP area,

and (iii) NLP tasks presented in the context of this thesis. Specifically, we provide a brief overview with references to previous literature rather than lengthy or detailed descriptions. This way, we aim to focus on techniques and tasks in order to allow the reader to understand terms described in subsequent chapters. This chapter contains six sections:

- In Section 1.1, we outline traditional approaches where hand-crafted features or external NLP tools are used.
- In Section 1.2, we introduce neural network methods and we describe various architectures and components frequently used for NLP.
- In Section 1.3, we discuss several learning methods that are commonly applied in the NLP area.
- In Section 1.4, we define a set of NLP tasks and we describe in more detail the tasks studied in this thesis.
- In Section 1.5, we summarize the contributions of this thesis.
- Finally, in Section 1.6, we list the publications presented in this thesis.

1.1 Traditional approaches in NLP

For a long time, feature-based methods or hand-crafted features were used in the NLP community. Specifically, in tasks such as text categorization [12], keyword extraction [13, 14], etc., features that look into the counts of words (see bag-of-words) or a relative importance score of the words inside the documents (see tf-idf) have been widely used. Alternative representations such as graph-based textual structures have also been introduced for information retrieval [15], keyword extraction [16] and text categorization [17, 18]. For the sequence labeling task, where the goal is to assign categorical labels to each word in a sentence, a number of different methods have been proposed (e.g., Hidden Markov Models (HMM) [19] and Conditional Random Fields (CRF) [20]). However, all of these methods heavily rely on hand-crafted features. Such hand-crafted features may include the distance between two examined terms in a document, the concatenation of the examined term with the (e.g., two, three) nearest tokens and so on. As another example, for the task of dependency parsing, where the goal is to analyze the grammatical structure of the sentence, again feature-based methods have been proposed. The main limitations of the aforementioned hand-engineered features are:

- The feature extraction procedure is time consuming and also depends on the application scenario (e.g., biology [21], social media context [22]), language (e.g., Dutch) [23] and the prior knowledge about each application (e.g., take specific characteristics for each language into account),
- Using high-dimensional features for representing textual documents leads to sparse representations and thus problems such as the curse of dimensionality might arise (i.e., we have too many features and too few examples),
- Using high-dimensional representations also leads to increased algorithm complexity.

1.2 Neural network approaches in NLP

Recently, deep learning with neural networks has been successfully applied to various NLP tasks ranging from NER [24, 25] and text categorization [26–28] to machine translation [29] and question answering [10]. Unlike feature-based methods that use sparse high-dimensional features, neural network architectures strongly rely on dense vector representations such as word embeddings (see word2vec [30], Glove [31]) to represent textual information. Depending on the task at hand (e.g., NER) various neural network approaches have been proposed. Specifically, Long Short-Term Memory (LSTM) networks [32], Convolutional Neural Networks (CNN) [33] and Transformers [34] have been exploited to extract useful high-level dense feature representations on top of the word embeddings layer. Results indicate that neural methods outperform hand-crafted representations in various settings and scenarios (e.g., machine translation [29], question answering [10]).

1.2.1 Embedding layer

As described in Section 1.1, natural language processing systems traditionally handle words as symbols, thus a word, such as the word “car”, is treated as a specific word id, e.g., “id 10”. This representation of words is not informative since relationships between similar terms are not taken into account. Specifically, the traditional bag-of-words and tf-idf representations fail to capture semantics between similar terms such as “car” and “bike”. For instance, both of them are vehicles, have wheels, etc. Moreover, such sparse representations lead to the curse of dimensionality as described above (see Section 1.1). To alleviate this, models that produce word

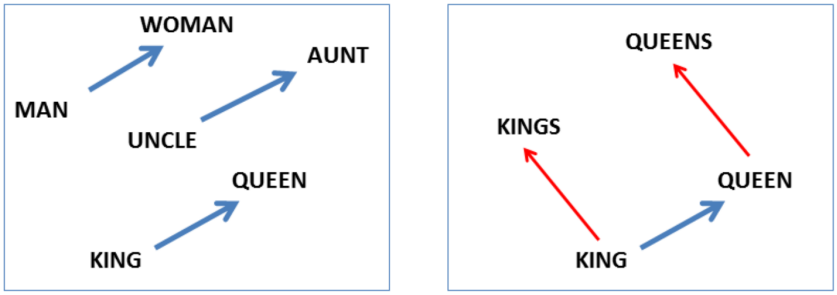

Figure 1.1: Left: Gender relations between 3 pairs of words in the vector space. Right: Singular/plural relations between 2 words. (Figure source [38])

representations (i.e., word embeddings), where semantically similar terms (i.e., words) are close to each other in a continuous vector space, have been proposed. An example of this type of vector representation for words (by projecting them to two dimensions) and the relationship between them is illustrated in Fig. 1.1. Word embeddings are commonly used as the first layer for many neural network models that achieve state-of-the-art performance in several NLP tasks (e.g., NER [24, 25, 35], dependency parsing [36, 37]).

1.2.1.1 Word embeddings

Mikolov et al. [30] proposed the word2vec toolkit, which is a learning model for word embeddings from raw text. Specifically, two types of models have been proposed: (i) the Continuous Bag-of-Words model (CBOW) and (ii) the Skip-Gram model. The CBOW model estimates the conditional probability of a particular word given the surrounding words within a specified window size. On the contrary, the Skip-Gram model predicts the surrounding words given the target word. Another well-known model for word embeddings is Glove, as proposed by Pennington et al. [31]. This is a "count-based" model since it is based on the computation of a co-occurrence matrix. In Chapter 3 where we train our own embeddings, we exploit the Skip-Gram model due to its capability to work well even on small collections and for rare words. In every other experiment presented in this thesis, we use pre-trained word embeddings as defined in previous works.

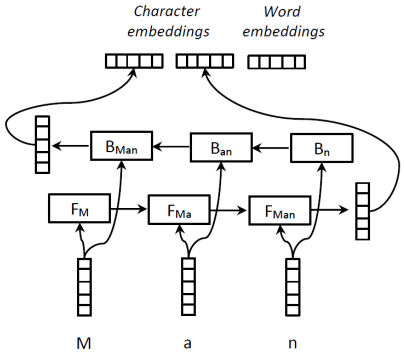

Figure 1.2: Embedding layer in detail. The characters of the word "Man" are represented by character vectors (i.e., embeddings) that are learned during training. The character embeddings are fed to a bidirectional LSTM (see Section 1.2.2) and the two final states (forward and backward) are concatenated. The vectors "*Character embeddings*" is the character-level representation of the word. This vector is then further concatenated to the word-level representation "*Word embeddings*" to obtain the complete word embedding vector.

1.2.1.2 Deep contextualized word representations

Words might have different meanings in various contexts. Thus, recently deep neural network architectures have been proposed to obtain word representations based on the context of the neighboring terms. This idea has been exploited either using LSTMs (see Section 1.2.2) or Transformers [34] in the ELMo [39], OpenAI-GPT [40, 41] and BERT [42] models. These word representations rely on deep neural network architectures that are pre-trained on the task of language modeling (i.e., increasing the model complexity/run-time compared to word embedding vectors). This way, they are able to obtain different word representations for each word and thus to model polysemous terms (e.g., the word "cell" can have several meanings based on the neighboring context) and improve the state-of-the-art on several NLP tasks.

1.2.1.3 Character embeddings

In addition to word embeddings, character embeddings are also commonly applied to NLP tasks (see neural NER [24, 25]). This type of embeddings is able to capture morphological features such as prefixes and suffixes. For instance, in the biological datasets for NER, the suffix "toxicity" can specify an *adverse drug event* entity such as "neurotoxicity" or "hepatotoxicity"

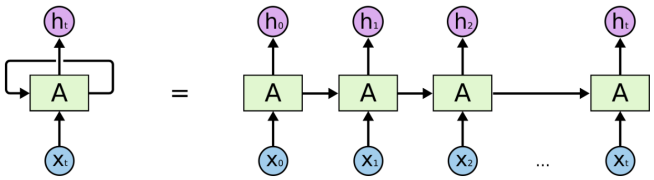

Figure 1.3: A typical RNN with a loop and the unrolled version. (Figure source [43])

and thus it is very informative. Another example might be the Dutch suffix "kamer" ("room" in English) in a real estate dataset which is used to specify entities of the type *space*, e.g., "badkamer" ("bathroom" in English) and "slaapkamer" ("bedroom" in English). Character-level embeddings are learned during the training phase [24, 25]. An example of a character embeddings layer is illustrated in Fig. 1.2. In this thesis, we experimentally show that character embeddings at the input layer are indeed beneficial for our neural architectures due to the aforementioned reasons.

1.2.2 RNN

Textual input is sequential and thus it is necessary to have neural network models that are able to capture sequential information. To this end, Recurrent Neural Networks (RNN) [44] have been proposed to model sequential data and allow information to flow from previous cells of the network (assuming that each word in the sequence is the input for each cell). Each cell of the RNN layer receives a token as input and produces an output. An example of an RNN layer is illustrated in Fig. 1.3. Although RNNs are good in capturing information present in previous timesteps, when the informative context is further back in the past, RNNs fail to well-capture this information (i.e., they remember only recent information). Since plain RNNs are not successful in capturing long-term dependencies [45, 46], a more advanced kind of RNNs has been proposed to overcome this issue: Long Short-term Memory (LSTM) cells have been successfully applied in several tasks to capture long-term dependencies. In this thesis, we mostly used LSTMs since we experiment with long documents.

1.2.3 CNN

As discussed in the introductory paragraph of Section 1.2, on top of the embeddings, we are interested to obtain high-level feature representations

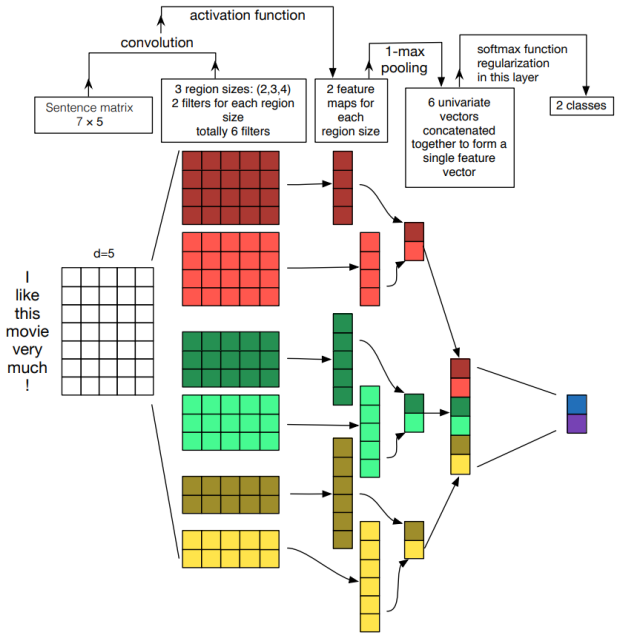

Figure 1.4: CNN layer for binary text classification. The filters, the convolutions, the result of the max-pooling and the softmax are depicted. (Figure source [47])

in various NLP tasks (see NER [35], text classification [26], summarization [9]). CNNs have been extensively used for this purpose in computer vision [48] and thus NLP researchers adopted CNNs to extract n-gram features. CNNs in NLP tasks are typically applied over the word embeddings layer. Specifically, a number of convolutional filters of different sizes are applied over the input layer. This phase is usually followed by a max-pooling operator. An example of a CNN layer for binary text classification is illustrated in Fig. 1.4. CNNs are not able to capture sequential information as opposed to LSTMs. The CNN filters are of fixed size and thus CNNs are able to capture only local information making them more suitable for applications such as text categorization (in which n-gram features are really informative).

1.3 Learning in NLP Tasks

Recent approaches for NLP tasks include neural network methods (as discussed in Section 1.2) that automatically extract useful information and

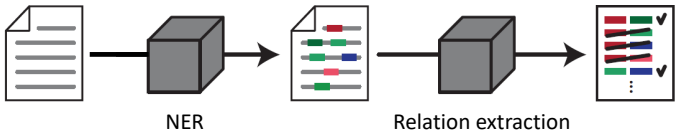

Figure 1.5: The task of NER and relation extraction in a pipeline fashion. The output of the first module (i.e., the predicted entities) are the input for the second module (i.e., the relation extractor).

achieve state-of-the-art performance. In this section, we present the most well-known methods to train models in different NLP tasks.

1.3.1 Single task learning

Typically, in supervised NLP settings, we are interested to optimize our model for a specific task. Specifically, the procedure is as follows: (i) collect data for the task at hand (e.g., NER), (ii) split the data into a train, development and test subsets, (iii) train the model, (iv) tune the parameters over a particular metric (e.g., F_1) on the development set, and (v) report the performance on the test set. However, there are cases where we study two or more tasks simultaneously (e.g., NER and relation extraction (RE), i.e., identify the *type* of the relation between two given entities) [49–53], part-of-speech (POS) tagging (i.e., identify the part-of-speech for each token in the sentence) and dependency parsing [54]). Traditionally, these tasks were studied using pipeline models where the output of the first module is propagated to the second module, etc. For instance, an example for the NER and relation extraction case is illustrated in Fig. 1.5.

In addition, it is known [49, 55] that the main drawbacks of studying tasks in a pipeline fashion are the following: (i) error propagation between the components (e.g., NER and RE) and (ii) possible useful information from one task is not exploited by the other (e.g., identifying a *Works for* relation might be helpful for the NER module in detecting the *type* of the two entities, i.e., Person (*PER*), Organization (*ORG*) and vice versa).

1.3.2 Multi-task learning

As mentioned in the previous section, there are cases where we study two or more tasks simultaneously (e.g., NER - relation extraction [49–53], POS tagging - chunking - dependency parsing - semantic relatedness - textual entailment [56], syntactic chunking - supertagging - POS tagging [57]) and we can benefit from the information learned during training for the studied tasks [58]. Thus, several complementary aspects of the input can be

encoded by applying the learning process of the various tasks simultaneously. Several works have been proposed in this direction. For instance, in Fig. 1.6, the model of [56] is illustrated, which is trained end-to-end for the tasks of POS tagging, chunking, dependency parsing, semantic relatedness, and textual entailment. This way their model is able to capture linguistic hierarchies starting from simpler tasks (i.e., POS tagging) to more complex ones (i.e., textual entailment). In another setting, Bingel and Søgaard [59] exploited the idea of multi-task learning to identify potential task relatedness among ten sequence labeling tasks such as chunking, keyword detection, POS tagging, etc. In additional work, Miwa and Bansal [49] introduced the usage of shared parameters for the task of end-to-end relation extraction (i.e., NER and relation extraction). In this thesis, we also exploit the idea of multi-task learning to share information between the studied tasks (see Chapters 3-4B).

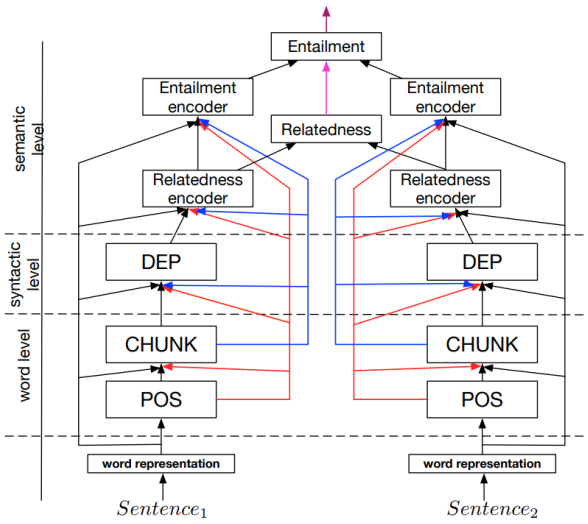

Figure 1.6: The joint many-task model proposed in [56] for POS tagging, chunking, dependency parsing, semantic relatedness, and textual entailment. (Figure source [56])

1.4 NLP tasks

The aim in NLP is to automatically obtain high-level language comprehension. This can be achieved by resolving several core NLP problems such as sequence labeling (see NER [35], POS tagging [60], etc.), dependency

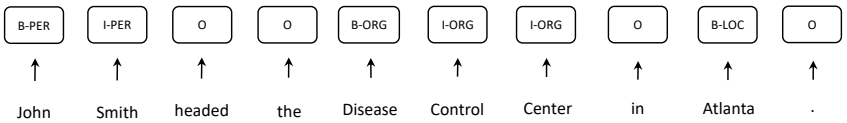

Figure 1.7: Example of a sentence using the BIO encoding scheme for the task of NER. For instance, the B-ORG and I-ORG tags indicate the beginning and the inside tokens of the entity "Disease Control Center", respectively.

parsing [37], question answering [10], text summarization [9], machine translation [11], etc. A wide variety of neural models have been recently proposed to solve all of the aforementioned tasks. In the following subsections, we present the core NLP problems that we have focused on in this thesis. Specifically, for each of the tasks, we describe its goal and we present the related work.

1.4.1 Sequence labeling

In this thesis, we focus on sequence labeling tasks such as the NER task (see Chapters 2-4B) and the sub-event detection problem (see Chapter 5). We formulate both tasks as a sequence labeling problem, similar to previous works [24, 25]. For the sequence labeling problem, usually the BIO (Beginning, Inside, Outside) encoding scheme is employed. For instance, in the NER task (i.e., identify the core entities of a sentence), each entity consists of multiple sequential tokens within the sentence and one should assign a tag for every token in the sentence. That way, the entity arguments (start and end position) and its *type* (e.g., *ORG*) are identified. To this end, (i) the B-*type* (beginning) to the first token of the entity, (ii) the I-*type* (inside) to every other token within the entity, and (iii) the O tag (outside) if a token is not part of an entity are assigned. Fig. 1.7 shows an example of the BIO encoding tags assigned to the tokens of the sentence. This way, both the boundaries (i.e., start and end positions of the entity) as well as their corresponding *type* are defined.

For sequence labeling, a number of different methods have been proposed, namely Hidden Markov Models (HMM) [19], Conditional Random Fields (CRF) [20], Maximum Margin Markov Network (M^3N) [61], generalized support vector machines for structured output (SVM^{struct}) [62] and Search-based Structured Prediction (SEARN) [63]. Those methods heavily rely on hand-crafted features and an in-depth review can be found in [64]. Several variations of these models that also require manual feature engineering have been used in different application settings (e.g., biology, so-

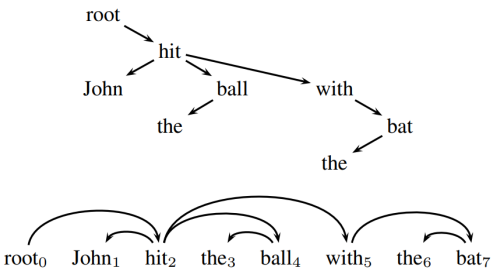

Figure 1.8: An example sentence with its corresponding dependency parse tree. (Figure source [72])

cial media context) and languages (e.g., Turkish) [65–68]. Recently, deep learning with neural networks has been succesfully applied to sequence labeling. Collobert et al. [69] proposed to use a convolutional neural network (CNN) followed by a CRF layer over a sequence of word embeddings. In this direction, RNNs have also been widely used. Gillick et al. [70] use a sequence-to-sequence approach for modeling the sequence labeling task. In addition, several variants of combinations between LSTM and CRF models have been proposed [24, 25, 71] achieving state-of-the-art performance on publicly available datasets.

1.4.2 Dependency parsing

Dependency parsing is a well studied task in the NLP community, which aims to analyze the grammatical structure of a sentence as illustrated in the example of Fig. 1.8. Specifically, the goal of the dependency parsing problem is to link each token to its syntactical parent token, so as to create the dependency parse (set of dependencies that form a *tree* structure) of the sentence. Assuming the sentence $S = \{w_0 w_1 ... w_t\}$ where t is the number of tokens within the sentence, a dependency is a pair (p, c) where $p \in S$ is the parent token and $c \in S$ is the child token. The entity w_0 is the dummy root-symbol that only appears as parent. There are two well-established ways to address the dependency parsing problem, via (i) graph-based and (ii) transition-based parsers.

Graph-based: Dependency parsing can be framed as a graph-based structured problem over a directed graph $G = (V, E)$, where the words and the dependencies among them constitute the set of nodes (V) and the edges (E) of the graph, respectively. Then the problem boils down to the search of the maximum spanning tree (MST) in the directed graph. This graph structure is also depicted in the upper part of Fig. 1.8. In the work of [72, 73]

dependency parsing requires the search of the highest scoring maximum spanning tree in graphs for both projective (dependencies are not allowed to cross) and non-projective (crossing dependencies are allowed) trees with the Eisner algorithm [74] and the Chu-Liu-Edmonds algorithm [75, 76] respectively. It was shown that exploiting higher-order information (e.g., siblings, grand-parental relation) in the graph, instead of just using first-order information (i.e., parent relations) [77, 78] may yield significant improvements of the parsing accuracy but comes at the cost of an increased model complexity. Koo et al. [79] made an important step towards globally normalized models with hand-crafted features, by adapting the Matrix-Tree Theorem (MTT) [80] to train over all non-projective dependency trees. Recent advances in neural graph-based parsing [37, 81, 82] include the use of LSTMs to capture richer contextual information compared to hand-crafted feature-based methods. Our work (in Chapters 3-4B) is conceptually related to Zhang et al. [37], who formulated the dependency parsing problem as a head selection problem.

Transition-based: Transition-based parsers [83, 84] replace the exact inference of the graph-based parsers by an approximate but faster inference method. The dependency parsing problem is now solved by an abstract state machine that gradually builds up the dependency tree token by token. The goal of this kind of parsers is to find the most probable transition sequence from an initial configuration to some terminal configuration (i.e., a dependency parse tree) given a permissible set of actions (i.e., LEFT-ARC, RIGHT-ARC, SHIFT) [85, 86]. In the simplest case (i.e., greedy inference), a classifier predicts the next transition based on the current configuration. Compared to graph-based dependency parsers, transition-based parsers are able to scale better due to the linear time complexity while graph-based complexity rises to $O(n^2)$. Chen and Manning [87] proposed a way of learning a neural network classifier for use in a greedy, transition-based dependency parser while using low-dimensional, dense word embeddings, without the need of manually extracting features. Globally normalized transition-based parsers [88] can be considered an extension of [87], as they perform beam search for maintaining multiple hypotheses and introduce global normalization with a CRF objective. Dyer et al. [36] introduced the stack-LSTM model with push and pop operations which is able to learn the parser transition states while maintaining a summary embedding of its contents. Transition-based systems are well-known for their speed and state-of-the-art performance and thus we include them in our study as baselines in Chapter 2.

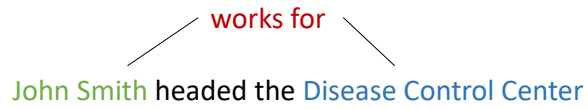

Figure 1.9: An example sentence for relation extraction where the two entities are colored in green and blue, respectively and the *type* of the relation between them is colored in red.

1.4.3 Relation extraction

The relation extraction (RE) task is defined as the identification of the relationships between pairs of entities and an example is illustrated in Fig. 1.9. For instance, assuming that we have entities e_1 and e_2 (known in advance) in sentence S, the aim of the task is to classify the *type* of the relation between the two entities. The main approaches for relation extraction rely either on hand-crafted features [89, 90] or neural networks [91, 92]. Feature-based methods focus on obtaining effective hand-crafted features, for instance defining kernel functions [89, 93] and designing lexical, syntactic, semantic features, etc. [90, 94]. Neural network models have been proposed to overcome the issue of manually designing hand-crafted features leading to improved performance. CNN-based [92, 95, 96] and RNN-based [97–99] models have been introduced to automatically extract lexical and sentence level features leading to a deeper language understanding. Vu et al. [100] combine CNNs and RNNs using an ensemble scheme to achieve state-of-the-art results.

1.5 Research contributions

In this section, we describe the main contribution of this thesis. Each chapter tackles a core NLP problem with new methods. Table 1.1 gives an overview of the various contributions presented in this thesis. In Chapters 2-4B, we propose novel methods for the task of end-to-end relation extraction (i.e., entity recognition and relation extraction) and in Chapter 5, we frame the problem of sub-event detection in Twitter streams as a sequence labeling problem [101]. The contribution of each chapter is summarized as follows:

- In Chapter 2, we (i) define a new real estate extraction problem where the aim is to recover a tree-like structured representation of the property (the *property tree*) based on its natural language description, (ii) introduce structured learning methods that solve the newly defined

Table 1.1: Overview of contributions presented in this thesis.

Chapter	Task	Contribution
2	Real-estate structure prediction problem	Define the problem and propose a two-step pipeline method
3	Real-estate structure prediction problem	A new neural joint model that outperform the two-step pipeline methods
4A	Multi-context entity recognition and relation extraction	A new neural joint model that outperform several state-of-the-art methods on the task
4B	Multi-context entity recognition and relation extraction	Add adversarial perturbations on top of our neural joint model presented in Chapter 4A
5	Sub-event detection in Twitter streams	Frame the task as a sequence labeling problem to take into account chronological information between consecutive tweets

problem, and (iii) experimentally evaluate our models on the newly created and annotated real-world data set.

- In Chapter 3, (i) we propose advanced neural models that consider the two subtasks presented in Chapter 2 jointly (i.e., a new joint model that encodes the two tasks of identifying entities as well as dependencies between them, as a single problem, without the need of parameter sharing or pre-training of the first entity recognition module separately), (ii) we compare the proposed joint model against established pipeline approaches and we report a consistent performance improvement, and (iii) we perform extensive analysis of several attention mechanisms that enable our LSTM-based model to focus on informative words and phrases, reporting an improved performance compared to previous models.

- In Chapter 4A, we focus on a new general purpose joint model that performs the two tasks of entity recognition and relation extraction simultaneously while reducing the complexity of the model described in Chapter 3. Our model achieves state-of-the-art performance in a number of different contexts (i.e., news, biomedical, real estate) and languages (i.e., English, Dutch) (including the dataset described in Chapter 2) without relying on any manually engineered features nor additional NLP tools.

- Chapter 4B, our contribution of the proposed method is twofold:

(i) we investigate the consistent effectiveness of adversarial perturbations as a regularization method over the multi-context baseline joint model presented in Chapter 4A, with (ii) a large scale experimental evaluation. Experimental results indicate that adversarial perturbations improve the results for each task separately (i.e., entity recognition and relation extraction), as well as the overall performance of our joint model (presented in Chapter 4A), while reaching high performance already during the first epochs of the training procedure.

- In Chapter 5, we address the problem of sub-event detection in Twitter streams. We frame the problem as a sequence labeling task to exploit the chronological relation between consecutive tweets. Our work does take into account the chronological order and we predict the presence and the type of a sub-event exploiting information from previous tweets. Specifically, we (i) propose a new neural baseline model that outperforms the state-of-the-art performance on the binary classification problem of detecting the presence/absence of sub-events in a sports stream, (ii) establish a new baseline for predicting also the sub-event *types*, (iii) explicitly take into account chronological information, i.e., the relation among consecutive tweets, by framing sub-event detection as a sequence labeling problem on top of our baseline model, and (iv) perform an experimental study, indicating the benefit of sequence labeling for sub-event detection in sports Twitter streams.

- In Chapter 6, we (i) summarize the core findings/modeling approaches, and (ii) outline future research directions opened by the work described in this thesis.

1.6 Publications

The research results obtained during this PhD research have been published in scientific journals and presented at a series of international conferences and workshops. The following list provides an overview of these publications.

1.6.1 Publications in international journals (listed in the Science Citation Index[1])

I **G. Bekoulis**, J. Deleu, T. Demeester, and C. Develder, *Joint Entity Recognition and Relation Extraction as a Multi-head Selection Problem*. Published in Expert Systems with Applications. 114: 34-45, 2018. **acceptance rate:** 12%

II **G. Bekoulis**, J. Deleu, T. Demeester, and C. Develder, *An attentive neural architecture for joint segmentation and parsing and its application to real estate ads*. Published in Expert Systems with Applications. 102, 100-112, 2018. **acceptance rate:** 12%

1.6.2 Publications in international conferences

III **G. Bekoulis**, J. Deleu, T. Demeester, and C. Develder, *Sub-event detection from twitter streams as a sequence labeling problem*. 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019. **acceptance rate:** 23%

IV **G. Bekoulis**, J. Deleu, T. Demeester, and C. Develder, *Adversarial training for multi-context joint entity and relation extraction*. 2018 Conference on Empirical Methods in Natural Language Processing. pp. 2830-2836, 2018. **acceptance rate:** 23.2%, 9.5% for oral

V **G. Bekoulis**, J. Deleu, T. Demeester, and C. Develder, *Reconstructing the house from the ad: Structured prediction on real estate classifieds*. 15th Conference of the European Chapter of the Association for Computational Linguistics. pp. 274-279, 2017. **acceptance rate:** 24%

[1]The publications listed are recognized as 'A1 publications', according to the following definition used by Ghent University: *"A1 publications are articles listed in the Science Citation Index, the Social Science Citation Index or the Arts and Humanities Citation Index of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper."*

1.6.3 Publications in international conferences (not included in this thesis)

VII S. Bitew, **G. Bekoulis**, J. Deleu, L. Sterckx, K. Zaporojets, T. Demeester, and C. Develder, *Predicting Suicide Risk from Online Postings in Reddit – The UGent-IDLab submission to the CLPysch 2019 Shared Task A*. 6th Ann. Workshop on Computational Linguistics and Clinical Psychology (CLPsych 2019) at NAACL-HLT, 2019.

VIII **G. Bekoulis**, F. Rousseau, *Graph-based Term Weighting Scheme for Topic Modeling*. 2016 IEEE 16th International Conference on Data Mining: Workshops (ICDMW). pp. 1039-1044, 2016.

References

[1] C. Perlich, B. Dalessandro, T. Raeder, O. Stitelman, and F. Provost. *Machine learning for targeted display advertising: transfer learning in action*. Machine Learning, 95(1):103–127, Apr 2014. Available from: https://doi.org/10.1007/s10994-013-5375-2, doi:10.1007/s10994-013-5375-2.

[2] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang. *Jointly Modeling Aspects, Ratings and Sentiments for Movie Recommendation (JMARS)*. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, pages 193–202, New York, NY, USA, 2014. ACM. Available from: http://doi.acm.org/10.1145/2623330.2623758, doi:10.1145/2623330.2623758.

[3] H. Wang, Z. Lu, H. Li, and E. Chen. *A dataset for research on short-text conversations*. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 935–945, 2013.

[4] A. Graves, A. r. Mohamed, and G. Hinton. *Speech recognition with deep recurrent neural networks*. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing, pages 6645–6649, Vancouver, Canada, 26–31 May. 2013. doi:10.1109/ICASSP.2013.6638947.

[5] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. *End to End Learning for Self-Driving Cars*. 2016.

[6] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. *Convolutional LSTM network: A machine learning approach for precipitation nowcasting*. In Advances in neural information processing systems, pages 802–810, 2015.

[7] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[8] T. Young, D. Hazarika, S. Poria, and E. Cambria. *Recent trends in deep learning based natural language processing*. ieee Computational intelligenCe magazine, 13(3):55–75, 2018.

[9] S. Narayan, S. B. Cohen, and M. Lapata. *DonâĂŹt Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization*. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1797–1807, 2018.

[10] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. *Bidirectional attention flow for machine comprehension*. In Proceedings of the International Conference for Learning Representations, 2017.

[11] D. Bahdanau, K. Cho, and Y. Bengio. *Neural machine translation by jointly learning to align and translate*. In Proceedings of the International Conference for Learning Representations, San Diego, USA, 7–9 May 2015.

[12] T. Joachims. *Text categorization with Support Vector Machines: learning with many relevant features*. In Proceedings of the 10th European Conference on Machine Learning, pages 137–142. Springer-Verlag, 1998.

[13] K. S. Hasan and V. Ng. *Automatic keyphrase extraction: A survey of the state of the art*. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 1262–1273, 2014.

[14] E. Papagiannopoulou and G. Tsoumakas. *A Review of Keyphrase Extraction*. arXiv preprint arXiv:1905.05044, 2019.

[15] R. Blanco and C. Lioma. *Graph-based term weighting for information retrieval*. Information retrieval, 15(1):54–92, 2012.

[16] R. Mihalcea and P. Tarau. *Textrank: Bringing order into text*. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 2004.

[17] F. D. Malliaros and K. Skianis. *Graph-based term weighting for text categorization*. In Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015, pages 1473–1479. ACM, 2015.

[18] G. Bekoulis and F. Rousseau. *Graph-based term weighting scheme for topic modeling*. In 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), pages 1039–1044. IEEE, 2016.

[19] L. Rabiner and B. Juang. *An introduction to hidden Markov models*. IEEE ASSP Magazine, 3(1):4–16, 1986. doi:10.1109/MASSP.1986.1165342.

[20] J. Lafferty, A. McCallum, and F. Pereira. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. In Proceedings of the 18th International Conference on Machine Learning, pages 282–289, San Francisco, USA, 28 Jun.–1 Jul. 2001. Morgan Kaufmann.

[21] H. Gurulingappa, A. Mateen-Rajpu, and L. Toldo. *Extraction of potential adverse drug events from medical case reports*. Journal of Biomedical Semantics, 3(1):1–15, 2012. doi:10.1186/2041-1480-3-15.

[22] J. Nichols, J. Mahmud, and C. Drews. *Summarizing Sporting Events Using Twitter*. In Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces, pages 189–198, New York, NY, USA, 2012. ACM. Available from: http://doi.acm.org/10.1145/2166966.2166999, doi:10.1145/2166966.2166999.

[23] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Reconstructing the house from the ad: Structured prediction on real estate classifieds*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 2, Short Papers), pages 274–279, Valencia, Spain, 3–7 Apr. 2017.

[24] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. *Neural Architectures for Named Entity Recognition*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270, San Diego, California, 12–17 Jun. 2016.

[25] X. Ma and E. Hovy. *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1064–1074, Berlin, Germany, 7–12 Aug. 2016.

[26] Y. Kim. *Convolutional Neural Networks for Sentence Classification*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751. Association for Computational Linguistics, 2014. Available from: http://aclweb.org/anthology/D14-1181, doi:10.3115/v1/D14-1181.

[27] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III. *Deep Unordered Composition Rivals Syntactic Methods for Text Classification*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1681–1691. Association for Computational Linguistics, 2015. Available from: http://aclweb.org/anthology/P15-1162, doi:10.3115/v1/P15-1162.

[28] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. *Hierarchical Attention Networks for Document Classification*. In Proceedings of the 2016 Conference of the North American Chapter of the

Association for Computational Linguistics: Human Language Technologies, pages 1480–1489. Association for Computational Linguistics, 2016. Available from: http://aclweb.org/anthology/N16-1174, doi:10.18653/v1/N16-1174.

[29] I. Sutskever, O. Vinyals, and Q. V. Le. *Sequence to Sequence Learning with Neural Networks*. In Proceedings of the 27th International Conference on Neural Information Processing Systems, pages 3104–3112, Montreal, Canada, 08–13 Dec. 2014. MIT Press.

[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. *Distributed Representations of Words and Phrases and their Compositionality*. In Proceedings of the 26th International Conference on Neural Information Processing Systems, pages 3111–3119, Nevada, United States, 5–10 Dec. 2013. Curran Associates, Inc.

[31] J. Pennington, R. Socher, and C. Manning. *Glove: Global vectors for word representation*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pages 1532–1543, 2014.

[32] S. Hochreiter and J. Schmidhuber. *Long Short-Term Memory*. Neural computation, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11):2278–2324, Nov 1998. doi:10.1109/5.726791.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. *Attention is all you need*. In Advances in neural information processing systems, pages 5998–6008, 2017.

[35] J. Chiu and E. Nichols. *Named Entity Recognition with Bidirectional LSTM-CNNs*. Transactions of the Association for Computational Linguistics, 4:357–370, 2016.

[36] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. *Transition-Based Dependency Parsing with Stack Long Short-Term Memory*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 334–343, Beijing, China, 26–31 Jul. 2015.

[37] X. Zhang, J. Cheng, and M. Lapata. *Dependency Parsing as Head Selection*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 1, Long Papers), pages 665–676, Valencia, Spain, 3–7 Apr. 2017.

[38] T. Mikolov, W.-t. Yih, and G. Zweig. *Linguistic regularities in continuous space word representations*. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 746–751, 2013.

[39] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. *Deep Contextualized Word Representations*. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, 2018.

[40] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. *Improving language understanding by generative pre-training*. 2018.

[41] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. *Language models are unsupervised multitask learners*. 2019.

[42] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. *Bert: Pre-training of deep bidirectional transformers for language understanding*. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2019.

[43] *Understanding LSTM Networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2019-04-30.

[44] J. L. Elman. *Finding structure in time*. Cognitive science, 14(2):179–211, 1990.

[45] Y. Bengio, P. Simard, and P. Frasconi. *Learning Long-term Dependencies with Gradient Descent is Difficult*. Transactions on neural networks, 5(2):157–166, 1994. doi:10.1109/72.279181.

[46] R. Pascanu, T. Mikolov, and Y. Bengio. *On the Difficulty of Training Recurrent Neural Networks*. In Proceedings of the 30th International Conference on International Conference on Machine Learning, pages 1310–1318, Atlanta, USA, 16–21 Jun. 2013. JMLR.org.

[47] Y. Zhang and B. Wallace. *A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification*. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 253–263, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. Available from: https://www.aclweb.org/anthology/I17-1026.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton. *Imagenet classification with deep convolutional neural networks*. In Advances in neural information processing systems, pages 1097–1105, 2012.

[49] M. Miwa and M. Bansal. *End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1105–1116, Berlin, Germany, 7–12 Aug. 2016.

[50] A. Katiyar and C. Cardie. *Going out on a limb: Joint Extraction of Entity Mentions and Relations without Dependency Trees*. In Proceedings of the 55st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, Canada, 2017.

[51] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *An attentive neural architecture for joint segmentation and parsing and its application to real estate ads*. Expert Systems with Applications, 102:100 – 112, 2018. doi:10.1016/j.eswa.2018.02.031.

[52] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Joint entity recognition and relation extraction as a multi-head selection problem*. Expert Systems with Applications, 114:34–45, 2018.

[53] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Adversarial training for multi-context joint entity and relation extraction*. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2830–2836, 2018.

[54] D. Q. Nguyen and K. Verspoor. *An Improved Neural Network Model for Joint POS Tagging and Dependency Parsing*. In Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, pages 81–91, 2018.

[55] M. Miwa and Y. Sasaki. *Modeling Joint Entity and Relation Extraction with Table Representation*. In Proceedings of the 2014 Conference

on Empirical Methods in Natural Language Processing, pages 1858–1869, Doha, Qatar, 25–29 Oct. 2014. Association for Computational Linguistics.

[56] K. Hashimoto, Y. Tsuruoka, R. Socher, et al. *A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks*. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1923–1933, 2017.

[57] A. Søgaard and Y. Goldberg. *Deep multi-task learning with low level tasks supervised at lower layers*. In The 54th Annual Meeting of the Association for Computational Linguistics, page 231, 2016.

[58] R. Caruana. *Multitask Learning: A Knowledge-Based Source of Inductive Bias*. In Proceedings of the Tenth International Conference on Machine Learning, pages 41–48. Morgan Kaufmann, 1993.

[59] J. Bingel and A. Søgaard. *Identifying beneficial task relations for multi-task learning in deep neural networks*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 164–169, 2017.

[60] W. Ling, C. Dyer, A. W. Black, I. Trancoso, R. Fermandez, S. Amir, L. Marujo, and T. Luis. *Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation*. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1520–1530, Lisbon, Portugal, 2015. Association for Computational Linguistics. doi:10.18653/v1/D15-1176.

[61] B. Taskar, C. Guestrin, and D. Koller. *Max-Margin Markov networks*. In Proceedings of the 16th International Conference on Neural Information Processing Systems, pages 25–32. MIT Press, Bangkok, Thailand, 1–5 Dec. 2003.

[62] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. *Support Vector Machine Learning for Interdependent and Structured Output Spaces*. In Proceedings of the 21st International Conference on Machine Learning, pages 104–112, Helsinki, Finland, 5–9 Jul. 2004. ACM. doi:10.1145/1015330.1015341.

[63] H. Daumé III, J. Langford, and D. Marcu. *Search-based structured prediction*. Machine Learning Journal, 75(3):297–325, 2009. doi:10.1007/s10994-009-5106-x.

[64] N. Nguyen and Y. Guo. *Comparisons of Sequence Labeling Algorithms and Extensions*. In Proceedings of the 24th International Conference on Machine Learning, pages 681–688, Corvallis, USA, 20–24 Jun. 2007. ACM. doi:10.1145/1273496.1273582.

[65] J. J. Jung. *Online named entity recognition method for microtexts in social networking services: A case study of twitter*. Expert Systems with Applications, 39(9):8066 – 8070, 2012. doi:10.1016/j.eswa.2012.01.136.

[66] D. Küçük and A. Yazıcı. *A hybrid named entity recognizer for Turkish*. Expert Systems with Applications, 39(3):2733 – 2742, 2012. doi:10.1016/j.eswa.2011.08.131.

[67] J. Atkinson and V. Bull. *A multi-strategy approach to biological named entity recognition*. Expert Systems with Applications, 39(17):12968 – 12974, 2012. doi:10.1016/j.eswa.2012.05.033.

[68] M. Konkol, T. Brychcín, and M. Konopík. *Latent semantics in Named Entity Recognition*. Expert Systems with Applications, 42(7):3470 – 3479, 2015. doi:10.1016/j.eswa.2014.12.015.

[69] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. *Natural Language Processing (Almost) from Scratch*. Journal of Machine Learning Research, 12:2493–2537, November 2011.

[70] D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya. *Multilingual Language Processing From Bytes*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1296–1306, San Diego, California, 12–17 Jun. 2016.

[71] Z. Huang, W. Xu, and K. Yu. *Bidirectional LSTM-CRF models for sequence tagging*. arXiv preprint arXiv:1508.01991, 2015.

[72] R. McDonald and F. Pereira. *Online Learning of Approximate Dependency Parsing Algorithms*. In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, pages 81–88, Trento, Italy, 5–6 Apr. 2007.

[73] R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. *Non-Projective Dependency Parsing using Spanning Tree Algorithms*. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pages 523–530, Vancouver, British Columbia, Canada, 06–08 Oct. 2005. Association for Computational Linguistics.

[74] J. M. Eisner. *Three New Probabilistic Models for Dependency Parsing: An Exploration*. In Proceedings of the 16th International Conference on Computational Linguistics (Volume 1), pages 340–345, Copenhagen, Denmark, 5–9 Aug. 1996.

[75] Y.-J. Chu and T.-H. Liu. *On shortest arborescence of a directed graph*. Scientia Sinica, 14:1396âĂŞ–1400, 1965.

[76] J. Edmonds. *Optimum branchings*. Journal of research of the National Bureau of Standards, 71B(4):233–240, 1967.

[77] X. Carreras. *Experiments with a Higher-Order Projective Dependency Parser*. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 957–961, Prague, Czech, 28–30 Jun. 2007. Association for Computational Linguistics.

[78] H. Zhang and R. McDonald. *Generalized Higher-Order Dependency Parsing with Cube Pruning*. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 320–331, Jeju Island, Korea, 12–14 Jul. 2012. Association for Computational Linguistics.

[79] T. Koo, A. Globerson, X. Carreras, and M. Collins. *Structured Prediction Models via the Matrix-Tree Theorem*. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 141–150, Prague, Czech, 28–30 Jun. 2007. Association for Computational Linguistics.

[80] W. T. Tutte. *Graph Theory*. In Encyclopedia of Mathematics and its Applications, volume 21, page 138. Cambridge University Press, 2001.

[81] E. Kiperwasser and Y. Goldberg. *Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations*. Transactions of the Association for Computational Linguistics, 4:313–327, 2016.

[82] W. Wang and B. Chang. *Graph-based Dependency Parsing with Bidirectional LSTM*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2306–2315, Berlin, Germany, 7–12 Aug. 2016.

[83] H. Yamada and Y. Matsumoto. *Statistical dependency analysis with support vector machines*. In Proceedings of the 8th International Workshop on Parsing Technologies, pages 195–206, Nancy, France, 23–25 Apr. 2003.

[84] J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. *Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines*. In Proceedings of the 10th Conference on Computational Natural Language Learning, pages 221–225, New York, USA, 8–9 Jun. 2006. Association for Computational Linguistics.

[85] J. Nivre. *An efficient algorithm for projective dependency parsing*. In Proceedings of the 8th International Workshop on Parsing Technologies, pages 149–160, Nancy, France, 23–25 Apr. 2003.

[86] J. Nivre. *Non-Projective Dependency Parsing in Expected Linear Time*. In Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, pages 351–359, Singapore, 2–7 Aug. 2009.

[87] D. Chen and C. Manning. *A Fast and Accurate Dependency Parser using Neural Networks*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pages 740–750, Doha, Qatar, 25–29 Oct. 2014. Association for Computational Linguistics.

[88] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins. *Globally Normalized Transition-Based Neural Networks*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2442–2452, Berlin, Germany, 7–12 Aug. 2016.

[89] D. Zelenko, C. Aone, and A. Richardella. *Kernel Methods for Relation Extraction*. Journal of Machine Learning Research, 3:1083–1106, 2003. doi:10.3115/1118693.1118703.

[90] N. Kambhatla. *Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations*. In Proceedings of the Annual Meeting of the Association for Computational Linguistics on Interactive poster and demonstration sessions, Barcelona, Spain, 2004. doi:10.3115/1219044.1219066.

[91] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. *Semantic compositionality through recursive matrix-vector spaces*. In Proceedings of the

2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211, Jeju Island, Korea, 12–14 Jul. 2012. Association for Computational Linguistics.

[92] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao. *Relation classification via convolutional deep neural network*. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 2335–2344, 2014.

[93] A. Culotta and J. Sorensen. *Dependency tree kernels for relation extraction*. In Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, pages 423–429, Barcelona, Spain, 2004. doi:10.3115/1218955.1219009.

[94] B. Rink and S. Harabagiu. *Utd: Classifying semantic relations by combining lexical and semantic resources*. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 256–259, Los Angeles, California, 2010. Association for Computational Linguistics.

[95] K. Xu, Y. Feng, S. Huang, and D. Zhao. *Semantic Relation Classification via Convolutional Neural Networks with Simple Negative Sampling*. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 536–540, Lisbon, Portugal, September 2015. Association for Computational Linguistics. Available from: http://aclweb.org/anthology/D15-1062.

[96] C. dos Santos, B. Xiang, and B. Zhou. *Classifying Relations by Ranking with Convolutional Neural Networks*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 626–634, Beijing, China, 26–31 Jul. 2015.

[97] R. Socher, D. Chen, C. D. Manning, and A. Ng. *Reasoning With Neural Tensor Networks for Knowledge Base Completion*. In Proceedings of the 26th International Conference on Neural Information Processing Systems, pages 926–934, Nevada, United States, 5–10 Dec. 2013. Curran Associates, Inc.

[98] D. Zhang and D. Wang. *Relation classification via recurrent neural network*. arXiv preprint arXiv:1508.01006, 2015.

[99] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin. *Classifying Relations via Long Short Term Memory Networks along Shortest Dependency*

Paths. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1785–1794, Lisbon, Portugal, 17–21 Sept. 2015. Association for Computational Linguistics.

[100] N. T. Vu, H. Adel, P. Gupta, and H. Schütze. *Combining Recurrent and Convolutional Neural Networks for Relation Classification*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 534–539, San Diego, California, June 2016. Available from: http://www.aclweb.org/anthology/N16-1065.

[101] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Sub-event detection from twitter streams as a sequence labeling problem*. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 745–750, Minneapolis, Minnesota, June 2019. doi:10.18653/v1/N19-1081.

2

Reconstructing the house from the ad: Structured prediction on real estate classifieds

In this chapter, we introduce a new structured prediction problem for real estate properties. In this task, we aim to recover a hierarchical structured representation of a property based only on its textual advertisement. In addition, we propose various methods to resolve the newly defined structured prediction problem. We split the problem into simpler subtasks that resolve the problem in a pipeline setting. Specifically, the pipeline of subtasks is the following: (i) named entity recognition for real estate entities, (ii) dependency parsing for predicting the part-of relationships between the identified pairs of real estate entities, and (iii) a maximum spanning tree algorithm for recovering the tree structured description of the property (i.e., the desired property tree).

⋆⋆⋆

G. Bekoulis, J. Deleu, T. Demeester and C. Develder

In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, 2017.

Abstract In this paper, we address the new (to the best of our knowledge) problem of extracting a structured description of real estate properties from their natural language descriptions in classifieds. We survey and present several models to (a) identify important entities of a property (e.g., rooms) from classifieds and (b) structure them into a tree format, with the entities as nodes and edges representing a part-of relation. Experiments show that a graph-based system deriving the tree from an initially fully connected entity graph outperforms a transition-based system starting from only the entity nodes, since it better reconstructs the tree.

2.1 Introduction

In the real estate domain, user-generated free text descriptions form a highly useful but unstructured representation of real estate properties. However, there is an increasing need for people to find useful (structured) information from large sets of such descriptions, and for companies to propose sales/rentals that best fit the clients' needs, while keeping human reading effort limited. For example, real estate descriptions in natural language may not be directly suited for specific search filters that potential buyers want to apply. On the other hand, a hierarchical data structure representing the real estate property enables specialized filtering (e.g., based on the number of bedrooms, number of floors, or the requirement of having a bathroom with a toilet on the first floor), and is expected to also benefit related applications such as automated price prediction [1, 2].

Our primary objective is to define the new real estate structure extraction problem, and explore its solution using combinations of state-of-the-art methods, thus establishing its difficulty by obtaining performance results for future reference. More specifically, we contribute with: (i) the definition of the real estate extraction problem, amounting to a tree-like structured representation of the property (the *property tree*) based on its natural language description; (ii) the introduction of structured learning methods that solve the newly defined problem; and (iii) experimental evaluation of the systems on a newly created and annotated real-world data set. For Part (ii), we break down the problem into simpler components, using (1) Conditional Random Fields (CRFs) for real estate entity recognition (where entities are floors, rooms, sub-spaces in rooms, etc.), (2) non-projective dependency parsing to predict the part-of relationships between such entities (comparing local and global graph-based, and transition-based algorithms), and (3) a maximum spanning tree algorithm for decoding the desired *property tree*.

2.2 Related work

The challenge in structured prediction largely stems from the size of the output space. Specifically in NLP, for sequence labeling (e.g., named entity recognition), which is the first building block of our system, a number of different methods have been proposed, namely CRFs [3], Maximum Margin Markov Network (M^3N) [4], SVM^{struct} [5] and SEARN [6].

We exploit dependency parsing methods for the construction of the *property tree* which is similar to the problem of learning the dependency arcs of a sentence. Dependency parsing research has focused on both graph-based and transition-based parsers. McDonald et al. [7, 8] have shown that treating dependency parsing as the search of the highest scoring maximum spanning tree in graphs yields efficient algorithms for both projective (dependencies are not allowed to cross) and non-projective (crossing dependencies are allowed) trees. Later, Koo et al. [9], adapted the Matrix-Tree Theorem [10] for globally normalized training over all non-projective dependency trees. On the other hand, transition-based dependency parsing aims to predict a transition sequence from an initial configuration to some terminal configuration and handles both projective and non-projective dependencies [11, 12]. Recent advances on those systems involve neural scoring functions [13] and globally normalized models [14].

More recently, a substantial amount of work (Kate and Mooney [15], Li and Ji [16], Miwa and Sasaki [17] and Li et al. [18]) jointly considered the two subtasks of entity recognition and dependency parsing. Our work is different since we aim to handle directed spanning trees, or equivalently non-projective dependency structures (i.e., the entities involved in a relation are not necessarily adjacent in the text since other entities may be mentioned in between), which complicates parsing.

2.3 Structured prediction of real estate properties

We now present the real estate extraction problem and our proposed proof-of-concept solutions.

2.3.1 Problem formulation

We define *entities* and *entity types* for our real estate extraction task. We define an **entity** as an unambiguous, unique part of a property with independent existence (e.g., bedroom, kitchen, attic). We define as *entity mention*, a textual phrase (e.g., "a small bedroom") that we can potentially link to one or more of the entities and whose semantic meaning unambiguously

Table 2.1: Real estate entity types.

Entity type	Description	Examples
property	The property.	bungalow, apartment
floor	A floor in a building.	ground floor
space	A room within the building.	bedroom, bathroom
subspace	A part of a room.	shower, toilet
field	An open space inside or outside the building.	bbq, garden
extra building	An additional building which is also part of the property.	garden house

represents a specific entity. Each entity can occur several times in the text, possibly with different mentions and we further classify entities into **types** as listed in Table 2.1.

The goal of our structured prediction task is to convert the given input text to a structured representation in the form of a so-called *property tree*, as illustrated in Fig. 2.1. That conversion implies both the detection of entities of various types (the "house" property entity, and the spaces "living room", "kitchen", etc.) as well as the part-of dependencies between them (e.g., that the "kitchen" is a part of the "house"). We cast the tree construction given the entities as a dependency parsing task over the search of the most probable *property tree*, since (i) this means decisions on all possible part-of relations are taken jointly (e.g., a certain room can only be part of a single floor), and (ii) we can deal with the fact that there are no hard a priori constraints on the types of entities that can be part of others (e.g., a room can be either part of a floor, or the property itself, like an apartment). It's worth mentioning that dependency annotations for our problem exhibit a significant number of non-projective arcs (26%), meaning that entities involved in the part-of relation are non-adjacent (i.e., interleaved by other entities), as intuitively expected.

2.3.2 Structured prediction model

We now describe the constituents of our pipeline to solve the *property tree* extraction from natural language ads, as sketched in Fig. 2.2: (1) recognize the entity mentions (Section 2.3.2.1), then (2) identify the part-of dependencies between those entity mentions (Section 2.3.2.2), and finally (3) construct the tree structure of the property (e.g., as in Fig. 2.1). In Step (2), we focus on comparing locally and globally trained graph-based models

```
Original ad:
The property includes an apartment house
with a garage. The house has living room,
kitchen and bathroom with shower.
-----------------------------------------
Structured representation:
house          | mention=apartment house
  living room  | mention=living room
  kitchen      | mention=kitchen
  bathroom     | mention=bathroom
    shower     | mention=shower
garage         | mention=garage
```

Figure 2.1: Sample unstructured ad and corresponding structured representation as a property tree.

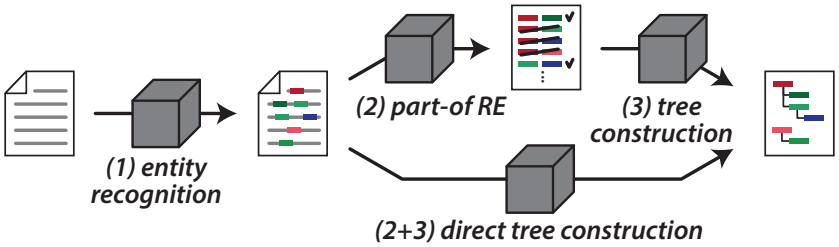

Figure 2.2: The full structured prediction pipeline.

and a transition-based one. We only explicitly perform Step (3) in graph-based models, by applying the maximum spanning-tree algorithm [19, 20] for the directed case (see McDonald et al. [7]). As an alternative, we use a transition-based system, which by definition deals with non-projective trees, and does not need spanning tree inference.

2.3.2.1 Sequence labeling

The first step in our structured prediction baseline is a sequence labeling task, similar to NER: given a real estate ad's plain text, we extract the entity mention boundaries and map the type of the entity mentions. We adopt linear chain CRFs, a special case of the CRF algorithm [3, 21], widely used for the problem of sequence labeling. We have used several features for our CRF model. A non-exhaustive list of features includes the current token,

the previous token, the next token, shape features (of previous, current and next tokens, e.g., Bill $\rightarrow$ Xxxx), various combinations of the aforementioned features (e.g., concatenation of previous and current tokens, concatenation of shape features), etc.

2.3.2.2 Part-of tree construction

The aim of this component is to connect each entity to its parent. This is similar to dependency parsing but instead of mapping the whole sentence, we map only the identified entity set x (e.g., house) to a dependency structure y. Given the entity set x with n terms, a dependency is a tuple (p, c) where $p \in \{0, ..., n\}$ is the index of the parent term in entity set x, $p = 0$ is the root-symbol (only appears as parent) and $c \in \{1, ..., n\}$ is the index of the child term in the entity set. We use $D(x)$ to refer to all possible dependencies of x and $T(x)$ to all possible dependency structures.

We now present our approaches to solve this part-of tree construction problem.

Locally trained model (Threshold/Edmonds)

We focus on local discriminative training methods [22] where a binary classifier learns the part-of relation model (Step (2)). Given a candidate parent-child pair, the classifier scores reflect how likely the part-of relation holds. The output is then used for the next and final Step (3) of constructing the *property tree*. Specifically, we construct a fully connected directed graph $G = \{V, E\}$ with the entities as nodes V, and edges E representing the part-of relation with the respective classifier scores as weights. A naive approach to obtain the tree prediction is threshold-based: keep all edges with weights exceeding a threshold. This is obviously not guaranteed to end up being a tree and might even contain cycles. Our approach directly aims at finding the maximum spanning tree inside the (directed) graph to enforce a tree structure. To this end, techniques designed for dependency parsing in natural text can be used, more in particular we use Edmonds' algorithm [7].

Globally trained model (MTT)

The Matrix-Tree theorem (MTT) [9] provides the algorithmic framework to train globally normalized models that involve directed spanning trees, i.e., score parse trees for a given sentence. Assume we have a vector θ in which each value $\theta_{h,m} \in \mathbb{R}$ corresponds to a weight $\forall(h, m) \in D(x)$, where h and m are the parent and child entities, respectively. The conditional

distribution over all dependency structures $y \in T(x)$ is:

$$P(y|x;\theta) = \frac{1}{Z(x;\theta)} \exp\left(\sum_{h,m \in y} \theta_{h,m}\right) \tag{2.1}$$

normalized by the partition function $Z(x;\theta)$, which would require a summation over the exponentially large number of all possible dependency structures in $T(x)$. However, the MTT allows directly computing $Z(x;\theta)$ as $\det(L(\theta))$, in which $L(\theta)$ is the Laplacian matrix of the graph.

Transition-based dependency parsing (TB)

Given that our system needs to be able to handle non-projective dependency arcs, we employ a greedy transition-based parsing system [12, 23] as the basis of our parser. The system is defined as a configuration $C = (\Sigma, B, A)$ which consists of Σ the stack, B the buffer and A the set of dependency arcs. The aim is, given an initial configuration and a set of permissible actions, to predict a transition sequence to some terminal configuration to derive a dependency parse tree. We define the initial configuration for an entity set $x = w_1, ..., w_n$ to be ([root],[$w_1, ..., w_n$],{}) and the terminal configuration ([0],[],A) (for any arc set A). The first three actions (LEFT-ARC, RIGHT-ARC, SHIFT) are defined similar to arc-standard systems [11] for projective dependency parsing. In addition, the SWAP operation reorders the input words, thus allowing to derive non-projective trees [12].

2.4 Experimental results

We present results for the total real estate framework as well as for each step individually.

2.4.1 Experimental setup

We collected 887,599 Dutch property advertisements from a real estate company.[1] Three human annotators manually annotated 2,318 ads (1 annotation per ad, $\sim$773 ads per annotator) by creating the property tree of the advertisements. The dataset is available for research purposes, see our Github codebase.[2] In our experiments, we use only the annotated text advertisements. We implemented the local model, the MTT and the non-projective transition-based system. The code thereof is available on

[1] https://www.realo.be/en
[2] https://github.com/bekou/ad_data

Table 2.2: Performance of the real estate entity recognition with hyperparameter $\lambda_{\text{CRF}} = 10$.

Entity type	TP	FP	FN	Precision	Recall	F_1
property	3170	1912	2217	0.62	0.59	0.61
floor	2685	515	529	0.84	0.84	0.84
space	11952	2053	2003	0.85	0.86	0.86
subspace	4338	575	1181	0.88	0.79	0.83
field	2083	700	718	0.75	0.74	0.75
extra building	253	34	143	0.88	0.64	0.74
Overall	24481	5789	6791	0.81	0.78	0.80

Github.[2] We also use our own CRF implementation. We measure precision, recall, and F_1 on the test set, and report averaged values in a 5-fold cross-validation setting.

2.4.2 Entity extraction

Table 2.2 presents our results for the sequence labeling subtask. We separately show the performance of our model for each entity type (see Table 2.1). Overall, the CRF performs well with a score of $F_1 = 0.80$. Specifically, space is the best performing entity type. Note that the space entity type is the most frequent one in our table. On the other hand, property is the least represented class, since the ads usually mention the property type only once. The performance of the property class is lower because it can have a wide range of values (e.g., "helios apartments", "milos villa"). Moreover, the entity mentions for the space type are better separable, as expected, since the mentions do not vary a lot (e.g., "shower", "bedroom").

2.4.3 Dependency parsing

The upper part of Table 2.3 lists the performance for the dependency parsing subtask by itself, assuming perfect real estate entity recognition: for this evaluation we used the gold standard provided by the annotations. We measure the performance on the threshold-based model, the logistic regression and the MTT scorings followed by Edmonds' algorithm for directed graphs to enforce a tree structure and the transition-based (TB) model. Note that in the case of known entities we have that there are exactly as many false positives as false negatives, since an incorrect edge prediction (FP) implies that the correct one has not been predicted (FN), and vice

Table 2.3: Performance of the three approaches on the structured prediction task. The top half are results for known entities (i.e., the gold standard as annotated), while the bottom half starts from the entities as found in Step (1) of our end-to-end pipeline ($\lambda_{\text{CRF}} = 10$ and $C = 1$).

	Model	TP	FP	FN	Precision	Recall	F_1
known entities	Thresh.	15723	6365	16461	0.71	0.49	0.58
	Edm.	22058	10126	10126	0.69	0.69	0.69
	MTT	22361	9823	9823	0.70	0.70	**0.70**
	TB	14816	17368	17368	0.46	0.46	0.46
full pipeline	Thresh.	9309	9846	22965	0.49	0.29	0.36
	Edm.	12859	17417	19415	0.42	0.40	**0.41**
	MTT	12426	17850	19848	0.41	0.39	0.40
	TB	9677	19043	22507	0.34	0.30	0.32

versa, because of the enforced tree structure that has to cover all entities. As expected, the MTT approach performs better than the others, because the globally trained model learns directed spanning trees. Predicting the maximum spanning tree (Edmonds') achieves higher F_1 score than simply considering the predictions of the classifier without any structural enforcement (threshold-based). The TB class of parsers is of great interest because of their speed, state-of-the-art performance [14] and the potential to be extended towards joint models (future work), although in our comparative study they tend to perform worse than the graph-based parsers, because of subsequent error propagation between the transitions [13].

2.4.4 Pipeline approach

The bottom rows in Table 2.3 refer to the pipeline approach combining both sequence labeling and dependency parsing subtasks: input entities for the parser are not necessarily correct. Given a new real estate ad, first the CRF identifies the entity mention token boundaries. Next the tree structure among the extracted entities is constructed. The locally trained approach yields marginally better performance than MTT: MTT learns spanning tree sequences as a whole, so it is harder to connect segments that are incorrect or incomplete. The TB system exhibits the same performance as in the case where entities were known, but we think that incorporating neural scoring functions [13] or using beam-search instead of using the greedy approach will improve performance [14].

2.5 Conclusion

In this paper, we presented a comparative study on the newly defined problem of extracting the structured description of real estate properties. We divided the problem into the sub-problems of sequence labeling and non-projective dependency parsing since existing joint models are restricted to non-crossing dependencies. Overall, MTT outperforms other approaches when the entities are known while adopting a maximum spanning tree algorithm using individual scored edge weights seems to be marginally better in our pipeline.

Acknowledgments

The presented research was performed within the MALIBU project, funded by Flanders Innovation & Entrepreneurship (VLAIO).

References

[1] K. Pace, R. Barry, O. W. Gilley, and C. Sirmans. *A method for spatial-temporal forecasting with an application to real estate prices*. International Journal of Forecasting, 16(2):229–246, April 2000. Available from: http://www.sciencedirect.com/science/article/pii/S0169207099000473.

[2] C. H. Nagaraja, L. D. Brown, and L. H. Zhao. *An autoregressive approach to house price modeling*. The Annals of Applied Statistics, 5(1):124–149, March 2011. Available from: http://dx.doi.org/10.1214/10-AOAS380.

[3] J. Lafferty, A. McCallum, and F. Pereira. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. In Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), pages 282–289, Massachusetts, USA, July 2001. Morgan Kaufmann.

[4] B. Taskar, C. Guestrin, and D. Koller. *Max-Margin Markov networks*. In Advances in neural information processing systems, volume 16, pages 25–32. MIT Press, 2003.

[5] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. *Support Vector Machine Learning for Interdependent and Structured Output Spaces*. In Proceedings of the Twenty-first International Conference on Machine Learning, page 104, Alberta, Canada, July 2004. ACM. Available from: http://doi.acm.org/10.1145/1015330.1015341.

[6] H. Daumé III, J. Langford, and D. Marcu. *Search-based structured prediction*. Machine Learning Journal (MLJ), 75(3):297–325, June 2009. Available from: http://dx.doi.org/10.1007/s10994-009-5106-x.

[7] R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. *Non-Projective Dependency Parsing using Spanning Tree Algorithms*. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pages 523–530, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/H05-1066.

[8] R. McDonald and F. Pereira. *Online Learning of Approximate Dependency Parsing Algorithms*. In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, pages 81–88, Trento, Italy, April 2007. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/E06-1011.

[9] T. Koo, A. Globerson, X. Carreras, and M. Collins. *Structured Prediction Models via the Matrix-Tree Theorem*. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 141–150, Prague, Czech Republic, June 2007. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/D/D07/D07-1015.

[10] W. T. Tutte. *Graph Theory*. In Encyclopedia of Mathematics and its Applications, volume 21, page 138. Cambridge University Press, 2001.

[11] J. Nivre. *An efficient algorithm for projective dependency parsing*. In Proceedings of the 8th International Workshop on Parsing Technologies, pages 149–160, Nancy, France, April 2003.

[12] J. Nivre. *Non-Projective Dependency Parsing in Expected Linear Time*. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 351–359, Suntec, Singapore, August 2009. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/P/P09/P09-1040.

[13] D. Chen and C. Manning. *A Fast and Accurate Dependency Parser using Neural Networks*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 740–750,

Doha, Qatar, October 2014. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/D14-1082.

[14] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins. *Globally Normalized Transition-Based Neural Networks*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2442–2452, Berlin, Germany, August 2016. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/P16-1231.

[15] R. J. Kate and R. Mooney. *Joint Entity and Relation Extraction Using Card-Pyramid Parsing*. In Proceedings of the Fourteenth Conference on Computational Natural Language Learning, pages 203–212, Uppsala, Sweden, July 2010. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/W10-2924.

[16] Q. Li and H. Ji. *Incremental Joint Extraction of Entity Mentions and Relations*. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 402–412, Baltimore, Maryland, June 2014. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/P14-1038.

[17] M. Miwa and Y. Sasaki. *Modeling Joint Entity and Relation Extraction with Table Representation*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1858–1869, Doha, Qatar, October 2014. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/D14-1200.

[18] F. Li, Y. Zhang, M. Zhang, and D. Ji. *Joint Models for Extracting Adverse Drug Events from Biomedical Text*. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016), pages 2838–2844, New York, USA, July 2016. IJCAI/AAAI Press. Available from: http://www.ijcai.org/Abstract/16/403.

[19] Y.-J. Chu and T.-H. Liu. *On shortest arborescence of a directed graph*. Scientia Sinica, 14:1396–1400, 1965.

[20] J. Edmonds. *Optimum branchings*. Journal of research of the National Bureau of Standards, 71B(4):233–240, 1967.

[21] F. Peng and A. McCallum. *Information extraction from research papers using conditional random fields*. Information processing & management,

42(4):963–979, July 2006. Available from: http://www.sciencedirect.com/science/article/pii/S0306457305001172.

[22] H. Yamada and Y. Matsumoto. *Statistical dependency analysis with support vector machines*. In Proceedings of the 8th International Workshop on Parsing Technologies, pages 195–206, Nancy, France, April 2003.

[23] B. Bohnet and J. Nivre. *A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing*. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1455–1465, Jeju Island, Korea, July 2012. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/D12-1133.

3

An attentive neural architecture for joint segmentation and parsing and its application to real estate ads

In this chapter we present a new joint model that is able to tackle the two core tasks (NER and dependency parsing presented in Chapter 2) for the real estate problem simultaneously. We alleviate issues identified on pipeline methods presented in Chapter 2 in constructing the property tree. In particular, we (i) avoid the error propagation between the two subtasks, and (ii) exploit the interactions between them. Note that in this chapter, we do not include transition-based systems in our experiments due to their already reported poor performance in the real estate task (see Chapter 2). Experimental results indicate an improved performance of more than three percentage points in F_1 score for the newly proposed joint model compared to the pipeline methods presented in Chapter 2. Moreover, we exploit the usage of attention methods, to encourage our model to focus on salient tokens, showcasing a further improvement of the proposed methodology for our application.

⋆⋆⋆

G. Bekoulis, J. Deleu, T. Demeester and C. Develder

Published in Expert Systems with Applications, Volume 102, 15 July 2018.

Abstract In processing human produced text using natural language processing (NLP) techniques, two fundamental subtasks that arise are (i) *segmentation* of the plain text into meaningful subunits (e.g., entities), and (ii) *dependency parsing*, to establish relations between subunits. Such structural interpretation of text provides essential building blocks for upstream expert system tasks: e.g., from interpreting textual real estate ads, one may want to provide an accurate price estimate and/or provide selection filters for end users looking for a particular property — which all could rely on knowing the types and number of rooms, etc. In this paper we develop a relatively simple and effective neural joint model that performs both segmentation and dependency parsing together, instead of one after the other as in most state-of-the-art works. We focus in particular on the real estate ad setting, aiming to convert an ad to a structured description, which we name *property tree*, comprising the tasks of (1) identifying important entities of a property (e.g., rooms) from classifieds and (2) structuring them into a tree format. In this work, we propose a new joint model that is able to tackle the two tasks simultaneously and construct the *property tree* by (i) avoiding the error propagation that would arise from the subtasks one after the other in a pipelined fashion, and (ii) exploiting the interactions between the subtasks. For this purpose, we perform an extensive comparative study of the pipeline methods and the new proposed joint model, reporting an improvement of over three percentage points in the overall edge F_1 score of the property tree. Also, we propose attention methods, to encourage our model to focus on salient tokens during the construction of the *property tree*. Thus we experimentally demonstrate the usefulness of attentive neural architectures for the proposed joint model, showcasing a further improvement of two percentage points in edge F_1 score for our application. While the results demonstrated are for the particular real estate setting, the model is generic in nature, and thus could be equally applied to other expert system scenarios requiring the general tasks of both (i) detecting entities (*segmentation*) and (ii) establishing relations among them (*dependency parsing*).

3.1 Introduction

Many consumer-oriented digital applications rely on input data provided by their target audience. For instance, real estate websites gather property descriptions for the offered classifieds, either from realtors or from indi-

vidual sellers. In such cases, it is hard to strike the right balance between structured and unstructured information: enforcing restrictions or structure upon the data format (i.e., predefined form) may reduce the amount or diversity of the data, while unstructured data (i.e., raw text) may require non-trivial (i.e., hard to automate) transformation to a more structured form to be useful/practical for the intended application. In the real estate domain, textual advertisements are an extremely useful but highly unstructured way of representing real estate properties. However, structured descriptions of the advertisements are very helpful, e.g., for real estate agencies to suggest the most appropriate sales/rentals for their customers, while keeping human reading effort limited. For example, special search filters, which are usually used by clients, cannot be directly applied to textual advertisements. On the contrary, a structured representation of the property (e.g., a tree format of the property) enables the simplification of the unstructured textual information by applying specific filters (e.g., based on the number of bedrooms, number of floors, or the requirement of having a bathroom with a toilet on the first floor), and it also benefits other related applications such as automated price prediction [1, 2].

The new real estate structured prediction problem as defined by [3] has as main goal to construct the tree-like representation of the property (i.e., the *property tree*) based on its natural language description. This can be approached as a relation extraction task by a pipeline of separate subtasks, comprising (i) named entity recognition (NER) [4] and (ii) relation extraction [5]. Unlike previous studies [6, 7] on relation extraction, in the work of [3], the relation extraction module is replaced by dependency parsing. Indeed, the relations that together define the structure of the house should form a tree, where entities are *part-of* one another (e.g., a floor is *part-of* a house, a room is *part-of* a floor). This *property tree* is structurally similar to a parse tree. Although the work of [3] is a step towards the construction of the *property tree*, it follows a pipeline setting, which suffers from two serious problems: (i) error propagation between the subtasks, i.e., NER and dependency parsing, and (ii) cross-task dependencies are not taken into account, e.g., terms indicating relations (includes, contains, etc.) between entities that can help the NER module are neglected. Due to the unidirectional nature of stacking the two modules (i.e., NER and dependency parsing) in the pipeline model, there is no information flowing from the dependency parsing to the NER subtask. This way, the parser is not able to influence the predictions of the NER. Other studies on similar tasks [6, 8] have considered the two subtasks jointly. They simultaneously extract entity mentions and relations between them usually by implementing a beam-search on top of the first module (i.e., NER), but these

methods require the manual extraction of hand-crafted features. Recently, deep learning with neural networks has received much attention. Several approaches [7, 9] apply long short-term memory (LSTM) recurrent neural networks and convolutional neural networks (CNNs) to achieve state-of-the-art performance on similar problems. Those models rely on shared parameters between the NER and relation extraction components, whereby the NER module is typically pre-trained separately, to improve the training effectiveness of the joint model.

In this work, we propose a new joint model to solve the real estate structured prediction problem. Our model is able to learn the structured prediction task without complicated feature engineering. Whereas previous studies [7, 9–11] on joint methods focus on the relation extraction problem, we construct the *property tree* which comes down to solving a dependency parsing problem, which is more constrained and hence more difficult. Therefore, previous methods are not directly comparable to our model and cannot be applied to our real estate task out-of-the-box. In this work, we treat the two subtasks as one by reformulating them into a head selection problem [12].

This paper is a follow-up work of [3]. Compared to the conference paper that introduced the real estate extraction task and applied some basic state-of-the-art techniques as a first baseline solution, we now introduce: (i) advanced neural models that consider the two subtasks jointly and (ii) modifications to the dataset annotation representations as detailed below. More specifically, the main contributions of this work are the following:

- We propose a new joint model that encodes the two tasks of identifying entities as well as dependencies between them, as a single head selection problem, without the need of parameter sharing or pre-training of the first entity recognition module separately. Moreover, instead of (i) predicting unlabeled dependencies and (ii) training an additional classifier to predict labels for the identified heads [12], our model already incorporates the dependency label predictions in its scoring formula.

- We compare the proposed joint model against established pipeline approaches and report an F_1 improvement of 1.4% in the NER and 6.2% in the dependency parsing subtask, corresponding to an overall edge F_1 improvement of 3.4% in the property tree.

- Compared to our original dataset [3], we introduce two extensions to the data: (i) we consistently assign the first mention of a particular

entity in order of appearance in the advertisement as the main mention of the entity. This results in an F_1 score increase of about 3% and 4% for the joint and pipeline models, respectively. (ii) We add the *equivalent* relation to our annotated dataset to explicitly express that several mentions across the ad may refer to the same entity.

- We perform extensive analysis of several attention mechanisms that enable our LSTM-based model to focus on informative words and phrases, reporting an improved F_1 performance of about 2.1%.

The rest of the paper is structured as follows. In Section 3.2, we review the related work. Section 3.3 defines the problem and in Section 3.4, we describe the methodology followed throughout the paper and the proposed joint model. The experimental results are reported in Section 3.5. Finally, Section 3.6 concludes our work.

3.2 Related work

The real estate structured prediction problem from textual advertisements can be broken down into the sub-problems of (i) sequence labeling (identifying the core parts of the property) and (ii) non-projective dependency parsing (connecting the identified parts into a tree-like structure) [3]. One can address these two steps either one by one in a pipelined approach, or simultaneously in a joint model. The pipeline approach is the most commonly used approach [3, 13, 14], treating the two steps independently and propagating the output of the sequence labeling subtask (e.g., named entity recognition) [15, 16] to the relation classification module [17, 18]. Joint models are able to simultaneously extract entity mentions and relations between them [6, 7]. In this work, we propose a new joint model that is able to recover the tree-like structure of the property and frame it as a dependency parsing problem, given the non-projective tree structure we aim to output. We now present related works for the sequence labeling and dependency parsing subtasks, as well as for the joint models.

3.2.1 Sequence labeling

Structured prediction problems become challenging due to the large output space. Specifically in NLP, sequence labeling (e.g., NER) is the task of identifying the entity mention boundaries and assigning a categorical label (e.g., POS tags) for each identified entity in the sentence. A number of different methods have been proposed, namely Hidden Markov Models (HMMs) [19], Conditional Random Fields (CRFs) [20], Maximum Mar-

gin Markov Network (M^3N) [21], generalized support vector machines for structured output (SVM^{struct}) [22] and Search-based Structured Prediction (SEARN) [23]. Those methods heavily rely on hand-crafted features and an in-depth review can be found in [24]. Several variations of these models that also require manual feature engineering have been used in different application settings (e.g., biology, social media context) and languages (e.g., Turkish) [25–28]. Recently, deep learning with neural networks has been succesfully applied to NER. Collobert et al. [29] proposed to use a convolutional neural network (CNN) followed by a CRF layer over a sequence of word embeddings. Recurrent Neural Networks (RNNs) constitute another neural network architecture that has attracted attention, due to the state-of-the-art performance in a series of NLP tasks (e.g., translation [30], parsing [31]). In this context, Gillick et al. [32] use a sequence-to-sequence approach for modeling the sequence labeling task. In addition, several variants of combinations between LSTM and CRF models have been proposed [16, 33, 34] achieving state-of-the-art performance on publicly available datasets.

3.2.2 Dependency parsing

Dependency parsing is a well studied task in the NLP community, which aims to analyze the grammatical structure of a sentence. We approach the problem of the *property tree* construction as a dependency parsing task i.e., to learn the dependency arcs of the classified. There are two well-established ways to address the dependency parsing problem, via graph-based and transition-based parsers.

Graph-based: In the work of [35, 36] dependency parsing requires the search of the highest scoring maximum spanning tree in graphs for both projective (dependencies are not allowed to cross) and non-projective (crossing dependencies are allowed) trees with the Eisner algorithm [37] and the Chu-Liu-Edmonds algorithm [38, 39] respectively. It was shown that exploiting higher-order information (e.g., siblings, grand-parental relation) in the graph, instead of just using first-order information (i.e., parent relations) [40, 41] may yield significant improvements of the parsing accuracy but comes at the cost of an increased model complexity. Koo et al. [42] made an important step towards globally normalized models with hand-crafted features, by adapting the Matrix-Tree Theorem (MTT) [43] to train over all non-projective dependency trees. We explore an MTT approach as one of the pipeline baselines. Similar to recent advances in neural graph-based parsing [12, 31, 44], we use LSTMs to capture richer contextual information compared to hand-crafted feature-based methods. Our work is conceptu-

ally related to [12], who formulated the dependency parsing problem as a head selection problem. We go a step further in that direction, in formulating the joint parsing and labeling problem in terms of selecting the most likely combination of head and label.

Transition-based: Transition-based parsers [45, 46] replace the exact inference of the graph-based parsers by an approximate but faster inference method. The dependency parsing problem is now solved by an abstract state machine that gradually builds up the dependency tree token by token. The goal of this kind of parsers is to find the most probable transition sequence from an initial configuration to some terminal configuration (i.e., a dependency parse tree, or in our case a *property tree*) given a permissible set of actions (i.e., LEFT-ARC, RIGHT-ARC, SHIFT) and they are able to handle both projective and non-projective dependencies [47, 48]. In the simplest case (i.e., greedy inference), a classifier predicts the next transition based on the current configuration. Compared to graph-based dependency parsers, transition-based parsers are able to scale better due to the linear time complexity while graph-based complexity rises to $O(n^2)$ in the non-projective case. Chen and Manning [49] proposed a way of learning a neural network classifier for use in a greedy, transition-based dependency parser while using low-dimensional, dense word embeddings, without the need of manually extracting features. Globally normalized transition-based parsers [50] can be considered an extension of [49], as they perform beam search for maintaining multiple hypotheses and introduce global normalization with a CRF objective. Dyer et al. [51] introduced the stack-LSTM model with push and pop operations which is able to learn the parser transition states while maintaining a summary embedding of its contents. Although transition-based systems are well-known for their speed and state-of-the-art performance, we do not include them in our study due to their already reported poor performance in the real estate task [3] compared to graph-based parsers. We hypothesize that the limited performance is due to the fact that in our problem instead of extracting features from neighboring tokens (similar to the dependency parsing), the features are local features around the non-adjacent entities. Thus, it is difficult for transition-based systems to find a complete list of transitions for extracting the correct relations by considering non-adjacent entities. Note that making one incorrect transition due to the use of local features around the entities can lead to a chain of incorrect transitions.

3.2.3 Joint learning

Adopting a pipeline strategy for the considered type of problems has two main drawbacks: (i) sequence labeling errors propagate to the dependency parsing step, e.g., an incorrectly identified part of the house (entity) could get connected to a truly existing entity, and (ii) interactions between the components are not taken into account (feedback between the subtasks), e.g., modeling the relation between two potential entities may help in deciding on the nature of the entities themselves. In more general relation extraction settings, a substantial amount of work [6, 8, 52] jointly considered the two subtasks of entity recognition and relation extraction. However, all of these models make use of hand-crafted features that: (i) require manual feature engineering, (ii) generalize poorly between various applications and (iii) may require a substantial computational cost.

Recent advances on joint models for general relation extraction consider the joint task using neural network architectures like LSTMs and CNNs [7, 9, 11]. Our work is however different from a typical relation extraction setup in that we aim to model directed spanning trees, or, equivalently, non-projective dependency structures. In particular, the entities involved in a relation are not necessarily adjacent in the text since other entities may be mentioned in between, which complicates parsing. Indeed, in this work we focus on dependency parsing due to the difficulty of establishing the tree-like structure instead of only relation extraction (where each entity can have arbitrary relation arcs, regardless of other entities and their relations), which is the case for previously cited joint models. Moreover, unlike most of these works that frame the problem as a stacking of the two components, or at least first train the NER module to recognize the entities and then further train together with the relation classification module, we include the NER directly inside the dependency parsing component.

In summary, the conceptual strengths of our *joint* segmentation and dependency parsing approach (described in detail in Section 3.4) will be the following: compared to state-of-the-art joint models in relation extraction, it (i) is generic in nature, without requiring any manual feature engineering, (ii) extracts a complete tree structure rather than a single binary relation instance.

3.3 Problem definition

In this section, we define the specific terms that are used in our real estate structured prediction problem. We define an entity as an unambigu-

Table 3.1: Real estate entity types.

Entity type	Description	Examples
property	The property.	bungalow, apartment
floor	A floor in a building.	ground floor
space	A room within the building.	bedroom, bathroom
subspace	A part of a room.	shower, toilet
field	An open space inside or outside the building.	bbq, garden
extra building	An additional building which is also part of the property.	garden house

ous, unique part of a property with independent existence (e.g., bedroom, kitchen, attic). An entity mention is defined as one or more sequential tokens (e.g., "large apartment") that can be potentially linked to one or more entities. An entity mention has a unique semantic meaning and refers to a specific entity, or a set of similar entities (e.g., "six bedrooms"). An entity itself is *part-of* another entity and can be mentioned in the text more than once with different entity mentions. For instance, a "house" entity could occur in the text with entity mentions "large villa" and "a newly built house". For the pipeline setting as presented in [3], we further classify entities into types (assign a named entity type to every word in the ad). The task is transformed to a sequence labeling problem using BIO (Beginning, Inside, Outside) encoding. The entity types are listed in Table 3.1. For instance, in the sequence of tokens "large apartment", B-PROPERTY is assigned to the token "large" as the beginning of the entity, I-PROPERTY in the token "apartment" as the inside of the entity but not the first token within the entity and O for all the other tokens that are not entities. Unlike previous studies [7, 9–11], for our joint model there is no need for this type of categorical classification into labels since the two components are treated unified as a single dependency parsing problem.

The goal of the real estate structured prediction task is to map the textual property classified into a tree-like structured representation, the so-called *property tree*, as illustrated in Fig. 3.1. In the pipeline setting, this conversion implies the detection of (i) entities of various types and (ii) the *part-of* dependencies between them. For instance, the entity "living room" is *part-of* the entity "large apartment". In the joint model, each token (e.g., "apartment", "living", "bathroom", "includes", "with", "3") is examined separately and 4 different types of relations are defined, namely *part-of*, *segment*, *skip* and *equivalent*. The *part-of* relation is similar to the way that

```
Original ad:
The property includes a large apartment with a
garage. The home has a living room, 3 spacious
bedrooms and a bathroom. The garage is equipped
with a gate and a bike wall bracket.
-----------------------------------------
Structured representation:
property              | mention=property
   apartment          | mention=large apartment, home
      living room     | mention=living room
      bedrooms        | mention=3 spacious bedrooms
      bathroom        | mention=bathroom
   garage             | mention=garage
      gate            | mention=gate
      wall bracket    | mention=bike wall bracket
```

Figure 3.1: Fictitious sample unstructured ad and corresponding structured representation as a property tree. Indentation indicates the *part-of* relations across the entities. For instance, the "apartment" is *part-of* the property while the "living room" is *part-of* the "apartment". On the left side (i.e., before the vertical bar), we denote the name of the concept for each part of the house (e.g., apartment) while on the right side (i.e., after the vertical bar), we mention the way that each concept literally exists in the text (e.g., large apartment, home). Note that the additional "ROOT" node on top of the tree has not been included to keep the example simpler.

it was defined in the pipeline setting but instead of examining entities, i.e., sequences of tokens (e.g., "living room"), we examine if a (individual) token is *part-of* another (individual) token (e.g., "room" is *part-of* the "apartment"). We encode the entity identification task with the *segment* label and we follow the same approach as in the *part-of* relationships for the joint model. Specifically, we examine if a token is a *segment* of another token (e.g., the token "room" is attached as a *segment* to the token "living", "3" is attached as a *segment* to the token "bedrooms" and "spacious" is also attached as a *segment* to the token "bedrooms" — this way we are able to encode the *segment* "3 spacious bedrooms"). By doing so, we cast the sequence labeling subtask to a dependency parsing problem. The tokens that are referring to the same entity belong to the *equivalent* relation ("home" is *equivalent* to "apartment"). Note that "home" is *equivalent* to "apartment" since in this example the "apartment" and "garage" are both *part-of* the "property". For each entity, we define the first mention in order of ap-

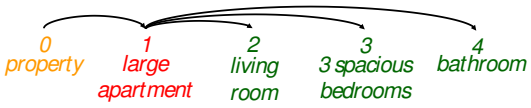

Figure 3.2: An example graph of projective *part-of* dependencies.

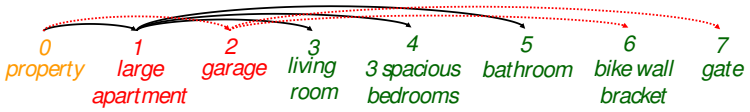

Figure 3.3: Graph representing the *part-of* dependencies of Fig. 3.1. The dashed arcs are representing the non-projective dependencies.

pearance in the text as main mention and the rest as *equivalent* to this main mention. Finally, each token that does not have any of the aforementioned types of relations has a *skip* relation with itself (e.g., "includes" has a *skip* relation with "includes"), such that each token has a uniquely defined head.

Thus, we cast the structured prediction task of extracting the *property tree* from the ad as a dependency parsing problem, where (i) an entity can be *part-of* only one (other) entity, because the decisions are taken simultaneously for all *part-of* relations (e.g., a certain room can only be *part-of* a single floor), and (ii) there are a priori no restrictions on the type of entities or tokens that can be *part-of* others (e.g., a room can be either *part-of* a floor, or the property itself, like an apartment). It is worth mentioning that dependency annotations for our problem exhibit a significant number of non-projective arcs (26%) where *part-of* dependencies are allowed to cross (see Fig. 3.3), meaning that entities involved in the *part-of* relation are non-adjacent (i.e., interleaved by other entities). For instance, all the entities or the tokens for the pipeline and the joint models, that are attached to the entity "garage" are overlapping with the entities that are attached to the entity "apartment", making parsing even more complicated: handling only projective dependencies as illustrated in Fig. 3.2 is an easier task. We note that the *segment* dependencies do not suffer from non-projectivity, since the tokens are always adjacent and sequential (e.g., "3 spacious bedrooms").

3.4 Methodology

We now describe the two approaches, i.e., the pipeline model and the joint model to construct the *property tree* of the textual advertisements, as illustrated in Fig. 3.4. For the pipeline system (Section 3.4.1), we (1) identify the entity mentions (Section 3.4.1.1), then (2) predict the *part-of* dependencies

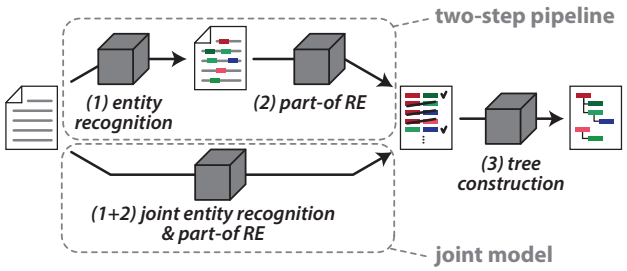

Figure 3.4: The full structured prediction system setup.

between them (Section 3.4.1.2), and finally (3) construct the tree representation (i.e., *property tree*) of the textual classified (e.g., as in Fig. 3.1). In Step (2), we apply locally or globally trained graph-based models. We represent the result of Step (2) as a graph model, and then solve Step (3) by applying the maximum spanning tree algorithm [38, 39] for the directed case (see [35]). We do not apply the well-known and fast transition-based systems with hand-crafted features for non-projective dependency structures [48, 53], given the previously established poor performance thereof in [3]. In Section 3.4.2, we describe the joint model where we perform Steps (1) and (2) jointly. For Step (3), we apply the maximum spanning tree algorithm [38, 39] similarly as to in the pipeline setting (Section 3.4.1).

3.4.1 Two-step pipeline

Below we revisit the pipeline approach presented in [3], which serves as the baseline which we compare the neural models against. As mentioned before, the pipeline model comprises two subtasks: (1) the sequence labeling and the (2) *part-of* tree construction. In the following subsections, we describe the methods applied for both.

3.4.1.1 Sequence labeling

The first step in our pipeline approach is the sequence labeling subtask which is similar to NER. Assuming a textual real estate classified, we (i) identify the entity mention boundaries and (ii) map each identified entity mention to a categorical label, i.e., entity type. In general, for sequence labeling tasks, it is beneficial to take into account correlations between labels of adjacent tokens, i.e., consider the neighborhood, and jointly find the most probable chain of labels for the given input sentence (Viterbi algorithm for the most probable assignment). For instance, in our problem where we

follow the NER standard BIO encoding [54], the I-PROPERTY cannot be followed by I-SPACE without first opening the type by B-SPACE. We use a special case of the CRF algorithm [20, 55], namely linear chain CRFs, which is commonly applied in the problem of sequence labeling to learn a direct mapping from the feature space to the output space (types) where we model label sequences jointly, instead of decoding each label independently. A linear-chain CRF with parameters w defines a conditional probability $P_w(y|x)$ for the sequence of labels $y = y_1, ..., y_N$ given the tokens of the text advertisement $x = x_1, ..., x_N$ to be

$$P_w(y|x) = \frac{1}{Z(x)} \exp(w^T \phi(x, y)), \tag{3.1}$$

where Z is the normalization constant and ϕ is the feature function that computes a feature vector given the advertisement and the sequence of labels.

3.4.1.2 Part-of tree construction

The aim of the *part-of* tree construction subtask is to link each entity to its parent. We approach the task as a dependency parsing problem but instead of connecting each token to its syntactical parent, we map only the entity set I (e.g., "large villa", "3 spacious bedrooms") that has already been extracted by the sequence labeling subtask to a dependency structure y. Assuming the entity set $I = \{e_0, e_1, ..., e_t\}$ where t is the number of identified entities, a dependency is a pair (p, c) where $p \in I$ is the parent entity and $c \in I$ is the child entity. The entity e_0 is the dummy root-symbol that only appears as parent.

We will compare two approaches to predict the *part-of* relations: a locally trained model (LTM) scoring all candidate edges independently, versus a global model (MTT) which jointly scores all edges as a whole.

Locally trained model (LTM)

In the locally trained model (LTM), we adopt a traditional local discriminative method and apply a binary classification framework [45] to learn the *part-of* relation model (Step (2)), based on standard relation extraction features such as the parent and child tokens and their types, the tokens in between, etc. For each candidate parent-child pair, the classifier gives a score that indicates whether it is probable for the *part-of* relation to hold between them. The output scores are then used for Step (3), to construct the final *property tree*. Following [35, 36], we view the entity set I as a fully connected directed graph $G = \{V, E\}$ with the entities $e_1, ..., e_t$ as vertices

(V) in the graph G, and edges E representing the *part-of* relations with the respective classifier scores as weights. One way to approach the problem is the greedy inference method where the predictions are made independently for each parent-child pair, thus neglecting that the global target output should form a *property tree*. We could adopt a threshold-based approach, i.e., keep all edges exceeding a threshold, which obviously is not guaranteed to end up with arc dependencies that form a tree structure (i.e., could even contain cycles). On the other hand, we can enforce the tree structure inside the (directed) graph by finding the maximum spanning tree. To this end, similar to [35, 36], we apply the Edmonds' algorithm to search for the most probable non-projective tree structure in the weighted fully connected graph G.

Globally trained model (MTT)

The Matrix-Tree theorem (MTT) [42] is a globally normalized statistical method that involves the learning of directed spanning trees. Unlike the locally trained models, MTT is able to learn tree dependency structures, i.e., scoring parse trees for a given sentence. We use $D(I)$ to refer to all possible dependencies of the identified entity set I, in which each dependency is represented as a tuple (h, m) in which h is the head (or parent) and m the modifier (or child). The set of all possible dependency structures for a given entity set I is written as $T(I)$. The conditional distribution over all dependency structures $y \in T(I)$ can then be defined as:

$$P(y|I;\theta) = \frac{1}{Z(I;\theta)} \exp\left(\sum_{h,m \in y} \theta_{h,m}\right) \tag{3.2}$$

in which the coefficients $\theta_{h,m} \in \mathbb{R}$ for each dependency (h, m) form the real-valued weight vector θ. The partition function $Z(I;\theta)$ is a normalization factor that alas cannot be computed by brute-force, since it requires a summation over all $y \in T(I)$, containing an exponential number of possible dependency structures. However, an adaptation of the MTT allows us the direct and efficient computation of the partition function $Z(I;\theta)$ as the determinant $det(L(\theta))$ where $L(\theta)$ is the Laplacian matrix of the graph. It is worth mentioning that although MTT learns spanning tree structures during training, at the prediction phase, it is still required to use the maximum spanning tree algorithm (Step (3)) [35, 36] as in the locally trained models.

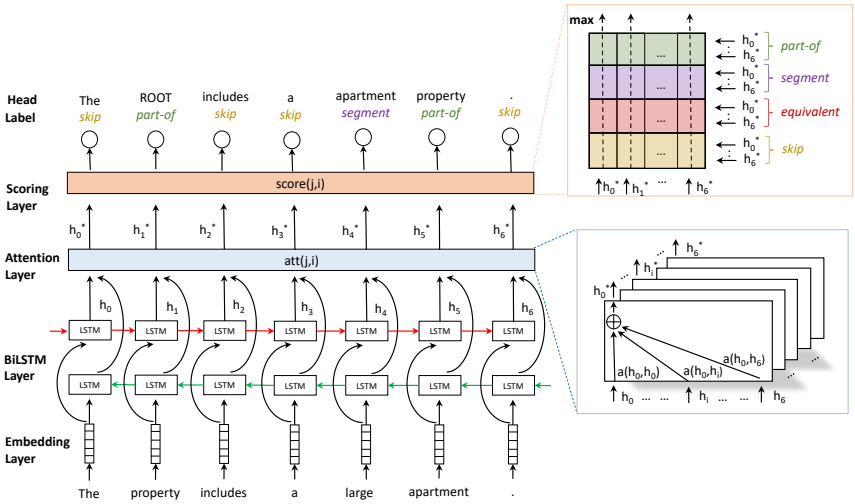

Figure 3.5: The architecture of the joint model.

3.4.2 Joint model

In this section, we present the new joint model sketched in Fig. 3.5, which simultaneously predicts the entities in the sentence and the dependencies between them, with the final goal of obtaining a tree structure, i.e., the *property tree*. We pose the problem of the identification of the entity mentions and the dependency arcs between them as a head selection problem [12]. Specifically, given as input a sentence of length N, the model outputs the predicted parent of each token of the advertisement and the most likely dependency label between them. We begin by describing how the tokens are represented in the model, i.e., with fixed pre-trained embeddings (Section 3.4.2.1), which form the input to an LSTM layer (Section 3.4.2.2). The LSTM outputs are used as input to the entity and dependency scoring layer (Section 3.4.2.3). As an extension of this model, we propose the use of various attention layers in between the LSTM and scoring layer, to encourage the model to focus on salient information, as described in Section 3.4.2.4. The final output of the joint model still is not guaranteed to form a tree structure. Therefore, we still apply Edmonds' algorithm (i.e., Step (3) from the pipeline approach), described in Section 3.4.2.5.

3.4.2.1 Embedding Layer

The embedding layer maps each token of the input sequence $x_1, ..., x_N$ of the considered advertisement to a low-dimensional vector space. We

obtain the word-level embeddings by training the Skip-Gram word2vec model [56] on a large (non-annotated) collection of property advertisements. We add a symbol x_0 in front of the N-length input sequence, which will act as the root of the property tree, and is represented with an all-zeros vector in the embedding layer.

3.4.2.2 Bidirectional LSTM encoding layer

Many neural network architectures have been proposed in literature, e.g., LSTMs [57], CNNs [58], Echo State Networks [59], or Stochastic Configuration Networks [60], to name only a few. Many others can be found in reference works on the topic [61, 62]. In this work, we use RNNs which have been proven to be particularly effective in a number of NLP tasks [7, 16, 30]. Indeed, RNNs are a common and reasonable choice to model sequential data and inherently able to cope with varying sequence lengths. Yet, plain vanilla RNNs tend to suffer from vanishing and/or exploding gradient problems and are hence not successful in capturing long-term dependencies [63, 64]. LSTMs are a more advanced kind of RNNs, which have been successfully applied in several tasks to capture long-term dependencies, as they are able to effectively overcome the vanishing gradient problem. For many NLP tasks, it is crucial to represent each word in its own context, i.e., to consider both past (left) and future (right) neighboring information. An effective solution to achieve this is using a bidirectional LSTM (BiLSTM). The basic idea is to encode each sequence from left to right (forward) and from right to left (backward). This way, there is one hidden state which represents the past information and another one for the future information. The high-level formulation of an LSTM is:

$$h_i, c_i = \text{LSTM}(w_i, h_{i-1}, c_{i-1}), \quad i = 0, ..., N \tag{3.3}$$

where in our setup $w_i \in \mathbb{R}^{\tilde{d}}$ is the word embedding for token x_i, and with the input and states for the root symbol x_0 initialized as zero vectors. Further, $h_i \in \mathbb{R}^d$ and $c_i \in \mathbb{R}^d$ respectively are the output and cell state for the ith position, where d is the hidden state size of the LSTM. Note that we chose the word embedding size the same as the LSTM hidden state size, or $\tilde{d} = d$. The outputs from left to right (forward) are written as $\vec{h_i}$ and the outputs from the backwards direction as $\overleftarrow{h_i}$. The two LSTMs' outputs at position i are concatenated to form the output h_i at that position of the BiLSTM:

$$h_i = [\vec{h_i}; \overleftarrow{h_i}], \quad i = 0, ..., N \tag{3.4}$$

3.4.2.3 Joint learning as head selection

In this subsection, we describe the joint learning task (i.e., identifying entities and predicting dependencies between them), which we formulate as a head selection problem [12]. Indeed, each word x_i should have a unique head (parent) — while it can have multiple dependent words — since the final output should form the *property tree*. Unlike the standard head selection dependency parsing framework [12], we predict the head y_i of each word x_i and the relation c_i between them jointly, instead of first obtaining binary predictions for unlabeled dependencies, followed by an additional classifier to predict the labels.

Given a text advertisement as a token sequence $x = x_0, x_1, ..., x_N$ where x_0 is the dummy root symbol, and a set $\mathcal{C} = \{part\text{-}of, segment, equivalent, skip\}$ of predefined labels (as defined in Section 3.3), we aim to find for each token $x_i,\ i \in \{0, ..., N\}$ the most probable head $x_j,\ j \in \{0, ..., N\}$ and the most probable corresponding label $c \in \mathcal{C}$. For convenience, we order the labels $c \in \mathcal{C}$ and identify them as $c_k,\ k \in \{0, ..., 3\}$. We model the joint probability of token x_j to be the head of x_i with c_k the relation between them, using a softmax:

$$P(head = x_j, label = c_k | x_i) = \frac{\exp(score(h_j, h_i, c_k))}{\sum_{\tilde{j},\tilde{k}} \exp(score(h_{\tilde{j}}, h_i, c_{\tilde{k}})} \tag{3.5}$$

where h_i and h_j are the BiLSTM encodings for words x_i and x_j, respectively. For the scoring formula $score(h_j, h_i, c_k)$ we use a neural network layer that computes the relative score between position i and j for a specific label c_k as follows:

$$score(h_j, h_i, c_k) = V_k^T \tanh(U_k h_j + W_k h_i + b_k) \tag{3.6}$$

with trainable parameters $V_k \in \mathbb{R}^l$, $U_k \in \mathbb{R}^{l \times 2d}$, $W_k \in \mathbb{R}^{l \times 2d}$, $b_k \in \mathbb{R}^l$, and l the layer width. As detailed in Section 3.5.1, we set l to be smaller than $2d$, similar to [65] due to the fact that training on superfluous information reduces the parsing speed and increases tendency towards overfitting. We train our model by minimizing the cross-entropy loss $\mathcal{L}$, written for the considered training instance as:

$$\mathcal{L} = \sum_{i=0}^{N} -\log P(head = y_i, label = c_i | x_i) \tag{3.7}$$

where $y_i \in x$ and $c_i \in \mathcal{C}$ are the ground truth head and label of x_i, respectively. After training, we follow a greedy inference approach and for each token, we simultaneously keep the highest scoring head $\hat{y}_i$ and label $\hat{c}_i$ for

x_i based on their estimated joint probability:

$$(\hat{y}_i, \hat{c}_i) = \underset{x_j \in x, c_k \in \mathcal{C}}{\operatorname{argmax}} P(head = x_j, label = c_k | x_i) \tag{3.8}$$

The predictions $(\hat{y}_i, \hat{c}_i)$ are made independently for each position i, neglecting that the final structure should be a tree. Nonetheless, as demonstrated in Section 3.5.2, the highest scoring neural models are still able to come up with a tree structure for 78% of the ads. In order to ensure a tree output in all cases, however, we apply Edmonds' algorithm on the output.

3.4.2.4 Attention Layer

The attention mechanism in our structured prediction problem aims to improve the model performance by focusing on information that is relevant to the prediction of the most probable head for each token. As attention vector, we construct the new context vector h_i^* as a weighted average of the BiLSTM outputs

$$h_j^* = \sum_{i=0}^{N} a(h_j, h_i) h_i \tag{3.9}$$

in which the coefficients $a(h_j, h_i)$, also called the attention weights, are obtained as follows:

$$a(h_j, h_i) = \frac{\exp(att(h_j, h_i))}{\sum_{\tilde{i}=0}^{N} \exp(att(h_j, h_{\tilde{i}}))}. \tag{3.10}$$

The attention function $att(h_j, h_i)$ is designed to measure some form of compatibility between the representation h_i for x_i and h_j for x_j, and the attention weights $a(h_j, h_i)$ are obtained from these scores by normalization using a softmax function. In the following, we will describe in detail the various attention models that we tested with our joint model.

Commonly used attention mechanisms

Three common attention mechanisms are listed in eqs. (3.11) to (3.13): the additive [66], bilinear, and multiplicative attention models [67], which have been extensively used in machine translation. Given the representations h_i and h_j for tokens x_i and x_j, we compute the attention scores as follows:

$$att_{additive}(h_j, h_i) = V_a \tanh(U_a h_j + W_a h_i + b_a) \tag{3.11}$$

$$att_{bilinear}(h_j, h_i) = h_j^T W_{bil} h_i \tag{3.12}$$

$$att_{multiplicative}(h_j, h_i) = h_j^T h_i \tag{3.13}$$

where $V_a \in \mathbb{R}^l$, $U_a, W_a \in \mathbb{R}^{l \times 2d}$, $W_{bil} \in \mathbb{R}^{2d \times 2d}$ and $b_a \in \mathbb{R}^l$ are learnable parameters of the model.

Biaffine attention

We use the biaffine attention model [65] which has been recently applied to dependency parsing and is a modification of the neural graph-based approach that was proposed by [31]. In this model, Dozat et al. [65] tried to reduce the dimensionality of the recurrent state of the LSTMs by applying a such neural network layer on top of them. This idea is based on the fact that there is redundant information in every hidden state that (i) reduces parsing speed and (ii) increases the risk of overfitting. To address these issues, they reduce the dimensionality and apply a nonlinearity afterwards. The deep bilinear attention mechanism is defined as follows:

$$h_i^{dep} = V_{dep} \tanh(U_{dep} h_i + b_{dep}) \tag{3.14}$$

$$h_j^{head} = V_{head} \tanh(U_{head} h_j + b_{head}) \tag{3.15}$$

$$att_{biaffine}(h_j^{head}, h_i^{dep}) = (h_j^{head})^T W_{bil} h_i^{dep} + B h_j^{head} \tag{3.16}$$

where $U_{dep}, U_{head} \in \mathbb{R}^{l \times 2d}$, $V_{dep}, V_{head} \in \mathbb{R}^{p \times l}$, $W_{bil} \in \mathbb{R}^{p \times p}$, $B \in \mathbb{R}^p$ and $b_{dep}, b_{head} \in \mathbb{R}^l$.

Tensor attention

This section introduces the Neural Tensor Network [68] that has been used as a scoring formula applied for relation classification between entities. The task can be described as link prediction between entities in an existing network of relationships. We apply the tensor scoring formula as if tokens are entities, by the following function:

$$att_{tensor}(h_j, h_i) = U_t \tanh(h_j^T W_t h_i + V_t(h_j + h_i) + b_t) \tag{3.17}$$

where $W_t \in \mathbb{R}^{2d \times l \times 2d}$, $V_t \in \mathbb{R}^{l \times 2d}$, $U_t \in \mathbb{R}^l$ and $b_t \in \mathbb{R}^l$.

Edge attention

In the edge attention model, we are inspired by [69], which applies neural message passing in chemical structures. Assuming that words are nodes inside the graph and the message flows from node x_i to x_j, we define the edge representation to be:

$$edge(h_j, h_i) = \tanh(U_e h_j + W_e h_i + b_e) \tag{3.18}$$

The edge attention formula is computed as:

$$att_{edge}(h_j, h_i) = \frac{1}{N} \left(A_{src} \sum_{\tilde{i}=0}^{N} edge(h_j, h_{\tilde{i}}) + A_{dst} \sum_{\tilde{j}=0}^{N} edge(h_{\tilde{j}}, h_i) \right) \tag{3.19}$$

where $U_e, W_e \in \mathbb{R}^{l \times 2d}$, $A_{src}, A_{dst} \in \mathbb{R}^{2d \times l}$ and $b_e \in \mathbb{R}^l$. The source and destination matrices respectively encode information for the start to the end node, in the directed edge. Running the edge attention model for several times can be achieved by stacking the edge attention layer multiple times. This is known as message passing phase and we can run it for several ($T > 1$) time steps to obtain more informative edge representations. We hypothesize that this model could potentially capture relationships between distant entities in a given sentence. This is because it has been designed to mimic traditional graph-based models presented in Chapter 2.

3.4.2.5 Tree construction step: Edmonds' algorithm

At decoding time, greedy inference is not guaranteed to end up with arc dependencies that form a tree structure and the classification decision may contain cycles. In this case, the output can be post-processed with a maximum spanning tree algorithm (as the third step in Fig. 3.4). We construct the fully connected directed graph $G = (V, E)$ where the vertices V are the tokens of the advertisement (that are not predicted as *skips*) and the dummy root symbol, E contains the edges representing the highest scoring relation (e.g., *part-of*, *segment*, *equivalent*) with the respective cross entropy scores serving as weights. Since G is a directed graph, $s(x_i, x_j)$ is not necessarily equal to $s(x_j, x_i)$. Similar to [35], we employ Edmonds' maximum spanning tree algorithm for directed graphs [38, 39] to build a non-projective parser. Indeed, in our setting, we have a significant number (26% in the dataset used for experiments, see further) of non-adjacent *part-of* and *equivalent* relations (non-projective). It is worth noting that in the case of *segment* relations, the words involved are not interleaved by other tokens and are always adjacent. We apply Edmonds' algorithm to every graph which is constructed to get the highest scoring graph structure, even in the cases where a tree is already formed by greedy inference. For *skips*, we consider the predictions as obtained from the greedy approach and we do not include them in the fully connected weighted graph, since Edmonds' complexity is $O(n^2)$ for dense graphs and might lead to slow decoding time.

3.5 Results and discussion

In this section, we present the experimental results of our study. We describe the dataset, the setup of the experiments and we compare the results of the methods analysed in the previous sections.

3.5.1 Experimental setup

Our dataset consists of a large collection (i.e., 887,599) of Dutch property advertisements from real estate agency websites. From this large dataset, a sub-collection of 2,318 classifieds have been manually annotated by 3 trained human annotators (1 annotation per ad, 773 ads per annotator). The annotations follow the format of the *property tree* that is described in detail in Section 3.3 and is illustrated in Fig. 3.1. The dataset is available for research purposes, see our Github codebase.[1] In the experiments, we use only the annotated text advertisements for the pipeline setting, i.e., LTM (locally trained model), MTT (globally trained model). In the case of the neural network approach, we train the embeddings on the large collection by using the word2vec model [56] whereas in the joint learning, we use only the annotated documents, similar to the pipeline approach. The code of the LTM and the MTT hand-crafted systems is available on Github.[1] We also use our own CRF implementation. The code for the joint model has been developed in Python with the Tensorflow machine learning library [70] and will be made public as well. For the evaluation, we use 70% for training, 15% for validation and 15% as test set. We measure the performance by computing the F_1 score on the test set. The accuracy metric can be misleading in our case since we have to deal with imbalanced data (the *skip* label is over-represented). We only report numbers on the structured classes, i.e., *segment* and *part-of* since the other dependencies (*skip*, *equivalent*) are auxiliary in the joint models and do not directly contribute to the construction of the actual *property tree*. For the overall F_1, we are again only considering the structured classes. Finally, we report the number of *property trees* (which shows how likely our model is to produce trees without applying Edmonds' algorithm, i.e., by greedy inference alone) for all the models before applying Edmonds' algorithm that guarantees the tree structure of the predictions.

For the pipeline models, we train the CRF with regularization parameter $\lambda_{CRF} = 10$ and the LTM and MTT with $C = 1$ based on the best hyperparameters on the validation set. As binary classifier, we use logistic regression. For the joint model, we train 128-dimensional word2vec embeddings on a collection of 887k advertisements. In general, using larger embeddings dimensions (e.g., 300), does not affect the performance of our models. We consistently used single-layer LSTMs through our experiments to keep our model relatively simple and to evaluate the various attention methods on top of that. We have also reported results on the joint model using a two-layer stacked LSTM joint model, although it needs a higher

[1] https://github.com/bekou/ad_data

computation time compared to a single-layer LSTM with an attention layer on top. The hidden size of the LSTMs is $d = 128$ and the size of the neural network used in the scoring and the attention layer is fixed to $l = 32$. The optimization algorithm used is Adam [71] with a learning rate of 10^{-3}. To reduce the effect of overfitting, we regularize our model using the dropout method [72]. We fix the dropout rate on the input of the LSTM layer to 0.5 to obtain significant improvements ($\sim$1%-2% F_1 score increase, depending on the model). For the two-layer LSTM, we fix the dropout rate to 0.3 in each of the input layers since this leads to largest performance increase on the validation set. We have also explored gradient clipping without further improvement on our results. In the joint model setting, we follow the evaluation strategy of early stopping [73, 74] based on the performance of the validation set. In most of the experiments, we obtain the best hyperparameters after $\sim$60 epochs.

3.5.2 Comparison of the pipeline and the joint model

One of the main contributions of our study is the comparison of the pipeline approach and the proposed joint model. We formulated the problem of identifying the entities (i.e., *segments*) and predicting the dependencies between them (i.e., construction of the *property tree*) as a joint model. Our neural model, unlike recent studies [7, 9] on joint models that use LSTMs to handle similar tasks, does not need two components to model the problem (i.e., NER and dependency parsing). To the best of our knowledge, our study is the first that formulates the task in an actual joint setting without the need to pre-train the sequence labeling component or for parameter sharing between them, since we use only one component for both subtasks. In Table 3.2, we present the results of the pipeline model (hand-crafted) and the proposed joint model (LSTM). The improvement of the joint model over the pipeline is unambiguous, i.e., 3.4% overall F_1 score difference between MTT (highest scoring pipeline model) and LSTM+E (LSTM model with Edmonds' algorithm). Note that in an additional experiment (not shown in Table 3.2), we observe that even by using randomly initialized word embeddings (i.e., without the use of pre-trained word embeddings), the joint model outperforms the pipeline methods by a large margin (i.e., the difference between the joint models with and without pre-trained word embeddings is about 1% F_1). This result indicates that the superiority of the joint model over the pipeline methods is due to the joint architecture and does not rely on transferring knowledge from a large corpus. An additional increase of $\sim$2.3% is achieved when we consider two-layer LSTMs (2xLSTM+E) for our joint model. All results in Table 3.2, except for the

LSTM, are presented using Edmonds' algorithm on top, to construct the *property tree*. Examining each label separately, we observe that the original LSTM+E model (73.7%) performs better by 1.4% in entity segmentation than the CRF (72.3%). The LSTM model achieves better performance in the entity recognition task since it has to learn the two subtasks simultaneously resulting in interactions between the components (i.e., NER and dependency parser). This way, the decisions for the entity recognition can benefit from predictions that are made for the *part-of* relations. Concerning the *part-of* dependencies, we note that the LSTMs outperform the hand-crafted approaches by 6.2%. Also, the number of valid trees that are constructed before applying Edmonds' algorithm is almost twice as high for the LSTM models. Stacking two-layer LSTMs results in an additional $\sim$1% improvement in the segmentation task and $\sim$3% in the *part-of* relations. The greedy inference for the hand-crafted methods does not produce well-formed trees, meaning that post-processing with Edmonds' algorithm (enforce tree structure) is expected to increase the performance of the hand-crafted models compared to the LSTM model performance. Indeed, the performance of the feature based hand-crafted models (i.e., LTM and MTT) without Edmonds' algorithm on top is not reported in Table 3.2 due to their poor performance in our task (i.e., $\sim$60% overall F_1 and $\sim$51% for *part-of*), but after post-processing with Edmonds' algorithm the performance significantly increases (i.e., $\sim$65%). On the other hand, applying Edmonds' algorithm on the LSTM model leads to marginally decreased performance ($\sim$0.2%) compared to the original LSTM model, probably indicating that enforcing structural constraints is not beneficial for a model that clearly has the ability to form valid tree structures during greedy inference. Although one might be tempted not to enforce the tree structure (post-process with Edmonds'), due to the nature of our problem, we have to enforce tree constraints in all of the models.

3.5.3 Comparison of the joint and the attention model

After having established the superior performance of the neural approach using LSTMs over the more traditional (LTM and MTT) methods based on hand-crafted features, we now discuss further improvements using attentive models. The attention mechanisms are designed to encourage the joint model to focus on informative tokens. We exploited several attention mechanisms as presented in Section 3.4.2.4. Table 3.2 shows the performance of the various models. Overall, the attention models are performing better in terms of overall F_1 score compared to the original joint model with Edmonds' algorithm on top. Although the performance of the Biaffine and

Table 3.2: Performance of the three approaches on the structured prediction task. The top rows are for the pipeline approach, i.e., *hand-crafted* features. The next block of results presents the results for the neural joint model based on LSTMs. The bottom block contains results of the joint models augmented with several attentive architectures. Edmonds' algorithm is applied in all of the models to retain the tree structure, except for the LSTM joint model. The LSTM+E is the LSTM model with Edmonds' algorithm included. The 2xLSTM+E is the same joint model but it simply uses a stack of two LSTM layers. In the experiments with attention, we use a one-stack LSTM. The rightmost column is the percentage of the ads that are valid trees before applying Edmonds' (i.e., Step (3) of Fig. 3.4), showing the ability of the model to form trees during greedy inference. In the Edge_i models, the number i stands for the number of times that we have run the message passing phase.

		Precision		Recall		F_1 (%)			Trees
	Model	segment	part-of	segment	part-of	segment	part-of	Overall	(% of ads)
Hand-crafted	LTM	73.77	60.53	70.98	60.40	72.35	60.47	64.76	37.18
	MTT	73.77	61.15	70.98	61.01	72.35	61.08	65.15	43.23
LSTM	LSTM	70.24	65.23	77.73	70.32	73.80	67.68	68.82	68.30
	LSTM+E	70.18	63.92	77.77	71.08	73.78	67.31	68.57	68.30
	2xLSTM+E	73.91	69.88	75.78	71.22	74.83	**70.54**	**70.90**	78.09
Attentive LSTM	Additive	72.97	65.71	76.45	70.90	74.67	68.21	69.46	74.35
	Bilinear	70.25	66.34	79.96	72.53	74.79	69.29	70.20	72.62
	Multiplicative	71.12	66.40	77.81	71.26	74.31	68.75	69.70	72.91
	Biaffine	70.01	64.67	78.32	71.04	73.93	67.71	68.75	74.06
	Tensor	71.53	64.68	76.17	70.79	73.78	67.60	68.68	69.16
	Edge_1	71.56	67.46	78.24	71.31	74.75	69.33	70.08	70.32
	Edge_2	72.03	66.09	75.35	70.99	73.65	68.46	69.12	73.48
	Edge_3	71.74	67.69	78.44	73.00	**74.94**	70.25	70.70	**78.96**

the Tensor models is limited compared to the improvement of the other attentive models, we focus on: (i) the Biaffine model since it achieved state-of-the-art performance on the dependency parsing task and (ii) the Tensor model because we were expecting that it would perform similarly to the Bilinear model (it has a bilinear tensor layer). Despite its simplicity, the Bilinear model is the second best performing attentive model in Table 3.2 in terms of overall F_1 score. Edge_3 (70.7% overall F_1 score) achieves better results than the other attention mechanisms in the entity recognition and in the dependency parsing tasks. We observe that running the message passing step multiple times in the Edge model, gives an increasing trend in the number of valid trees that were constructed before applying the maximum spanning tree algorithm. This is not surprising since we expect that running the message passing phase multiple times leads to improved edge representations. The maximum number of trees without post-processing by Edmonds' is attained when we run the message passing for 3 times whereas further increasing the number beyond 3 (e.g., 4) appears no longer beneficial. Stacking a second LSTM layer on top of the joint model

(2xLSTM+E) marginally improves the performance by 0.2% compared to the Edge_3 attention model. But adding a second LSTM layer comes with the additional cost of an increased computation time compared to the joint models with the attention layers on top. This illustrates that: (i) there might be some room for marginally improving the attention models even further, and (ii) we do not have to worry about the quadratic nature of our approach since in terms of speed the attentive models are able to surpass the two-layer LSTMs. The sequential processing of the LSTMs might be the reason that slows down the computation time for the 2xLSTM over the rest of the attentive models. Specifically, on an Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz processor, the best performing model (i.e., Edge_3) takes $\sim$2 minutes per epoch while in the 2xLSTM case, it takes $\sim$2.5 minutes leading to a slowdown of $\sim$25%. The percentage of the ads that are valid trees is 1% better in the Edge_3 over the two-layer LSTM showcasing the ability of the Edge model to form more valid trees during greedy inference.

3.5.4 Discussion

In this section, we discuss some additional aspects of our problem and the approaches that we follow to handle them. As we mentioned before, a single entity can be present in the text with multiple mentions. This brings an extra difficulty to our task. For instance, in the example of Fig. 3.1, the entity "large apartment" is expressed in the ad with the mentions "large apartment" and "home". Hence it is confusing to which mention the other entities should be attached to. One way would be to attach them to both and then eliminate one of the connections using Edmonds' spanning tree algorithm, which is the approach adopted in [3]. The problematic issue with this approach is that the spanning tree algorithm would randomly remove all mentions but one, possibly resulting in uncertain outcomes. To avoid this problem, we now use as the main mention for an entity the first mention in order of appearance in the text (e.g., "large apartment" in our example) and the remaining mentions (e.g., "home") are attached as *equivalent* mentions to the main one. Usually, the most informative mention for an entity is the one that appears first, because we again refer to an entity mentioned before, often with a shorter description. Following our intuition, the neural model increases its overall performance by $\sim$3% (from 66% to 69% and more than 5% in the *part-of* relation) and the pipeline approaches by almost 4% (from 61%, reported in [3] to 65% and more than 5% in the *part-of* relation).

We also experimented with introducing the *equivalent* relations. Although it is a strongly under-represented class in the dataset and the model

performs poorly for this label (an *equivalent* edge F_1 score of 10%), introducing the *equivalent* label is the natural way of modeling our problem (i.e., assigning each additional mention as *equivalent* to the main mention). We find out that introducing this type of relation leads to a slight decrease ($\sim$1%) in the *part-of* and a marginal increase ($\sim$0.3%) in the *segment* relations which are the main relations while retaining the nature of our problem. In the pipeline approach, it results in an 9% drop in the F_1 score of the *part-of* relation. This is the reason that the results as presented in Table 3.2 do not consider the *equivalent* relation for the hand-crafted model to make a fair comparison in the structured classes.

We believe our experimental comparison of the various architectural model variations provides useful findings for practitioners. Specifically, for applications requiring both segmentation (entity recognition) and dependency parsing (structured prediction), our findings can be qualitatively summarized as follows: (i) joint modeling is the most appropriate approach since it reduces error propagation between the components, (ii) the LSTM model is much more effective (than models relying on hand-crafted features) because it automatically extracts informative features from the raw text, (iii) attentive models are proven effective because they encourage the model to focus on salient tokens, (iv) the edge attention model leads to an improved performance since it better encodes the information flow between the entities by using graph representations, and (v) stacking a second LSTM marginally increases the performance, suggesting that there might be some room for slight improvement of the attention models by adding LSTM layers.

Finally, we point out how exactly our model relates to state-of-the-art in the field. Our joint model is able to both extract entity mentions (i.e., perform segmentation) and do dependency parsing, which we demonstrate on the real estate problem. Previous studies [6, 8, 52] that jointly considered the two subtasks (i.e., segmentation and relation extraction): (i) require manual feature engineering and (ii) generalize poorly between various applications. On the other hand, in our work, we rely on neural network methods (i.e., LSTMs) to automatically extract features from the real estate textual descriptions and perform the two tasks jointly. Although there are other methods which use neural network architectures [7, 9, 11] that focus on the relation extraction problem, our work is different in that we aim to model directed spanning trees and thus to solve the dependency parsing problem which is more constrained and difficult (than extracting single instances of binary relations). Moreover, the cited methods require either parameter sharing or pre-training of the segmentation module, which complicates learning. Therefore, cited methods are not directly comparable to

our model and cannot be applied to our real estate task out-of-the-box. However, our model's main limitation is the quadratic scoring layer that increases the time complexity of the segmentation task from linear (which is the complexity of a conditional random field, CRF) to $O(n^2)$. As a result, it sacrifices standard linear complexity of the segmentation task, in order to reduce the error propagation between the subtasks and thus perform learning in a joint, end-to-end differentiable, setting.

3.6 Conclusions

In this paper, we proposed an LSTM-based neural model to jointly perform *segmentation* and *dependency parsing*. We apply it to a real estate use case processing textual ads, thus (1) identifying important entities of the property (e.g., rooms) and (2) structuring them into a tree format based on the natural language description of the property. We compared our model with the traditional pipeline approaches that have been adapted to our task and we reported an improvement of 3.4% overall edge F_1 score. Moreover, we experimented with different attentive architectures and stacking of a second LSTM layer over our basic joint model. The results indicate that exploiting attention mechanisms that encourage our model to focus on informative tokens, improves the model performance (increase of overall edge F_1 score with $\sim$2.1%) and increases the ability to form valid trees in the prediction phase (4% to 10% more valid trees for the two best scoring attention mechanisms) before applying the maximum spanning tree algorithm.

The contribution of this study to the research in expert and intelligent systems is three-fold: (i) we introduce a generic joint model, simultaneously solving both subtasks of *segmentation* (i.e., entity extraction) and *dependency parsing* (i.e., extracting relationships among entities), that unlike previous work in the field does not rely on manually engineered features, (ii) in particular for the real estate domain, extracting a structured *property tree* from a textual ad, we refine the annotations and additionally propose attention models, compared to initial work on this application, and finally (iii) we demonstrate the effectiveness of our proposed generic joint model with extensive experiments (see aforementioned F_1 improvement of 2.1%). Despite the experimental focus on the real estate domain, we stress that the model is generic in nature, and could be equally applied to other expert system scenarios requiring the general tasks of both detecting entities (*segmentation*) and establishing relations among them (*dependency parsing*). We furthermore note that our model, rather than focusing on extracting a single binary relation from a sentence (as in traditional relation extraction settings), produces a complete tree structure.

Future work can evaluate the value of our joint model we introduced in other specific application domains (e.g., biology, medicine, news) for expert and intelligent systems. For example, the method can be evaluated for entity recognition and binary relation extraction (the ACE 04 and ACE 05 datasets; see [7]) or in adverse drug effects from biomedical texts (see [10]). In terms of model extensions and improvements, one research issue is to address the time complexity of the NER part by modifying the quadratic scoring layer for this component. An additional research direction is to investigate different loss functions for the NER component (e.g., adopting a conditional random field (CRF) approach), since this has been proven effective in the NER task on its own [16]. Moreover, the (low) performance of our model on finding coreference mentions (i.e., *equivalent* relation) can be improved by removing the *equivalent* relation from the quadratic scoring function and identifying coreference clusters using end-to-end coreference resolution models [75] in a multi-task learning setting. Especially, for the real-estate case that the documents are really long and thus the entity mentions are non-adjacent, the coreference performance can be further improved by using multi-hop coreference models (e.g., [76]) to create globally consistent coreference clusters. A final extension we envision is to enable multi-label classification of relations among entity pairs.

Acknowledgments

The presented research was partly performed within the MALIBU project, funded by Flanders Innovation & Entrepreneurship (VLAIO) contract number IWT 150630.

References

[1] K. Pace, R. Barry, O. W. Gilley, and C. Sirmans. *A method for spatial-temporal forecasting with an application to real estate prices*. International Journal of Forecasting, 16(2):229–246, 2000. doi:10.1016/S0169-2070(99)00047-3.

[2] C. H. Nagaraja, L. D. Brown, and L. H. Zhao. *An autoregressive approach to house price modeling*. The Annals of Applied Statistics, 5(1):124–149, 2011. doi:10.1214/10-AOAS380.

[3] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Reconstructing the house from the ad: Structured prediction on real estate classifieds*. In Proceedings of the 15th Conference of the European Chapter of the

Association for Computational Linguistics: (Volume 2, Short Papers), pages 274–279, Valencia, Spain, 3–7 Apr. 2017.

[4] D. Nadeau and S. Sekine. *A survey of named entity recognition and classification*. Lingvisticae Investigationes, 30(1):3–26, 2007. doi:10.1075/li.30.1.03nad.

[5] N. Bach and S. Badaskar. *A review of relation extraction*. Literature review for Language and Statistics II, 2007.

[6] Q. Li and H. Ji. *Incremental Joint Extraction of Entity Mentions and Relations*. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 402–412, Baltimore, USA, 23–25 Jun. 2014.

[7] M. Miwa and M. Bansal. *End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1105–1116, Berlin, Germany, 7–12 Aug. 2016.

[8] R. J. Kate and R. Mooney. *Joint Entity and Relation Extraction Using Card-Pyramid Parsing*. In Proceedings of the 14th Conference on Computational Natural Language Learning, pages 203–212, Uppsala, Sweden, 15–16 Jul. 2010. Association for Computational Linguistics.

[9] S. Zheng, Y. Hao, D. Lu, H. Bao, J. Xu, H. Hao, and B. Xu. *Joint entity and relation extraction based on a hybrid neural network*. Neurocomputing, 257:59–66, 2017. doi:10.1016/j.neucom.2016.12.075.

[10] F. Li, Y. Zhang, M. Zhang, and D. Ji. *Joint Models for Extracting Adverse Drug Events from Biomedical Text*. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pages 2838–2844, New York, USA, 9–15 Jul. 2016. IJCAI/AAAI Press.

[11] F. Li, M. Zhang, G. Fu, and D. Ji. *A neural joint model for entity and relation extraction from biomedical text*. BMC Bioinformatics, 18(1):1–11, 2017. doi:10.1186/s12859-017-1609-9.

[12] X. Zhang, J. Cheng, and M. Lapata. *Dependency Parsing as Head Selection*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 1, Long Papers), pages 665–676, Valencia, Spain, 3–7 Apr. 2017.

[13] K. Fundel, R. Küffner, and R. Zimmer. *RelEx-Relation extraction using dependency parse trees*. Bioinformatics, 23(3):365–371, 2007. doi:10.1093/bioinformatics/btl616.

[14] H. Gurulingappa, A. Mateen-Rajpu, and L. Toldo. *Extraction of potential adverse drug events from medical case reports*. Journal of Biomedical Semantics, 3(1):1–15, 2012. doi:10.1186/2041-1480-3-15.

[15] J. Chiu and E. Nichols. *Named Entity Recognition with Bidirectional LSTM-CNNs*. Transactions of the Association for Computational Linguistics, 4:357–370, 2016.

[16] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. *Neural Architectures for Named Entity Recognition*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270, San Diego, California, 12–17 Jun. 2016.

[17] C. dos Santos, B. Xiang, and B. Zhou. *Classifying Relations by Ranking with Convolutional Neural Networks*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 626–634, Beijing, China, 26–31 Jul. 2015.

[18] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin. *Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths*. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1785–1794, Lisbon, Portugal, 17–21 Sept. 2015. Association for Computational Linguistics.

[19] L. Rabiner and B. Juang. *An introduction to hidden Markov models*. IEEE ASSP Magazine, 3(1):4–16, 1986. doi:10.1109/MASSP.1986.1165342.

[20] J. Lafferty, A. McCallum, and F. Pereira. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. In Proceedings of the 18th International Conference on Machine Learning, pages 282–289, San Francisco, USA, 28 Jun.–1 Jul. 2001. Morgan Kaufmann.

[21] B. Taskar, C. Guestrin, and D. Koller. *Max-Margin Markov networks*. In Proceedings of the 16th International Conference on Neural Information Processing Systems, pages 25–32. MIT Press, Bangkok, Thailand, 1–5 Dec. 2003.

[22] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. *Support Vector Machine Learning for Interdependent and Structured Output Spaces*. In Proceedings of the 21st International Conference on Machine Learning, pages 104–112, Helsinki, Finland, 5–9 Jul. 2004. ACM. doi:10.1145/1015330.1015341.

[23] H. Daumé III, J. Langford, and D. Marcu. *Search-based structured prediction*. Machine Learning Journal, 75(3):297–325, 2009. doi:10.1007/s10994-009-5106-x.

[24] N. Nguyen and Y. Guo. *Comparisons of Sequence Labeling Algorithms and Extensions*. In Proceedings of the 24th International Conference on Machine Learning, pages 681–688, Corvallis, USA, 20–24 Jun. 2007. ACM. doi:10.1145/1273496.1273582.

[25] J. J. Jung. *Online named entity recognition method for microtexts in social networking services: A case study of twitter*. Expert Systems with Applications, 39(9):8066–8070, 2012. doi:10.1016/j.eswa.2012.01.136.

[26] D. Küçük and A. Yazıcı. *A hybrid named entity recognizer for Turkish*. Expert Systems with Applications, 39(3):2733–2742, 2012. doi:10.1016/j.eswa.2011.08.131.

[27] J. Atkinson and V. Bull. *A multi-strategy approach to biological named entity recognition*. Expert Systems with Applications, 39(17):12968–12974, 2012. doi:10.1016/j.eswa.2012.05.033.

[28] M. Konkol, T. Brychcín, and M. Konopík. *Latent semantics in Named Entity Recognition*. Expert Systems with Applications, 42(7):3470–3479, 2015. doi:10.1016/j.eswa.2014.12.015.

[29] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. *Natural Language Processing (Almost) from Scratch*. Journal of Machine Learning Research, 12:2493–2537, November 2011.

[30] I. Sutskever, O. Vinyals, and Q. V. Le. *Sequence to Sequence Learning with Neural Networks*. In Proceedings of the 27th International Conference on Neural Information Processing Systems, pages 3104–3112, Montreal, Canada, 08–13 Dec. 2014. MIT Press.

[31] E. Kiperwasser and Y. Goldberg. *Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations*. Transactions of the Association for Computational Linguistics, 4:313–327, 2016.

[32] D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya. *Multilingual Language Processing From Bytes*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1296–1306, San Diego, California, 12–17 Jun. 2016.

[33] Z. Huang, W. Xu, and K. Yu. *Bidirectional LSTM-CRF models for sequence tagging*. arXiv preprint arXiv:1508.01991, 2015.

[34] X. Ma and E. Hovy. *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1064–1074, Berlin, Germany, 7–12 Aug. 2016.

[35] R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. *Non-Projective Dependency Parsing using Spanning Tree Algorithms*. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pages 523–530, Vancouver, British Columbia, Canada, 06–08 Oct. 2005. Association for Computational Linguistics.

[36] R. McDonald and F. Pereira. *Online Learning of Approximate Dependency Parsing Algorithms*. In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, pages 81–88, Trento, Italy, 5–6 Apr. 2007.

[37] J. M. Eisner. *Three New Probabilistic Models for Dependency Parsing: An Exploration*. In Proceedings of the 16th International Conference on Computational Linguistics (Volume 1), pages 340–345, Copenhagen, Denmark, 5–9 Aug. 1996.

[38] Y.-J. Chu and T.-H. Liu. *On shortest arborescence of a directed graph*. Scientia Sinica, 14:1396–1400, 1965.

[39] J. Edmonds. *Optimum branchings*. Journal of research of the National Bureau of Standards, 71B(4):233–240, 1967.

[40] X. Carreras. *Experiments with a Higher-Order Projective Dependency Parser*. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 957–961, Prague, Czech, 28–30 Jun. 2007. Association for Computational Linguistics.

[41] H. Zhang and R. McDonald. *Generalized Higher-Order Dependency Parsing with Cube Pruning*. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 320–331, Jeju Island, Korea, 12–14 Jul. 2012. Association for Computational Linguistics.

[42] T. Koo, A. Globerson, X. Carreras, and M. Collins. *Structured Prediction Models via the Matrix-Tree Theorem*. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 141–150,

Prague, Czech, 28–30 Jun. 2007. Association for Computational Linguistics.

[43] W. T. Tutte. *Graph Theory*. In Encyclopedia of Mathematics and its Applications, volume 21, page 138. Cambridge University Press, 2001.

[44] W. Wang and B. Chang. *Graph-based Dependency Parsing with Bidirectional LSTM*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2306–2315, Berlin, Germany, 7–12 Aug. 2016.

[45] H. Yamada and Y. Matsumoto. *Statistical dependency analysis with support vector machines*. In Proceedings of the 8th International Workshop on Parsing Technologies, pages 195–206, Nancy, France, 23–25 Apr. 2003.

[46] J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. *Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines*. In Proceedings of the 10th Conference on Computational Natural Language Learning, pages 221–225, New York, USA, 8–9 Jun. 2006. Association for Computational Linguistics.

[47] J. Nivre. *An efficient algorithm for projective dependency parsing*. In Proceedings of the 8th International Workshop on Parsing Technologies, pages 149–160, Nancy, France, 23–25 Apr. 2003.

[48] J. Nivre. *Non-Projective Dependency Parsing in Expected Linear Time*. In Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, pages 351–359, Singapore, 2–7 Aug. 2009.

[49] D. Chen and C. Manning. *A Fast and Accurate Dependency Parser using Neural Networks*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pages 740–750, Doha, Qatar, 25–29 Oct. 2014. Association for Computational Linguistics.

[50] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins. *Globally Normalized Transition-Based Neural Networks*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2442–2452, Berlin, Germany, 7–12 Aug. 2016.

[51] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. *Transition-Based Dependency Parsing with Stack Long Short-Term Memory*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 334–343, Beijing, China, 26–31 Jul. 2015.

[52] M. Miwa and Y. Sasaki. *Modeling Joint Entity and Relation Extraction with Table Representation*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pages 1858–1869, Doha, Qatar, 25–29 Oct. 2014. Association for Computational Linguistics.

[53] B. Bohnet and J. Nivre. *A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing*. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1455–1465, Jeju Island, Korea, 12–14 Jul. 2012. Association for Computational Linguistics.

[54] L. Ratinov and D. Roth. *Design Challenges and Misconceptions in Named Entity Recognition*. In Proceedings of the 13th Conference on Computational Natural Language Learning, pages 147–155, Boulder, USA, 4–5 Jun. 2009. Association for Computational Linguistics.

[55] F. Peng and A. McCallum. *Information extraction from research papers using conditional random fields*. Information processing & management, 42(4):963–979, July 2006. doi:10.1016/j.ipm.2005.09.002?

[56] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. *Distributed Representations of Words and Phrases and their Compositionality*. In Proceedings of the 26th International Conference on Neural Information Processing Systems, pages 3111–3119, Nevada, United States, 5–10 Dec. 2013. Curran Associates, Inc.

[57] S. Hochreiter and J. Schmidhuber. *Long Short-Term Memory*. Neural computation, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[58] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. *Backpropagation Applied to Handwritten Zip Code Recognition*. Neural Computation, 1(4):541–551, Dec 1989. doi:10.1162/neco.1989.1.4.541.

[59] H. Jaeger. *The "echo state" approach to analysing and training recurrent neural networks-with an erratum note'*. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report, 148(34):13, 2010.

[60] D. Wang and M. Li. *Stochastic Configuration Networks: Fundamentals and Algorithms*. IEEE Transactions on Cybernetics, 47(10):3466–3479, Oct 2017. doi:10.1109/TCYB.2017.2734043.

[61] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[62] Y. Goldberg and G. Hirst. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[63] Y. Bengio, P. Simard, and P. Frasconi. *Learning Long-term Dependencies with Gradient Descent is Difficult*. Transactions on neural networks, 5(2):157–166, 1994. doi:10.1109/72.279181.

[64] R. Pascanu, T. Mikolov, and Y. Bengio. *On the Difficulty of Training Recurrent Neural Networks*. In Proceedings of the 30th International Conference on International Conference on Machine Learning, pages 1310–1318, Atlanta, USA, 16–21 Jun. 2013. JMLR.org.

[65] T. Dozat and C. D. Manning. *Deep biaffine attention for neural dependency parsing*. In Proceedings of the International Conference for Learning Representations, pages 1–8, Toulon, France, 24–26 Apr. 2017.

[66] O. Vinyals, M. Fortunato, and N. Jaitly. *Pointer Networks*. In Proceedings of the 28th International Conference on Neural Information Processing Systems, pages 2692–2700, Montreal, Canada, 7–12 Dec. 2015. Curran Associates, Inc.

[67] T. Luong, H. Pham, and C. D. Manning. *Effective Approaches to Attention-based Neural Machine Translation*. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421, Lisbon, Portugal, 17–21 Sept. 2015. Association for Computational Linguistics.

[68] R. Socher, D. Chen, C. D. Manning, and A. Ng. *Reasoning With Neural Tensor Networks for Knowledge Base Completion*. In Proceedings of the 26th International Conference on Neural Information Processing Systems, pages 926–934, Nevada, United States, 5–10 Dec. 2013. Curran Associates, Inc.

[69] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. *Neural Message Passing for Quantum Chemistry*. In Proceedings of the 34th International Conference on Machine Learning, pages 1263–1272, Sydney, Australia, 2017. PMLR.

[70] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. *TensorFlow: A System for Large-scale Machine Learning*. In Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, pages 265–283, Berkeley, CA, USA, 2016.

[71] D. Kingma and J. Ba. *Adam: A method for stochastic optimization*. In International Conference on Learning Representations, San Diego, USA, 2015.

[72] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[73] R. Caruana, S. Lawrence, and L. Giles. *Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping*. In Proceedings of the 13th International Conference on Neural Information Processing Systems, pages 381–387, Denver, USA, 2000. MIT Press.

[74] A. Graves, A. r. Mohamed, and G. Hinton. *Speech recognition with deep recurrent neural networks*. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing, pages 6645–6649, Vancouver, Canada, 26–31 May. 2013. doi:10.1109/ICASSP.2013.6638947.

[75] K. Lee, L. He, M. Lewis, and L. Zettlemoyer. *End-to-end Neural Coreference Resolution*. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 188–197, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. Available from: https://www.aclweb.org/anthology/D17-1018, doi:10.18653/v1/D17-1018.

[76] K. Lee, L. He, and L. Zettlemoyer. *Higher-Order Coreference Resolution with Coarse-to-Fine Inference*. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short

Papers), pages 687–692, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. Available from: https://www.aclweb.org/anthology/N18-2108, doi:10.18653/v1/N18-2108.

4A

Joint entity recognition and relation extraction as a multi-head selection problem

In this chapter, we present a new general purpose joint model that is able to reduce the quadratic complexity of the sequence labeling module presented in Chapter 3. Unlike the work presented in previous chapters (see Chapter 2 and Chapter 3) where we focused on a particular problem (i.e., recovering the tree structured description of a real estate property given its natural language description), in this chapter, we evaluate our model on various scenarios. To this end, solutions such as the MTT that have been proposed for structured prediction problems are not applicable in a more general setting. The same holds for the attentive architectures that are presented in Chapter 3 since this kind of schemes complicate the model (i.e., a simple model is always preferred compared to a more complex one) and cannot generalize well on several datasets (i.e., each attention mechanism can be beneficial in a different setting). Specifically, our model achieves state-of-the-art performance in a number of different contexts (i.e., news, biomedical, real estate) and languages (i.e., English, Dutch) without relying on any manually engineered features nor additional NLP tools.

⋆ ⋆ ⋆

G. Bekoulis, J. Deleu, T. Demeester and C. Develder

Published in Expert Systems with Applications, Volume 114, 30 December 2018.

Abstract State-of-the-art models for joint entity recognition and relation extraction strongly rely on external natural language processing (NLP) tools such as POS (part-of-speech) taggers and dependency parsers. Thus, the performance of such joint models depends on the quality of the features obtained from these NLP tools. However, these features are not always accurate for various languages and contexts. In this paper, we propose a joint neural model which performs entity recognition and relation extraction simultaneously, without the need of any manually extracted features or the use of any external tool. Specifically, we model the entity recognition task using a CRF (Conditional Random Fields) layer and the relation extraction task as a multi-head selection problem (i.e., potentially identify multiple relations for each entity). We present an extensive experimental setup, to demonstrate the effectiveness of our method using datasets from various contexts (i.e., news, biomedical, real estate) and languages (i.e., English, Dutch). Our model outperforms the previous neural models that use automatically extracted features, while it performs within a reasonable margin of feature-based neural models, or even beats them.

4A.1 Introduction

The goal of entity recognition and relation extraction is to discover relational structures of entity mentions from unstructured texts. It is a central problem in information extraction since it is critical for tasks such as knowledge base population and question answering.

The problem is traditionally approached in a pipeline setting as two separate subtasks, namely (i) named entity recognition (NER) [1] and (ii) relation extraction (RE) [2]. The main limitations of the pipeline models are: (i) error propagation between the components (i.e., NER and RE) and (ii) possible useful information from the one task is not exploited by the other (e.g., identifying a *Works for* relation might be helpful for the NER module in detecting the *type* of the two entities, i.e., Person (*PER*), Organization (*ORG*) and vice versa). On the other hand, more recent studies propose to use joint models to detect entities and their relations overcoming the aforementioned issues and achieving state-of-the-art performance [3, 4].

The previous joint models heavily rely on hand-crafted features. Recent

advances in neural networks alleviate the issue of manual feature engineering, but some of them still depend on NLP tools (e.g., POS taggers, dependency parsers). Miwa and Bansal [5] propose a Recurrent Neural Network (RNN)-based joint model that uses a bidirectional sequential LSTM (Long Short Term Memory) to model the entities and a tree-LSTM that takes into account dependency tree information to model the relations between the entities. The dependency information is extracted using an external dependency parser. Similarly, in the work of [6] for entity and relation extraction from biomedical text, a model which also uses tree-LSTMs is applied to extract dependency information. Gupta et al. [7] propose a method that relies on RNNs but uses a lot of hand-crafted features and additional NLP tools to extract features such as POS-tags, etc. Adel and Schütze [8] replicate the context around the entities with Convolutional Neural Networks (CNNs). Note that the aforementioned works examine pairs of entities for relation extraction, rather than modeling the whole sentence directly. This means that relations of other pairs of entities in the same sentence — which could be helpful in deciding on the relation *type* for a particular pair — are not taken into account. Katiyar and Cardie [9] propose a neural joint model based on LSTMs where they model the whole sentence at once, but still they do not have a principled way to deal with multiple relations. Bekoulis et al. [10] introduce a quadratic scoring layer to model the two tasks simultaneously. The limitation of this approach is that only a single relation can be assigned to a token, while the time complexity for the entity recognition task is increased compared to the standard approaches with linear complexity.

In this work, we focus on a new general purpose joint model that performs the two tasks of entity recognition and relation extraction simultaneously, and that can handle multiple relations together. Our model achieves state-of-the-art performance in a number of different contexts (i.e., news, biomedical, real estate) and languages (i.e., English, Dutch) without relying on any manually engineered features nor additional NLP tools. In summary, our proposed model (which will be detailed next in Section 4A.3) solves several shortcomings that were identified in related works (Section 4A.2) for joint entity recognition and relation extraction: (i) our model does not rely on external NLP tools nor hand-crafted features, (ii) entities and relations within the same text fragment (typically a sentence) are extracted simultaneously, where (iii) an entity can be involved in multiple relations at once.

Specifically, the model of Miwa and Bansal [5] depends on dependency parsers, which perform particularly well on specific languages (i.e., English) and contexts (i.e., news). Yet, our ambition is to develop a model that

generalizes well in various setups, therefore using only automatically extracted features that are learned during training. For instance, [5] and [6] use exactly the same model in different contexts, i.e., news (ACE04) and biomedical data (ADE), respectively. Comparing our results to the ADE dataset, we obtain a 1.8% improvement on the NER task and ∼3% on the RE task. On the other hand, our model performs within a reasonable margin (∼0.6% in the NER task and ∼1% on the RE task) on the ACE04 dataset without the use of pre-calculated features. This shows that the model of [5] strongly relies on the features extracted by the dependency parsers and cannot generalize well into different contexts where dependency parser features are weak. Comparing to Adel and Schütze [8], we train our model by modeling all the entities and the relations of the sentence at once. This type of inference is beneficial in obtaining information about neighboring entities and relations instead of just examining a pair of entities each time. Finally, we solve the underlying problem of the models proposed by [9] and [11], who essentially assume classes (i.e., relations) to be mutually exclusive: we solve this by phrasing the relation extraction component as a multi-label prediction problem.[1]

To demonstrate the effectiveness of the proposed method, we conduct the largest experimental evaluation to date (to the best of our knowledge) in jointly performing both entity recognition and relation extraction (see Section 4A.4 and Section 4A.5), using different datasets from various domains (i.e., news, biomedical, real estate) and languages (i.e., English and Dutch). Specifically, we apply our method to four datasets, namely ACE04 (news), Adverse Drug Events (ADE), Dutch Real Estate Classifieds (DREC) and CoNLL'04 (news). Our method outperforms all state-of-the-art methods that do not rely on any additional features or tools, while performance is very close (or even better in the biomedical dataset) compared to methods that do exploit hand-engineered features or NLP tools.

4A.2 Related work

The tasks of entity recognition and relation extraction can be applied either one by one in a pipeline setting [11–13] or in a joint model [4, 5, 10]. In this section, we present related work for each task (i.e., named entity recognition and relation extraction) as well as prior work into joint entity and relation extraction.

[1]Note that another difference is that we use a CRF layer for the NER part, while [9] uses a softmax and [11] uses a quadratic scoring layer; see further, when we discuss performance comparison results in Section 4A.5.

4A.2.1 Named entity recognition

In our work, NER is the first task which we solve in order to address the end-to-end relation extraction problem. A number of different methods for the NER task that are based on hand-crafted features have been proposed, such as CRFs [14], Maximum Margin Markov Networks [15] and support vector machines (SVMs) for structured output [16], to name just a few. Recently, deep learning methods such as CNN- and RNN-based models have been combined with CRF loss functions [17–20] for NER. These methods achieve state-of-the-art performance on publicly available NER datasets without relying on hand-crafted features.

4A.2.2 Relation extraction

We consider relation extraction as the second task of our joint model. The main approaches for relation extraction rely either on hand-crafted features [21, 22] or neural networks [23, 24]. Feature-based methods focus on obtaining effective hand-crafted features, for instance defining kernel functions [21, 25] and designing lexical, syntactic, semantic features, etc. [22, 26]. Neural network models have been proposed to overcome the issue of manually designing hand-crafted features leading to improved performance. CNN- [24, 27, 28] and RNN-based [29–31] models have been introduced to automatically extract lexical and sentence level features leading to a deeper language understanding. Vu et al. [32] combine CNNs and RNNs using an ensemble scheme to achieve state-of-the-art results.

4A.2.3 Joint entity and relation extraction

Entity and relation extraction includes the task of (i) identifying the entities (described in Section 4A.2.1) and (ii) extracting the relations among them (described in Section 4A.2.2). Feature-based joint models [3, 4, 33, 34] have been proposed to simultaneously solve the entity recognition and relation extraction (RE) subtasks. These methods rely on the availability of NLP tools (e.g., POS taggers) or manually designed features and thus (i) require additional effort for the data preprocessing, (ii) perform poorly in different application and language settings where the NLP tools are not reliable, and (iii) increase the computational complexity (since these models require the use of computationally-intensive external tools during inference time). In this paper, we introduce a joint neural network model to overcome the aforementioned issues and to automatically perform end-to-end relation extraction without the need of any manual feature engineering or the use of additional NLP components. Neural network approaches have been

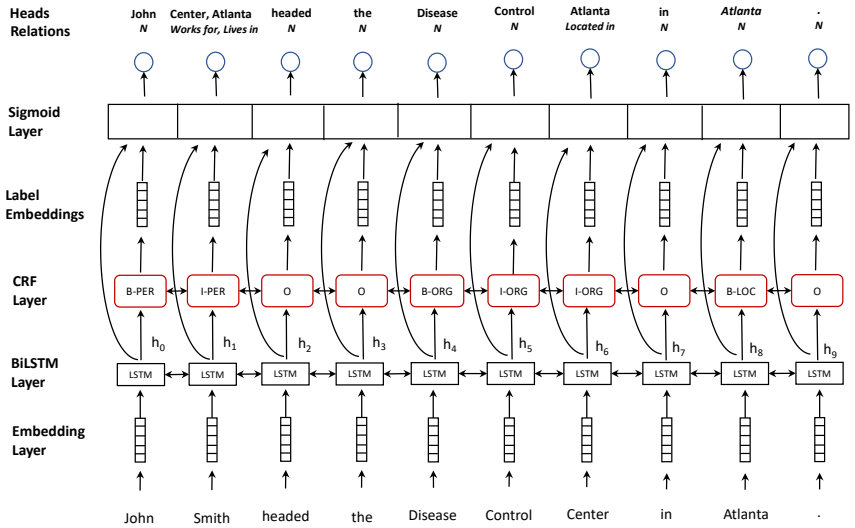

Figure 4A.1: The multi-head selection model for joint entity and relation extraction. The input of our model is the words of the sentence which are then represented as word vectors (i.e., embeddings). The BiLSTM layer extracts a left+right context aware representation for each word. Then the CRF and the sigmoid layers are able to produce the outputs for the two tasks. The outputs for each token (e.g., Smith) are: (i) an entity recognition label (e.g., I-*PER*) and (ii) a set of tuples comprising the head tokens of the entity and the types of relations between them (e.g., {(Center, *Works for*), (Atlanta, *Lives in*)}). The outputs of each subtask (i.e., NER and relation extraction) are depicted in red and blue, respectively.

considered to address the problem in a joint setting (end-to-end relation extraction) and typically include the use of RNNs and CNNs [5, 6, 35]. Specifically, Miwa and Bansal [5] propose the use of bidirectional tree-structured RNNs to capture dependency tree information (where parse trees are extracted using state-of-the-art dependency parsers) which has been proven beneficial for relation extraction [27, 31]. Li et al. [6] apply the work of Miwa and Bansal [5] to biomedical text, reporting state-of-the-art performance for two biomedical datasets. Gupta et al. [7] propose the use of a lot of hand-crafted features along with RNNs. Adel and Schütze [8] solve the entity classification task (which is different from NER since in entity classification the boundaries of the entities are known and only the *type* of the entity should be predicted) and relation extraction problems using an approximation of a global normalization objective (i.e., CRF): they replicate the context of the sentence (left and right part of the entities) to feed

one entity pair at a time to a CNN for relation extraction. Thus, they do not simultaneously infer other potential entities and relations within the same sentence. Katiyar and Cardie [9] and Bekoulis et al. [10] investigate RNNs with attention for extracting relations between entity mentions without using any dependency parse tree features.

Different from [9], in this work, we frame the problem as a multi-head selection problem by using a sigmoid loss to obtain multiple relations and a CRF loss for the NER component. This way, we are able to independently predict classes that are not mutually exclusive, instead of assigning equal probability values among the tokens. We overcome the issue of additional complexity described by [10], by dividing the loss functions into a NER and a relation extraction component. Moreover, we are able to handle multiple relations instead of just predicting single ones, as was described for the application of structured real estate advertisements of [10].

4A.3 Joint model

In this section, we present our multi-head joint model as illustrated in Fig. 4A.1. The model is able to simultaneously identify the entities (i.e., types and boundaries) and all the possible relations between them at once. We formulate the problem as a multi-head selection problem extending previous work [10, 36], overcoming the issue of additional complexity (by the NER module) and handling multiple relations together as described in Section 4A.2.3. By multi-head, we mean that any particular entity may be involved in multiple relations with other entities. The basic layers of the model, shown in Fig. 4A.1, are: (i) an embedding layer, (ii) a bidirectional sequential LSTM (BiLSTM) layer, (iii) a CRF layer and (iv) a sigmoid scoring layer. In Fig. 4A.1, an example sentence from the CoNLL04 dataset is presented. The input of our model is a sequence of tokens (i.e., words of the sentence) which are then represented as word vectors (i.e., word embeddings). The BiLSTM layer is able to extract a more complex representation for each word that incorporates the context via the RNN structure. Then the CRF and the sigmoid layers are able to produce the outputs for the two tasks. The outputs for each token (e.g., Smith) are twofold: (i) an entity recognition label (e.g., I-*PER*, denoting the token is inside a named entity of *type PER*) and (ii) a set of tuples comprising the head tokens of the entity and the types of relations between them (e.g., {(Center, *Works for*), (Atlanta, *Lives in*)}). Since we assume token-based encoding, we consider only the last token of the entity as head of another token, eliminating redundant relations. For instance, there is a *Works for* relation between entities "John Smith" and "Disease Control Center". Instead of connecting all tokens of

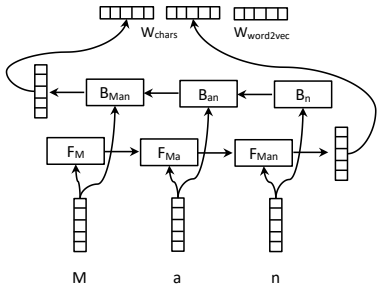

Figure 4A.2: Embedding layer in detail. The characters of the word "Man" are represented by character vectors (i.e., embeddings) that are learned during training. The character embeddings are fed to a BiLSTM and the two final states (forward and backward) are concatenated. The vector w_{chars} is the character-level representation of the word. This vector is then further concatenated to the word-level representation $w_{word2vec}$ to obtain the complete word embedding vector.

the entities, we connect only "Smith" with "Center". Also, for the case of no relation, we introduce the "N" label and we predict the token itself as the head.

4A.3.1 Embedding layer

Given a sentence $w = w_1, ..., w_n$ as a sequence of tokens, the word embedding layer is responsible to map each token to a word vector ($w_{word2vec}$). For this we use pre-trained word embeddings using the Skip-Gram word2vec model [37].

In this work, we also use character embeddings since they are commonly applied to neural NER [19, 20]. This type of embeddings is able to capture morphological features such as prefixes and suffixes. For instance, in the Adverse Drug Events (ADE) dataset, the suffix "toxicity" can specify an *adverse drug event* entity such as "neurotoxicity" or "hepatotoxicity" and thus it is very informative. Another example might be the Dutch suffix "kamer" ("room" in English) in the Dutch Real Estate Classifieds (DREC) dataset which is used to specify the *space* entities "badkamer" ("bathroom" in English) and "slaapkamer" ("bedroom" in English). Character-level embeddings are learned during training, similar to [20] and [19]. In the work of [19], character embeddings lead to a performance improvement of up to 3% in terms of NER F_1 score. In our work, by incorporating character embeddings, we report in Table 4A.2 an increase of $\sim$2% overall F_1 scoring points. For more details, see Section 4A.5.2.

Figure 4A.2 illustrates the neural architecture for word embedding generation based on its characters. The characters of each word are represented by character vectors (i.e., embeddings). The character embeddings are fed to a BiLSTM and the two final states (forward and backward) are concatenated. The vector w_{chars} is the character-level representation of the word. This vector is then further concatenated to the word-level representation $w_{word2vec}$ to obtain the complete word embedding vector.

4A.3.2 Bidirectional LSTM encoding layer

RNNs are commonly used in modeling sequential data and have been successfully applied in various NLP tasks [5, 19, 38]. In this work, we use multi-layer LSTMs, a specific kind of RNNs which are able to capture long term dependencies well [39, 40]. We employ a BiLSTM which is able to encode information from left to right (past to future) and right to left (future to past). This way, we can combine bidirectional information for each word by concatenating the forward ($\vec{h_i}$) and the backward ($\overleftarrow{h_i}$) output at timestep i. The BiLSTM output at timestep i can be written as:

$$h_i = [\vec{h_i}; \overleftarrow{h_i}], \quad i = 0, ..., n \tag{4A.1}$$

4A.3.3 Named entity recognition

We formulate the entity identification task as a sequence labeling problem, similar to previous work on joint learning models [5, 6, 9] and named entity recognition [19, 20] using the BIO (Beginning, Inside, Outside) encoding scheme. Each entity consists of multiple sequential tokens within the sentence and we should assign a tag for every token in the sentence. That way we are able to identify the entity arguments (start and end position) and its *type* (e.g., *ORG*). To do so, we assign the B-*type* (beginning) to the first token of the entity, the I-*type* (inside) to every other token within the entity and the O tag (outside) if a token is not part of an entity. Fig. 4A.1 shows an example of the BIO encoding tags assigned to the tokens of the sentence. In the CRF layer, one can observe that we assign the B-*ORG* and I-*ORG* tags to indicate the beginning and the inside tokens of the entity "Disease Control Center", respectively. On top of the BiLSTM layer, we employ either a softmax or a CRF layer to calculate the most probable entity tag for each token. We calculate the score of each token w_i for each entity tag:

$$s^{(e)}(h_i) = V^{(e)} f(U^{(e)} h_i + b^{(e)}) \tag{4A.2}$$

where the superscript (e) is used for the notation of the NER task, $f(\cdot)$ is an element-wise activation function (i.e., *relu*, *tanh*), $V^{(e)} \in \mathbb{R}^{p \times l}$, $U^{(e)} \in$

$\mathbb{R}^{l \times 2d}$, $b^{(e)} \in \mathbb{R}^l$, with d as the hidden size of the LSTM, p the number of NER tags (e.g., B-*ORG*) and l the layer width. We calculate the probabilities of all the candidate tags for a given token w_i as $\Pr(tag \mid w_i) = \text{softmax}(s(h_i))$ where $\Pr(tag \mid w_i) \in \mathbb{R}^p$. In this work, we employ the softmax approach only for the entity classification (EC) task (which is similar to NER) where we need to predict only the entity *types* (e.g., *PER*) for each token assuming boundaries are given. The CRF approach is used for the NER task which includes both entity *type* and boundaries recognition.

In the softmax approach, we assign entity *types* to tokens in a greedy way at prediction time (i.e., the selected tag is just the highest scoring tag over all possible set of tags). Although assuming an independent tag distribution is beneficial for entity classification tasks (e.g., POS tagging), this is not the case when there are strong dependencies between the tags. Specifically, in NER, the BIO tagging scheme forces several restrictions (e.g., B-*LOC* cannot be followed by I–*PER*). The softmax method allows local decisions (i.e., for the tag of each token w_i) even though the BiLSTM captures information about the neighboring words. Still, the neighboring tags are not taken into account for the tag decision of a specific token. For example, in the entity "John Smith", tagging "Smith" as *PER* is useful for deciding that "John" is B-*PER*. To this end, for NER, we use a linear-chain CRF, similar to [19] where an improvement of $\sim$1% F_1 NER points is reported when using CRF. In our case, with the use of CRF we also report a $\sim$1% overall performance improvement as observed in Table 4A.2 (see Section 4A.5.2). Assuming the word vector w, a sequence of score vectors $s_1^{(e)}, ..., s_n^{(e)}$ and a vector of tag predictions $y_1^{(e)}, ..., y_n^{(e)}$, the linear-chain CRF score is defined as:

$$S\left(y_1^{(e)}, \ldots, y_n^{(e)}\right) = \sum_{i=0}^{n} s^{(e)}_{i, y_i^{(e)}} + \sum_{i=1}^{n-1} T_{y_i^{(e)}, y_{i+1}^{(e)}} \tag{4A.3}$$

where $S \in \mathbb{R}$, $s^{(e)}_{i, y_i^{(e)}}$ is the score of the predicted tag for token w_i, T is a square transition matrix in which each entry represents transition scores from one tag to another. $T \in \mathbb{R}^{(p+2)\times(p+2)}$ because $y_0^{(e)}$ and $y_n^{(e)}$ are two auxiliary tags that represent the starting and the ending tags of the sentence, respectively. Then, the probability of a given sequence of tags over all possible tag sequences for the input sentence w is defined as:

$$\Pr\left(y_1^{(e)}, \ldots, y_n^{(e)} \;\middle|\; w\right) = \frac{e^{S(y_1^{(e)}, \ldots, y_n^{(e)})}}{\sum_{\tilde{y_1}^{(e)}, \ldots, \tilde{y_n}^{(e)}} e^{S(\tilde{y_1}^{(e)}, \ldots, \tilde{y_n}^{(e)})}} \tag{4A.4}$$

We apply Viterbi to obtain the tag sequence $\hat{y}^{(e)}$ with the highest score.

We train both the softmax (for the EC task) and the CRF layer (for NER) by minimizing the cross-entropy loss $\mathcal{L}_{\text{NER}}$. We also use the entity tags as input to our relation extraction layer by learning label embeddings, motivated by [5] where an improvement of 2% F_1 is reported (with the use of label embeddings). In our case, label embeddings lead to an increase of 1% F_1 score as reported in Table 4A.2 (see Section 4A.5.2). The input to the next layer is twofold: the output states of the LSTM and the learned label embedding representation, encoding the intuition that knowledge of named entities can be useful for relation extraction. During training, we use the gold entity tags, while at prediction time we use the predicted entity tags as input to the next layer. The input to the next layer is the concatenation of the hidden LSTM state h_i with the label embedding g_i for token w_i:

$$z_i = [h_i; g_i], \;\; i = 0, ..., n \tag{4A.5}$$

4A.3.4 Relation extraction as multi-head selection

In this subsection, we describe the relation extraction task, formulated as a multi-head selection problem [10, 36]. In the general formulation of our method, each token w_i can have multiple heads (i.e., multiple relations with other tokens). We predict the tuple $(\hat{y}_i, \hat{c}_i)$ where $\hat{y}_i$ is the vector of heads and $\hat{c}_i$ is the vector of the corresponding relations for each token w_i. This is different for the previous standard head selection for dependency parsing method [36] since (i) it is extended to predict multiple heads and (ii) the decisions for the heads and the relations are jointly taken (i.e., instead of first predicting the heads and then in a next step the relations by using an additional classifier). Given as input a token sequence w and a set of relation labels $\mathcal{R}$, our goal is to identify for each token w_i, $i \in \{0, ..., n\}$ the vector of the most probable heads $\hat{y}_i \subseteq w$ and the vector of the most probable corresponding relation labels $\hat{r}_i \subseteq \mathcal{R}$. We calculate the score between tokens w_i and w_j given a label r_k as follows:

$$s^{(r)}(z_j, z_i, r_k) = V^{(r)} f(U^{(r)} z_j + W^{(r)} z_i + b^{(r)}) \tag{4A.6}$$

where the superscript (r) is used for the notation of the relation task, $f(\cdot)$ is an element-wise activation function (i.e., *relu*, *tanh*), $V^{(r)} \in \mathbb{R}^l$, $U^{(r)} \in \mathbb{R}^{l \times (2d+b)}$, $W^{(r)} \in \mathbb{R}^{l \times (2d+b)}$, $b^{(r)} \in \mathbb{R}^l$, d is the hidden size of the LSTM, b is the size of the label embeddings and l the layer width. We define

$$\Pr\big(head = w_j, label = r_k \mid w_i\big) = \sigma(s^{(r)}(z_j, z_i, r_k)) \tag{4A.7}$$

to be the probability of token w_j to be selected as the head of token w_i with the relation label r_k between them, where $\sigma(.)$ stands for the sigmoid

function. We minimize the cross-entropy loss $\mathcal{L}_{\text{rel}}$ during training:

$$\mathcal{L}_{\text{rel}} = \sum_{i=0}^{n} \sum_{j=0}^{m} -\log \Pr\big(\textit{head} = y_{i,j}, \textit{relation} = r_{i,j} \mid w_i\big) \tag{4A.8}$$

where $y_i \subseteq w$ and $r_i \subseteq \mathcal{R}$ are the ground truth vectors of heads and associated relation labels of w_i and m is the number of relations (heads) for w_i. After training, we keep the combination of heads $\hat{y}_i$ and relation labels $\hat{r}_i$ exceeding a threshold based on the estimated joint probability as defined in Eq. (4A.7). Unlike previous work on joint models [9], we are able to predict multiple relations considering the classes as independent and not mutually exclusive (the probabilities do not necessarily sum to 1 for different classes). For the joint entity and relation extraction task, we calculate the final objective as $\mathcal{L}_{\text{NER}} + \mathcal{L}_{\text{rel}}$.

4A.3.5 Edmonds' algorithm

Our model is able to simultaneously extract entity mentions and the relations between them. To demonstrate the effectiveness and the general purpose nature of our model, we also test it on the recently introduced Dutch real estate classifieds (DREC) dataset [11] where the entities need to form a tree structure. By using thresholded inference, a tree structure of relations is not guaranteed. Thus we should enforce tree structure constraints to our model. To this end, we post-process the output of our system with Edmonds' maximum spanning tree algorithm for directed graphs [41, 42]. A fully connected directed graph $G = (V, E)$ is constructed, where the vertices V represent the last tokens of the identified entities (as predicted by NER) and the edges E represent the highest scoring relations with their scores as weights. Edmonds' algorithm is applied in cases a tree is not already formed by thresholded inference.

4A.4 Experimental setup

4A.4.1 Datasets and evaluation metrics

We conduct experiments on four datasets: (i) Automatic Content Extraction, ACE04 [43], (ii) Adverse Drug Events, ADE [44], (iii) Dutch Real Estate Classifieds, DREC [11] and (iv) the CoNLL'04 dataset with entity and relation recognition corpora [45]. Our code is available in our Github codebase.[2]

[2]https://github.com/bekou/multihead_joint_entity_relation_extraction

ACE04: There are seven main entity *types* namely Person (*PER*), Organization (*ORG*), Geographical Entities (*GPE*), Location (*LOC*), Facility (*FAC*), Weapon (*WEA*) and Vehicle (*VEH*). Also, the dataset defines seven relation *types*: Physical (*PHYS*), Person-Social (*PER-SOC*), Employment-Membership-Subsidiary (*EMP-ORG*), Agent-Artifact (*ART*), PER-ORG affiliation (*Other-AFF*), GPE affiliation (*GPE-AFF*), and Discourse (*DISC*). We follow the cross-validation setting of [3] and [5]. We removed DISC and did 5-fold cross-validation on the *bnews* and *nwire* subsets (348 documents). We obtained the preprocessing script from Miwa's Github codebase.[3] We measure the performance of our system using micro F_1 scores, Precision and Recall on both entities and relations. We treat an entity as correct when the entity type and the region of its head are correct. We treat a relation as correct when its type and argument entities are correct, similar to [5] and [9]. We refer to this type of evaluation as *strict*.[4] We select the best hyperparameter values on a randomly selected validation set for each fold, selected from the training set (15% of the data) since there are no official train and validation splits in the work of [5].

CoNLL04: There are four entity *types* in the dataset (*Location*, *Organization*, *Person*, and *Other*) and five relation *types* (*Kill*, *Live in*, *Located in*, *OrgBased in* and *Work for*). We use the splits defined by [7] and [8]. The dataset consists of 910 training instances, 243 for validation and 288 for testing.[5] We measure the performance by computing the F_1 score on the test set. We adopt two evaluation settings to compare to previous work. Specifically, we perform an EC task assuming the entity boundaries are given similar to [7] and [8]. To obtain comparable results, we omit the entity class "Other" when computing the EC score. We score a multi-token entity as correct if at least one of its comprising token *types* is correct assuming that the boundaries are given; a relation is correct when the *type* of the relation and the argument entities are both correct. We report macro-average F_1 scores for EC and RE to obtain comparable results to previous studies. Moreover, we perform actual NER evaluation instead of just EC, reporting results using the *strict* evaluation metric.

DREC: The dataset consists of 2,318 classifieds as described in the work of [10]. There are 9 entity *types*: *Neighborhood*, *Floor*, *Extra building*, *Subspace*, *Invalid*, *Field*, *Other*, *Space* and *Property*. Also, there are two relation classes *Part-of* and *Equivalent*. The goal is to identify important entities of a property (e.g., floors, spaces) from classifieds and structuring them into a tree format to get the structured description of the property. For the evaluation,

[3] https://github.com/tticoin/LSTM-ER/tree/master/data/ace2004

[4] For the CoNLL04, DREC and ADE datasets, the head region covers the whole entity (start and end boundaries). The ACE04 already defines the head region of an entity.

[5] http://cistern.cis.lmu.de/globalNormalization/globalNormalization_all.zip

we use 70% for training, 15% for validation and 15% as test set in the same splits as defined in [10]. We measure the performance by computing the F_1 score on the test set. To compare our results with previous work [10], we use the *boundaries* evaluation setting. In this setting, we count an entity as correct if the boundaries of the entity are correct. A relation is correct when the relation is correct and the argument entities are both correct. Also, we report results using the *strict* evaluation for future reference.
ADE: There are two *types* of entities (*drugs* and *diseases*) in this dataset and the aim of the task is to identify the *types* of entities and relate each *drug* with a *disease* (adverse drug events). There are 6,821 sentences in total and similar to previous work [6, 46], we remove ∼130 relations with overlapping entities (e.g., "lithium" is a drug which is related to "lithium intoxication"). Since there are no official sets, we evaluate our model using 10-fold cross-validation where 10% of the data was used as validation and 10% for test set similar to [6]. The final results are displayed in F_1 metric as a macro-average across the folds. The dataset consists of 10,652 entities and 6,682 relations. We report results similar to previous work on this dataset using the *strict* evaluation metric.

4A.4.2 Word embeddings

We use pre-trained word2vec embeddings used in previous work, so as to retain the same inputs for our model and to obtain comparable results that are not affected by the input embeddings. Specifically, we use the 200-dimensional word embeddings used in the work of [5] for the ACE04 dataset trained on Wikipedia.[6] We obtained the 50-dimensional word embeddings used by Adel and Schütze [8] trained also on Wikipedia for the CoNLL04 corpus.[5] We use the 128-dimensional word2vec embeddings used by [10] trained on a large collection of 887k Dutch property advertisements[7] for the DREC dataset. Finally, for the ADE dataset, we used 200-dimensional embeddings used by [6] and trained on a combination of texts from PubMed and PMC with texts extracted from English Wikipedia [47].[8]

4A.4.3 Hyperparameters and implementation details

We have developed our joint model by using Python with the TensorFlow machine learning library [48]. The only layer of our models that is fixed is

[6] http://tti-coin.jp/data/wikipedia200.bin
[7] https://drive.google.com/uc?id=1Dvibr-Ps4G_GI6eDx9bMXnJphGhH_M1z&export=download
[8] http://evexdb.org/pmresources/vec-space-models/wikipedia-pubmed-and-PMC-w2v.bin

the word-embeddings layer with pre-trained embeddings to have fair comparison to previous work. All other parameters of the neural network (including character embeddings) are randomly initialized and then trained. For the hyperparameter tuning, we used grid-search to find the best hyperparameters. Specifically, we defined a set of values for each of the hyperparameters (e.g., for dropout [0.1, 0.2, 0.3, 0.4, 0.5]) and we tried all the possible combinations with the different values of the hyperparameters. Training is performed using the Adam optimizer [49] with a learning rate of 10^{-3}. We fix the size of the LSTM to $d = 64$ and the layer width of the neural network to $l = 64$ (both for the entity and the relation scoring layers). We use dropout [50] to regularize our network. Dropout is applied in the input embeddings and in between the hidden layers for both tasks. Different dropout rates have been applied but the best dropout values (0.2 to 0.4) for each dataset have been used. The hidden dimension for the character-based LSTMs is 25 (for each direction). We also fixed our label embeddings to be of size $b = 25$ for all the datasets except for CoNLL04 where the label embeddings were not beneficial and thus were not used. We experimented with *tanh* and *relu* activation functions (recall that this is the function $f(\cdot)$ from the model description). We use the *relu* activation only in the ACE04 and *tanh* in all other datasets. We employ the technique of early stopping based on the validation set. In all the datasets examined in this study, we obtain the best hyperparameters after 60 to 200 epochs depending on the size of the dataset. We select the best epoch according to the results in the validation set. For more details about the effect of each hyperparameter to the model performance see the Appendix.

4A.5 Results and discussion

4A.5.1 Results

In Table 4A.1, we present the results of our analysis. Note that based on simple binomial distributions, we also compute confidence intervals for each dataset. The first column indicates the considered dataset. In the second column, we denote the model which is applied (i.e., previous work and the proposed models). The proposed models are the following: (i) *multi-head* is the proposed model with the CRF layer for NER and the sigmoid loss for multiple head prediction, (ii) *multi-head+E* is the proposed model with addition of Edmonds' algorithm to guarantee a tree-structured output for the DREC dataset, (iii) *single-head* is the proposed method but it predicts only one head per token using a softmax loss instead of a sigmoid, and (iv) *multi-head EC* is the proposed method with a softmax to predict the

entity classes assuming that the boundaries are given, and the sigmoid loss for multiple head selection. Table 4A.1 also indicates whether the different settings include hand-crafted features or features derived from NLP tools (e.g., POS taggers, dependency parsers). We use the ✓ symbol to denote that the model includes this kind of additional features and the ✗ symbol to denote that the model is only based on automatically extracted features. Note that all the variations of our model do not rely on any additional features. In the next column, we declare the type of evaluation conducted for each experiment. We include here different evaluation types to be able to compare our results against previous studies. Specifically, we use three evaluation types, namely:

(i) *Strict*: an entity is considered correct if the boundaries and the *type* of the entity are both correct; a relation is correct when the *type* of the relation and the argument entities are both correct,

(ii) *Boundaries*: an entity is considered correct if only the boundaries of the entity are correct (entity *type* is not considered); a relation is correct when the *type* of the relation and the argument entities are both correct,

(iii) *Relaxed*: we score a multi-token entity as correct if at least one of its comprising token *types* is correct assuming that the boundaries are given; a relation is correct when the *type* of the relation and the argument entities are both correct.

In the next three columns, we present the results for the entity identification task (Precision, Recall, F_1) and then (in the subsequent three columns) the results of the relation extraction task (Precision, Recall, F_1). Finally, in the last column, we report an additional F_1 measure which is the average F_1 performance of the two subtasks. We mark with bold font in Table 4A.1, the best result for each dataset among those models that use only automatically extracted features. Considering the results in the ACE04, we observe that our model outperforms the model of [9] by $\sim$2% in both tasks. This improvement can be explained by the use of the multi-head selection method which can naturally capture multiple relations and model them as a multi-label problem. Unlike the work of [9], the class probabilities do not necessarily sum up to one since the classes are considered independent. Moreover, we use a CRF-layer to model the NER task to capture dependencies between sequential tokens. Finally, we obtain more effective word representations by using character-level embeddings. On the other hand, our model performs within a reasonable margin ($\sim$0.5% for the NER task and $\sim$1% for the RE task) compared to [5]. This difference is explained by

Table 4A.1: Comparison of our method (multi-head) with the state-of-the-art on the ACE04, CoNLL04, DREC and ADE datasets. The models: (i) multi-head+E (the model + the Edmond algorithm to produce a tree-structured output), (ii) single-head (the model predicts only one head per token) and (iii) multi-head EC (the model predicts only the entity classes assuming that the boundaries are given) are slight variations of the multi-head model adapted for each dataset and evaluation. The ✓and ✗ symbols indicate whether or not the models rely on any hand-crafted features or additional tools. Note that all the variations of our models do not rely on any additional features. We include here different evaluation types (*strict*, *relaxed* and *boundaries*) to be able to compare our results against previous studies. Finally, we report results in terms of Precision, Recall, F_1 for the two subtasks as well as overall F_1, averaging over both subtasks. Bold entries indicate the best result among models that only consider automatically learned features. Note that we also compute confidence intervals for each dataset based on simple binomial distributions for the proposed models.

	Settings	Pre-calculated Features	Evaluation	Entity			Relation			Overall F_1
				P	R	F_1	P	R	F_1	
ACE 04	Miwa & Bansal (2016) [5]	✓	*strict*	80.8	82.9	81.8	48.7	48.1	48.4	65.1
	Katiyar & Cardie (2017) [9]	✗	*strict*	81.2	78.1	79.6	46.4	45.5	45.7	62.7
	multi-head	✗	*strict*	81.0	81.3	**81.2**	50.1	44.5	**47.1**	**64.2**±0.53
CoNLL 04	Gupta et al. (2016) [7]	✓	*relaxed*	92.5	92.1	92.4	78.5	63.0	69.9	81.2
	Gupta et al. (2016) [7]	✗	*relaxed*	88.5	88.9	88.8	64.6	53.1	58.3	73.6
	Adel & Schütze [8]	✗	*relaxed*	-	-	82.1	-	-	62.5	72.3
	multi-head EC	✗	*relaxed*	93.4	93.2	**93.3**	73.0	63.4	**67.0**	**80.1**±0.6
	Miwa & Sasaki (2014) [4]	✓	*strict*	81.2	80.2	80.7	76.0	50.9	61.0	70.9
	multi-head	✗	*strict*	83.8	84.1	**83.9**	63.8	60.4	**62.0**	**73.0**±0.5
DREC	Bekoulis et al. (2018) [10]	✗	*boundaries*	77.9	80.3	79.1	49.2	50.1	49.7	64.4
	multi-head+E	✗	*boundaries*	79.8	84.9	**82.3**	50.5	55.3	**52.8**	**67.5**
	single-head	✗	*strict*	78.8	84.2	**81.4**	50.5	54.3	**52.3**	**66.9**
	multi-head	✗	*strict*	78.9	83.9	81.3	50.0	54.7	52.2	66.8±0.35
ADE	Li et al. (2016) [46]	✓	*strict*	79.5	79.6	79.5	64.0	62.9	63.4	71.4
	Li et al. (2017) [6]	✓	*strict*	82.7	86.7	84.6	67.5	75.8	71.4	78.0
	multi-head	✗	*strict*	84.7	88.1	**86.4**	72.1	77.2	**74.5**	**80.4**±0.17

the fact that the model of [5] relies on POS tagging and syntactic features derived by dependency parsing. However, this kind of features relies on NLP tools that are not always accurate for various languages and contexts. For instance, the same model is adopted by the work of [6] for the ADE biomedical dataset and in this dataset our model reports more than 3% improvement in the RE task. This shows that our model is able to produce automatically extracted features which perform reasonably well in all contexts (e.g., news, biomedical).

For the CoNLL04 dataset, there are two different evaluation settings, namely *relaxed* and *strict*. In the *relaxed* setting, we perform an EC task instead of NER assuming that the boundaries of the entities are given. We adopt this setting to produce comparable results with previous studies [7, 8]. Similar to [8], we present results of single models and no ensembles. We observe that our model outperforms all previous models that do

not rely on complex hand-crafted features by a large margin (>4% for both tasks). Unlike these previous studies that consider pairs of entities to obtain the entity types and the corresponding relations, we model the whole sentence at once. That way, our method is able to directly infer all entities and relations of a sentence and benefit from their possible interactions that cannot be modeled when training is performed for each entity pair individually, one at a time. In the same setting, we also report the results of [7] in which they use multiple complicated hand-crafted features coming from NLP tools. Our model performs slightly better for the EC task and within a margin of 1% in terms of overall F_1 score. The difference in the overall performance is due to the fact that our model uses only automatically generated features. We also report results on the same dataset conducting NER (i.e., predicting entity types and boundaries) and evaluating using the *strict* evaluation measure, similar to [4]. Our results are not directly comparable to the work of [4] because we use the splits provided by [7]. However, in this setting we present the results from [4] as reference. We report an improvement of $\sim$2% overall F_1 score, which suggests that our neural model is able to extract more informative representations compared to feature-based approaches.

We also report results for the DREC dataset, with two different evaluation settings. Specifically, we use the *boundaries* and the *strict* settings. We transform the previous results from [10] to the *boundaries* setting to make them comparable to our model since in their work, they report token-based F_1 score, which is not a common evaluation metric in relation extraction problems. Also, in their work, they focus on identifying only the boundaries of the entities and not the *types* (e.g., *Floor*, *Space*). In the *boundaries* evaluation, we achieve $\sim$3% improvement for both tasks. This is due to the fact that their quadratic scoring layer is beneficial for the RE task, yet complicates NER, which is usually modeled as a sequence labeling task. Moreover, we report results using the *strict* evaluation which is used in most related works. Using the prior knowledge that each entity has only one head, we can simplify our model and predict only one head each time (i.e., using a softmax loss). The difference between the single and the multi-head models is marginal (<0.1% for both tasks). This shows that our model (multi-head) can adapt to various environments, even if the setting is single head (in terms of the application, and thus also in both training and test data).

Finally, we compare our model with previous work [6, 46] on the ADE dataset. The previous models [6, 46] both use hand-crafted features or features derived from NLP tools. However, our model is able to outperform both models using the *strict* evaluation metric. We report an improvement

Table 4A.2: Ablation tests on the ACE04 test dataset.

Settings	Entity			Relation			Overall F_1
	P	R	F_1	P	R	F_1	
Multi-head	81.0	81.3	81.1	50.1	44.4	47.1	64.1
−Label embeddings	80.6	80.9	80.7	50.0	42.9	46.1	63.4
−Character embeddings	80.4	79.5	79.9	49.0	41.6	45.0	62.5
−CRF loss	80.4	81.5	80.9	47.3	42.8	44.9	62.9

of ∼2% in the NER and ∼3% in the RE tasks, respectively. The work of [6] is similar to [5] and strongly relies on dependency parsers to extract syntactic information. A possible explanation for the better result obtained from our model is that the pre-calculated syntactic information obtained using external tools either is not so accurate or important for biomedical data.

4A.5.2 Analysis of feature contribution

We conduct ablation tests on the ACE04 dataset reported in Table 4A.2 to analyze the effectiveness of the various parts of our joint model. The performance of the RE task decreases (∼1% in terms of F_1 score) when we remove the label embeddings layer and only use the LSTM hidden states as inputs for the RE task. This shows that the NER labels, as expected, provide meaningful information for the RE component.

Removing character embeddings also degrades the performance of both NER (∼1%) and RE (∼2%) tasks by a relatively large margin. This illustrates that composing words by the representation of characters is effective, and our method benefits from additional information such as capital letters, suffixes and prefixes within the token (i.e., its character sequences).

Finally, we conduct experiments for the NER task by removing the CRF loss layer and substituting it with a softmax. Assuming independent distribution of labels (i.e., softmax) leads to a slight decrease in the F_1 performance of the NER module and a ∼2% decrease in the performance of the RE task. This happens because the CRF loss is able to capture the strong tag dependencies (e.g., I-*LOC* cannot follow B-*PER*) that are present in the dataset instead of just assuming that the tag decision for each token is independent from tag decisions of neighboring tokens.

4A.6 Conclusion

In this work, we present a joint neural model to simultaneously extract entities and relations from textual data. Our model comprises a CRF layer for the entity recognition task and a sigmoid layer for the relation extraction

task. Specifically, we model the relation extraction task as a multi-head selection problem since one entity can have multiple relations. Previous models on this task rely heavily on external NLP tools (i.e., POS taggers, dependency parsers). Thus, the performance of these models is affected by the accuracy of the extracted features. Unlike previous studies, our model produces automatically generated features rather than relying on handcrafted ones, or existing NLP tools. Given its independence from such NLP or other feature generating tools, our approach can be easily adopted for any language and context. We demonstrate the effectiveness of our approach by conducting a large scale experimental study. Our model is able to outperform neural methods that automatically generate features while the results are marginally similar (or sometimes better) compared to feature-based neural network approaches. In addition, compared to our previous work presented in Chapter 3, we reduce the NER complexity by removing the NER module from the quadratic layer. Note that although this is not mentioned in the results section, during our early experiments, we observed a better F_1 performance of our new method (over the one reported in Chapter 3). We hypothesize that this is because neighboring tokens are more likely to belong to the same segment rather than tokens that are non-adjacent. Thus, although the quadratic layer is really crucial for obtaining state-of-the-art performance for the RE task, it basically complicates NER leading to a slight performance decrease.

As future work, we aim to explore the effectiveness of entity pre-training for the entity recognition module. This approach has been proven beneficial in the work of [5] for both the entity and the relation extraction modules. Moreover, as an interesting follow-up experiment, we could stack an additional layer of NER (either softmax or CRF layer) on top of the relation module to get direct feedback from the relation extraction layer. In addition, we are planning to explore a way to reduce the calculations in the quadratic relation scoring layer. For instance, a straightforward way to do so is to use in the sigmoid layer only the tokens that have been identified as entities.

Appendix

In this section, we report additional results for our multi-head selection framework. Specifically, we (i) compare our model with the model of [19] (i.e., optimize only over the NER task), (ii) explore several hyperparameters of the network (e.g., dropout, LSTM size, character embeddings size), and (iii) report F_1 score using different word embeddings compared to the embeddings used in previous works.

In Table 4A.1 of the main paper, we focused on comparing our model against other *joint* models that are able to solve the two tasks (i.e., NER and relation extraction) simultaneously, mainly demonstrating superiority of phrasing the relation extraction as a multi-head selection problem (enabling the extraction of multiple relations at once). Here, in Table 4A.3, we evaluate the performance of just the first module of our joint multi-head model: we compare the performance of the NER component of our model against the state-of-the-art NER model of [19]. The results indicate a marginal performance improvement of our model over Lample's NER baseline in 3 out of 4 datasets. The improvement of our model's NER part is not substantial, since (i) our NER part is almost identical to Lample's, and (ii) recent advances in NER performance among neural systems are relatively small (improvements in the order of few 0.1 F_1 points – for instance, the contribution of [20] and [19] on the CoNLL-2003 test set is 0.01% and 0.17% F_1 points, respectively). This slight improvement suggests that the interaction of the two components by sharing the underlying LSTM layer is indeed beneficial (e.g., identifying a *Works for* relation might be helpful for the NER module in detecting the *type* of the two entities, i.e., *PER*, *ORG* and vice versa). Note that improving NER in isolation was not the objective of our multi-head model, but we rather aimed to compare our model against other joint models that solve the task of entity recognition and relation identification *simultaneously*. We thus did not envision to claim or achieve state-of-the-art performance in each of the individual building blocks of our joint model.

Tables 4A.4, 4A.5 and 4A.6 show the performance of our model on the test set for different values of the embedding dropout, LSTM layer dropout and the LSTM output dropout hyperparameters, respectively. Note that the hyperparameter values used for the results in Section 4A.5 were obtained by tuning over the development set, and these are indicated in bold face in the tables below. We vary one hyperparameter at a time in order to assess the effect of a particular hyperparameter. The main outcomes from these tables are twofold: (i) low dropout values (e.g., 0, 0.1) lead to a performance decrease in the overall F_1 score (see Table 4A.5 where a $\sim$3% F_1 decrease is reported on the ACE04 dataset) and (ii) average dropout values (i.e., 0.2-0.4) lead to consistently similar results.

In Tables 4A.7, 4A.8, 4A.9 and 4A.10, we report results for different values of the LSTM size, the size of the character embeddings, the size of the label embeddings and the layer width of the neural network l (both for the entity and the relation scoring layers), respectively. The reported results show that different hyperparameters settings do lead to noticeable performance differences, but we do not observe any clear trend. Moreover, we

have not observed any significant performance improvement that affects the overall ranking of the models as reported in Table 4A.1. On the other hand, the results indicate that increasing (character and label) embedding size and layer dimensions leads to a slight decrease in performance for the CoNLL04 dataset. This can be explained by the fact that the CoNLL04 dataset is relatively small and using more trainable model parameters (i.e., larger hyperparameter values) can make our multi-head selection method to overfit quickly on the training set. In almost any other case, variation of the hyperparameters does not affect the ranking of the models reported in Table 4A.1.

Table 4A.3: Comparison of the multi-head selection model (only the NER component) against the NER baseline of [19]. Bold font indicates the best results for each dataset.

	Model	Entity		
		P	R	F_1
ACE 04	NER baseline	81.0	81.1	81.1
	multi-head	81.0	81.3	**81.1**
CoNLL 04	NER baseline	84.3	83.1	83.7
	multi-head	83.7	84.0	**83.9**
DREC	NER baseline	78.2	84.8	**81.4**
	multi-head	78.9	83.9	81.3
ADE	NER baseline	83.9	88.5	86.2
	multi-head	84.7	88.1	**86.4**

Table 4A.4: Model performance for different embedding dropout values. Bold entries indicate the result reported in Section 4A.5.

	Embedding Dropout	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	0.5	80.6	81.0	80.8	47.6	43.2	45.3	63.1
	0.4	80.9	81.3	81.1	49.9	43.5	46.5	63.8
	0.3	81.0	81.3	**81.1**	50.1	44.4	**47.1**	**64.1**
	0.2	81.1	81.5	81.3	49.8	42.4	45.8	63.5
	0.1	80.8	81.0	80.9	47.7	42.9	45.2	63.0
	0	80.2	80.4	80.3	47.0	43.5	45.2	62.7
CoNLL 04	0.5	82.5	83.6	83.0	69.2	52.3	59.6	71.3
	0.4	83.6	83.0	83.3	65.1	51.4	57.4	70.4
	0.3	82.1	84.2	83.2	64.7	57.8	61.0	72.1
	0.2	84.0	84.6	84.3	71.9	54.7	62.1	73.2
	0.1	83.7	84.0	**83.9**	63.7	60.4	**62.0**	**72.9**
	0	82.7	84.7	83.7	66.2	56.6	61.0	72.3
DREC	0.5	78.1	84.5	81.2	51.1	53.8	52.4	66.8
	0.4	78.4	84.7	81.4	51.8	53.5	52.7	67.1
	0.3	78.9	83.9	**81.3**	50.0	54.7	**52.2**	**66.8**
	0.2	78.1	84.1	81.0	51.6	54.1	52.8	66.9
	0.1	78.8	83.3	81.0	49.3	52.6	50.9	66.0
	0	78.4	82.3	80.3	50.6	52.6	51.5	65.9
ADE	0.5	84.7	88.6	86.6	72.6	78.8	75.6	81.1
	0.4	84.5	88.2	86.3	71.9	77.9	74.8	80.5
	0.3	84.7	88.1	**86.4**	72.1	77.2	**74.5**	**80.4**
	0.2	84.6	87.9	86.2	72.3	77.3	74.8	80.5
	0.1	85.1	87.4	86.2	72.9	76.7	74.7	80.5
	0	83.6	87.0	85.3	71.0	75.9	73.4	79.3

Table 4A.5: Model performance for different LSTM layer dropout values. Bold entries indicate the result reported in Section 4A.5.

	LSTM	Entity			Relation			
	Dropout	P	R	F_1	P	R	F_1	Overall F_1
ACE 04	0.5	80.2	80.0	80.1	48.2	38.8	43.0	61.6
	0.4	81.1	81.3	81.2	50.5	42.0	45.9	63.5
	0.3	81.1	81.6	81.4	50.3	44.1	47.0	64.2
	0.2	81.0	81.3	**81.1**	50.1	44.4	**47.1**	**64.1**
	0.1	81.2	81.3	81.2	48.2	41.5	44.6	62.9
	0	80.5	79.9	80.2	46.7	39.3	42.7	61.4
CoNLL 04	0.5	84.1	86.2	85.2	59.3	60.1	59.7	72.4
	0.4	84.4	85.4	84.9	63.7	62.5	63.1	74.0
	0.3	86.4	85.7	86.0	65.1	60.6	62.8	74.4
	0.2	84.7	85.9	85.3	68.0	59.4	63.4	74.3
	0.1	83.7	84.0	**83.9**	63.7	60.4	**62.0**	**72.9**
	0	84.1	82.7	83.4	65.0	52.1	57.8	70.6
DREC	0.5	77.7	84.8	81.1	49.4	53.6	51.4	66.3
	0.4	78.6	83.9	81.2	50.6	54.6	52.5	66.8
	0.3	78.9	83.9	**81.3**	50.0	54.7	**52.2**	**66.8**
	0.2	77.8	83.6	80.6	49.2	53.7	51.3	66.0
	0.1	78.9	83.6	81.2	51.3	53.1	52.2	66.7
	0	78.5	80.1	79.3	50.3	49.9	50.1	64.7
ADE	0.5	85.0	88.2	86.6	72.7	78.1	75.3	80.9
	0.4	84.6	88.3	86.4	72.2	78.0	74.9	80.7
	0.3	84.6	88.6	86.5	72.2	78.8	75.3	80.9
	0.2	84.7	88.1	**86.4**	72.1	77.2	**74.5**	**80.4**
	0.1	84.3	87.9	86.1	72.0	77.5	74.6	80.4
	0	83.8	87.6	85.6	70.5	76.9	73.6	79.6

Table 4A.6: Model performance for different LSTM output dropout values. Bold entries indicate the best result reported in Section 4A.5.

	LSTM output	Entity			Relation			
	Dropout	P	R	F_1	P	R	F_1	Overall F_1
ACE 04	0.5	81.2	81.7	81.5	51.1	41.9	46.0	63.8
	0.4	81.2	81.7	81.4	51.4	42.7	46.7	64.0
	0.3	81.3	81.7	81.5	48.6	44.2	46.3	63.9
	0.2	81.0	81.3	**81.1**	50.1	44.4	**47.1**	**64.1**
	0.1	81.0	81.1	81.0	47.5	42.8	45.0	63.0
	0	80.1	80.6	80.3	47.2	40.5	43.6	62.0
CoNLL 04	0.5	85.8	86.8	86.3	64.1	59.0	61.4	73.9
	0.4	83.2	84.8	84.08	66.0	61.3	63.6	73.8
	0.3	85.1	84.8	85.0	64.8	55.4	59.7	72.3
	0.2	84.1	84.5	84.3	66.0	57.5	61.5	72.9
	0.1	83.7	84.0	**83.9**	63.7	60.4	**62.0**	**72.9**
	0	83.6	84.8	84.2	65.2	53.7	58.9	71.6
DREC	0.5	78.7	84.2	81.3	51.2	52.6	51.9	66.6
	0.4	78.4	85.2	81.6	50.3	55.4	52.7	67.2
	0.3	78.9	83.9	**81.3**	50.0	54.7	**52.2**	**66.8**
	0.2	77.8	84.6	81.1	51.0	54.1	52.5	66.8
	0.1	78.8	83.7	81.2	51.7	54.7	53.2	67.2
	0	77.6	83.8	80.6	51.1	51.3	51.2	65.9
ADE	0.5	84.3	87.9	86.1	71.5	77.2	74.2	80.2
	0.4	85.1	88.1	86.6	72.8	77.8	75.2	80.9
	0.3	84.2	88.0	86.1	71.8	77.4	74.5	80.3
	0.2	84.7	88.1	**86.4**	72.1	77.2	**74.5**	**80.4**
	0.1	84.6	88.0	86.3	72.3	77.4	74.8	80.5
	0	84.4	88.1	86.2	71.6	77.8	74.6	80.4

Table 4A.7: Model performance for different LSTM size values. Bold entries indicate the result reported in Section 4A.5.

	LSTM Size	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	32	80.9	81.2	81.1	50.3	42.6	46.1	63.6
	64	81.0	81.3	**81.1**	50.1	44.4	**47.1**	**64.1**
	128	80.3	80.8	80.5	47.3	41.7	44.3	62.4
CoNLL 04	32	82.8	83.1	82.9	65.7	58.2	61.8	72.3
	64	83.7	84.0	**83.9**	63.7	60.4	**62.0**	**72.9**
	128	82.4	83.0	82.7	64.8	53.7	58.8	70.7
DREC	32	77.7	85.4	81.4	50.9	52.3	51.6	66.5
	64	78.9	83.9	**81.3**	50.0	54.7	**52.2**	**66.8**
	128	79.0	83.4	81.2	51.2	53.6	52.4	66.8
ADE	32	83.8	87.7	85.7	70.4	76.8	73.5	79.6
	64	84.7	88.1	**86.4**	72.1	77.2	**74.5**	**80.4**
	128	84.2	87.8	86.0	71.3	76.7	73.9	80.0

Table 4A.8: Model performance for different character embeddings size values. Bold entries indicate the result reported in Section 4A.5.

	Character Embeddings	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	15	81.0	81.5	81.2	47.8	44.7	46.2	63.7
	25	81.0	81.3	**81.1**	50.1	44.4	**47.1**	**64.1**
	50	81.3	81.5	81.4	49.7	44.0	46.7	64.0
CoNLL 04	15	83.3	84.3	83.8	66.0	57.1	61.2	72.5
	25	83.7	84.0	**83.9**	63.7	60.4	**62.0**	**72.9**
	50	85.1	82.9	84.0	59.8	52.6	55.9	70.0
DREC	15	79.7	84.1	81.8	52.5	55.3	53.8	67.8
	25	78.9	83.9	**81.3**	50.0	54.7	**52.2**	**66.8**
	50	78.0	84.8	81.3	51.0	54.2	52.6	66.9
ADE	15	84.8	88.0	86.3	72.7	77.5	75.0	80.7
	25	84.7	88.1	**86.4**	72.1	77.2	**74.5**	**80.4**
	50	84.6	88.0	86.3	72.1	77.4	74.7	80.5

Table 4A.9: Model performance for different label embeddings size values. Bold entries indicate the result reported in Section 4A.5.

	Label Embeddings	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	15	80.9	81.2	81.1	49.2	43.8	46.3	63.7
	25	81.0	81.3	**81.1**	50.1	44.4	**47.1**	**64.1**
	50	81.1	81.6	81.3	48.0	44.4	46.1	63.7
CoNLL 04	15	84.6	83.5	84.0	62.2	56.1	59.0	71.5
	0	83.7	84.0	**83.9**	63.7	60.4	**62.0**	**72.9**
	50	82.3	84.1	83.2	59.3	55.9	57.5	70.3
DREC	15	78.4	84.8	81.5	51.8	53.2	52.5	67.0
	25	78.9	83.9	**81.3**	50.0	54.7	**52.2**	**66.8**
	50	78.9	84.8	81.7	51.3	53.2	52.2	67.0
ADE	15	84.4	88.1	86.2	71.9	77.4	74.6	80.4
	25	84.7	88.1	**86.4**	72.1	77.2	**74.5**	**80.4**
	50	84.8	88.6	86.6	72.4	78.6	75.4	81.0

Table 4A.10: Model performance for different layer widths l of the neural network (both for the entity and the relation scoring layers). Bold entries indicate the result reported in Section 4A.5.

	Hidden layer Size	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
ACE 04	32	81.0	81.0	81.0	48.8	43.2	45.8	63.4
	64	81.0	81.3	**81.1**	50.1	44.4	**47.1**	**64.1**
	128	81.3	81.3	81.3	51.5	43.6	47.3	64.3
CoNLL 04	32	82.2	84.2	83.2	65.9	59.2	62.4	72.8
	64	83.7	84.0	**83.9**	63.7	60.4	**62.0**	**72.9**
	128	82.6	83.6	83.1	64.4	55.4	59.6	71.4
DREC	32	79.6	84.2	81.8	52.4	51.4	51.9	66.9
	64	78.9	83.9	**81.3**	50.0	54.7	**52.2**	**66.8**
	128	78.3	84.4	81.3	48.5	53.0	50.7	66.0
ADE	32	84.3	88.5	86.3	71.6	78.5	74.9	80.6
	64	84.7	88.1	**86.4**	72.1	77.2	**74.5**	**80.4**
	128	84.8	88.5	86.6	72.2	78.2	75.1	80.8

Table 4A.11: Model performance for different embeddings on the ACE04 dataset. Bold entries indicate the result reported in Section 4A.5.

Embeddings	Size	Entity			Relation			Overall F_1
		P	R	F_1	P	R	F_1	
Miwa & Bansal (2016) [5]	200	81.0	81.3	**81.1**	50.1	44.4	**47.1**	**64.1**
Adel & Schütze [8]	50	82.1	79.8	80.9	49.1	41.4	44.9	62.9
Li et al. (2017) [6]	200	81.5	81.3	81.4	46.5	44.4	45.4	63.4

In the main results (see Section 4A.5), to guarantee a fair comparison to previous work and to obtain comparable results that are not affected by the input embeddings, we use embeddings used also in prior studies. To assess the performance of our system to input variations, we also report results using different word embeddings (see Table 4A.11) (i.e., [6, 8]) on the ACE04 dataset. Our results showcase that our model, even when using different word embeddings, is still performing better compared to other works that, like ours, do not rely on additional NLP tools. The main goal of this appendix was to demonstrate that even with different hyperparameter values, different embeddings, the performance of our model does not vary a lot (i.e., $\sim 1\%$ F_1 scoring points depending on the size of each dataset).

References

[1] D. Nadeau and S. Sekine. *A survey of named entity recognition and classification*. Lingvisticae Investigationes, 30(1):3–26, 2007. doi:10.1075/li.30.1.03nad.

[2] N. Bach and S. Badaskar. *A review of relation extraction*. Literature review for Language and Statistics II, 2007.

[3] Q. Li and H. Ji. *Incremental Joint Extraction of Entity Mentions and Relations*. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 402–412, Baltimore, USA, 23–25 Jun. 2014.

[4] M. Miwa and Y. Sasaki. *Modeling Joint Entity and Relation Extraction with Table Representation*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pages 1858–1869, Doha, Qatar, 25–29 Oct. 2014. Association for Computational Linguistics.

[5] M. Miwa and M. Bansal. *End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1105–1116, Berlin, Germany, 7–12 Aug. 2016.

[6] F. Li, M. Zhang, G. Fu, and D. Ji. *A neural joint model for entity and relation extraction from biomedical text*. BMC Bioinformatics, 18(1):1–11, 2017. doi:10.1186/s12859-017-1609-9.

[7] P. Gupta, H. Schütze, and B. Andrassy. *Table filling multi-task recurrent neural network for joint entity and relation extraction*. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2537–2547, 2016.

[8] H. Adel and H. Schütze. *Global Normalization of Convolutional Neural Networks for Joint Entity and Relation Classification*. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[9] A. Katiyar and C. Cardie. *Going out on a limb: Joint Extraction of Entity Mentions and Relations without Dependency Trees*. In Proceedings of the 55st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, Canada, 2017.

[10] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *An attentive neural architecture for joint segmentation and parsing and its application to real estate ads*. Expert Systems with Applications, 102:100–112, 2018. doi:10.1016/j.eswa.2018.02.031.

[11] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Reconstructing the house from the ad: Structured prediction on real estate classifieds*. In Proceedings of the 15th Conference of the European Chapter of the

Association for Computational Linguistics: (Volume 2, Short Papers), pages 274–279, Valencia, Spain, 3–7 Apr. 2017.

[12] K. Fundel, R. Küffner, and R. Zimmer. *RelEx-Relation extraction using dependency parse trees*. Bioinformatics, 23(3):365–371, 2007. doi:10.1093/bioinformatics/btl616.

[13] H. Gurulingappa, A. Mateen-Rajpu, and L. Toldo. *Extraction of potential adverse drug events from medical case reports*. Journal of Biomedical Semantics, 3(1):1–15, 2012. doi:10.1186/2041-1480-3-15.

[14] J. Lafferty, A. McCallum, and F. Pereira. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. In Proceedings of the 18th International Conference on Machine Learning, pages 282–289, San Francisco, USA, 28 Jun.–1 Jul. 2001. Morgan Kaufmann.

[15] B. Taskar, C. Guestrin, and D. Koller. *Max-Margin Markov networks*. In Proceedings of the 16th International Conference on Neural Information Processing Systems, pages 25–32. MIT Press, Bangkok, Thailand, 1–5 Dec. 2003.

[16] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. *Support Vector Machine Learning for Interdependent and Structured Output Spaces*. In Proceedings of the 21st International Conference on Machine Learning, pages 104–112, Helsinki, Finland, 5–9 Jul. 2004. ACM. doi:10.1145/1015330.1015341.

[17] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. *Natural Language Processing (Almost) from Scratch*. Journal of Machine Learning Research, 12:2493–2537, November 2011.

[18] Z. Huang, W. Xu, and K. Yu. *Bidirectional LSTM-CRF models for sequence tagging*. arXiv preprint arXiv:1508.01991, 2015.

[19] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. *Neural Architectures for Named Entity Recognition*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270, San Diego, California, 12–17 Jun. 2016.

[20] X. Ma and E. Hovy. *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1064–1074, Berlin, Germany, 7–12 Aug. 2016.

[21] D. Zelenko, C. Aone, and A. Richardella. *Kernel Methods for Relation Extraction*. Journal of Machine Learning Research, 3:1083–1106, 2003. doi:10.3115/1118693.1118703.

[22] N. Kambhatla. *Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations*. In Proceedings of the Annual Meeting of the Association for Computational Linguistics on Interactive poster and demonstration sessions, Barcelona, Spain, 2004. doi:10.3115/1219044.1219066.

[23] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. *Semantic compositionality through recursive matrix-vector spaces*. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211, Jeju Island, Korea, 12–14 Jul. 2012. Association for Computational Linguistics.

[24] D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao. *Relation classification via convolutional deep neural network*. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 2335–2344, 2014.

[25] A. Culotta and J. Sorensen. *Dependency tree kernels for relation extraction*. In Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, pages 423–429, Barcelona, Spain, 2004. doi:10.3115/1218955.1219009.

[26] B. Rink and S. Harabagiu. *Utd: Classifying semantic relations by combining lexical and semantic resources*. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 256–259, Los Angeles, California, 2010. Association for Computational Linguistics.

[27] K. Xu, Y. Feng, S. Huang, and D. Zhao. *Semantic Relation Classification via Convolutional Neural Networks with Simple Negative Sampling*. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 536–540, Lisbon, Portugal, September 2015. Association for Computational Linguistics. Available from: http://aclweb.org/anthology/D15-1062.

[28] C. dos Santos, B. Xiang, and B. Zhou. *Classifying Relations by Ranking with Convolutional Neural Networks*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 626–634, Beijing, China, 26–31 Jul. 2015.

[29] R. Socher, D. Chen, C. D. Manning, and A. Ng. *Reasoning With Neural Tensor Networks for Knowledge Base Completion*. In Proceedings of the 26th International Conference on Neural Information Processing Systems, pages 926–934, Nevada, United States, 5–10 Dec. 2013. Curran Associates, Inc.

[30] D. Zhang and D. Wang. *Relation classification via recurrent neural network*. arXiv preprint arXiv:1508.01006, 2015.

[31] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin. *Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths*. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1785–1794, Lisbon, Portugal, 17–21 Sept. 2015. Association for Computational Linguistics.

[32] N. T. Vu, H. Adel, P. Gupta, and H. Schütze. *Combining Recurrent and Convolutional Neural Networks for Relation Classification*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 534–539, San Diego, California, June 2016. Available from: http://www.aclweb.org/anthology/N16-1065.

[33] R. J. Kate and R. Mooney. *Joint Entity and Relation Extraction Using Card-Pyramid Parsing*. In Proceedings of the 14th Conference on Computational Natural Language Learning, pages 203–212, Uppsala, Sweden, 15–16 Jul. 2010. Association for Computational Linguistics.

[34] B. Yang and C. Cardie. *Joint Inference for Fine-grained Opinion Extraction*. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1640–1649, Sofia, Bulgaria, August 2013. Available from: http://www.aclweb.org/anthology/P13-1161.

[35] S. Zheng, Y. Hao, D. Lu, H. Bao, J. Xu, H. Hao, and B. Xu. *Joint entity and relation extraction based on a hybrid neural network*. Neurocomputing, 257:59–66, 2017. doi:10.1016/j.neucom.2016.12.075.

[36] X. Zhang, J. Cheng, and M. Lapata. *Dependency Parsing as Head Selection*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 1, Long Papers), pages 665–676, Valencia, Spain, 3–7 Apr. 2017.

[37] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. *Distributed Representations of Words and Phrases and their Compositionality*.

In Proceedings of the 26th International Conference on Neural Information Processing Systems, pages 3111–3119, Nevada, United States, 5–10 Dec. 2013. Curran Associates, Inc.

[38] I. Sutskever, O. Vinyals, and Q. V. Le. *Sequence to Sequence Learning with Neural Networks*. In Proceedings of the 27th International Conference on Neural Information Processing Systems, pages 3104–3112, Montreal, Canada, 08–13 Dec. 2014. MIT Press.

[39] Y. Bengio, P. Simard, and P. Frasconi. *Learning Long-term Dependencies with Gradient Descent is Difficult*. Transactions on neural networks, 5(2):157–166, 1994. doi:10.1109/72.279181.

[40] R. Pascanu, T. Mikolov, and Y. Bengio. *On the Difficulty of Training Recurrent Neural Networks*. In Proceedings of the 30th International Conference on International Conference on Machine Learning, pages 1310–1318, Atlanta, USA, 16–21 Jun. 2013. JMLR.org.

[41] Y.-J. Chu and T.-H. Liu. *On shortest arborescence of a directed graph*. Scientia Sinica, 14:1396–1400, 1965.

[42] J. Edmonds. *Optimum branchings*. Journal of research of the National Bureau of Standards, 71B(4):233–240, 1967.

[43] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel. *The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation*. In Proceedings Fourth International Conference on Language Resources and Evaluation, volume 2, page 1, Lisbon, Portugal, 2004.

[44] H. Gurulingappa, A. M. Rajput, A. Roberts, J. Fluck, M. Hofmann-Apitius, and L. Toldo. *Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports*. Journal of Biomedical Informatics, 45(5):885–892, 2012. doi:10.1016/j.jbi.2012.04.008.

[45] D. Roth and W.-t. Yih. *A Linear Programming Formulation for Global Inference in Natural Language Tasks*. In HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004), pages 1–8, Boston, USA, 2004. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/W04-2401.

[46] F. Li, Y. Zhang, M. Zhang, and D. Ji. *Joint Models for Extracting Adverse Drug Events from Biomedical Text*. In Proceedings of the Twenty-Fifth

International Joint Conference on Artificial Intelligence, pages 2838–2844, New York, USA, 9–15 Jul. 2016. IJCAI/AAAI Press.

[47] S. Moen and T. S. S. Ananiadou. *Distributional semantics resources for biomedical text processing*. In Proceedings of the 5th International Symposium on Languages in Biology and Medicine, pages 39–43, Tokyo, Japan, 2013.

[48] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. *TensorFlow: A System for Large-scale Machine Learning*. In Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, pages 265–283, Berkeley, CA, USA, 2016.

[49] D. Kingma and J. Ba. *Adam: A method for stochastic optimization*. In International Conference on Learning Representations, San Diego, USA, 2015.

[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

4B

Adversarial training for multi-context joint entity and relation extraction

In this chapter, we explain how adversarial perturbations can be applied on top of our joint model (described in Chapter 4A) to improve the performance of the named entity recognition and relation extraction tasks. The core contribution of this chapter is the use of adversarial perturbations as an extension in the training procedure for the joint extraction task. Experimental results illustrate that adversarial perturbations improve the performance of the joint model, not only (i) in terms of overall performance, but also (ii) in terms of performance for each task separately, and (iii) even from the first training epochs onwards.

⋆⋆⋆

G. Bekoulis, J. Deleu, T. Demeester and C. Develder

In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018.

Abstract Adversarial training (AT) is a regularization method that can be used to improve the robustness of neural network methods by adding

small perturbations in the training data. We show how to use AT for the tasks of entity recognition and relation extraction. In particular, we demonstrate that applying AT to a general purpose baseline model for jointly extracting entities and relations, allows improving the state-of-the-art effectiveness on several datasets in different contexts (i.e., news, biomedical, and real estate data) and for different languages (English and Dutch).

4B.1 Introduction

Many neural network methods have recently been exploited in various natural language processing (NLP) tasks, such as parsing [1], POS tagging [2], relation extraction [3], translation [4], and joint tasks [5]. However, Szegedy et al. [6] observed that intentional small scale perturbations (i.e., adversarial examples) to the input of such models may lead to incorrect decisions (with high confidence). Goodfellow et al. [7] proposed adversarial training (AT) (for image recognition) as a regularization method which uses a mixture of clean and adversarial examples to enhance the robustness of the model. Although AT has recently been applied in NLP tasks (e.g., text classification [8]), this paper — to the best of our knowledge — is the first attempt investigating regularization effects of AT in a joint setting for two related tasks.

We start from a baseline joint model that performs the tasks of named entity recognition (NER) and relation extraction at once. Previously proposed models (summarized in Section 4B.2) exhibit several issues that the neural network-based baseline approach (detailed in Section 4B.3.1) overcomes: (i) our model uses automatically extracted features without the need of external parsers nor manually extracted features (see [5, 9, 10]), (ii) all entities and the corresponding relations within the sentence are extracted at once, instead of examining one pair of entities at a time (see [11]), and (iii) we model relation extraction in a multi-label setting, allowing multiple relations per entity (see [12, 13]). The core contribution of the paper is the use of AT as an extension in the training procedure for the joint extraction task (Section 4B.3.2).

To evaluate the proposed AT method, we perform a large scale experimental study in this joint task (see Section 4B.4), using datasets from different contexts (i.e., news, biomedical, real estate) and languages (i.e., English, Dutch). We use a strong baseline that outperforms all previous models that rely on automatically extracted features, achieving state-of-the-art performance (Section 4B.5). Compared to the baseline model, applying AT during training leads to a consistent additional increase in joint extraction effectiveness.

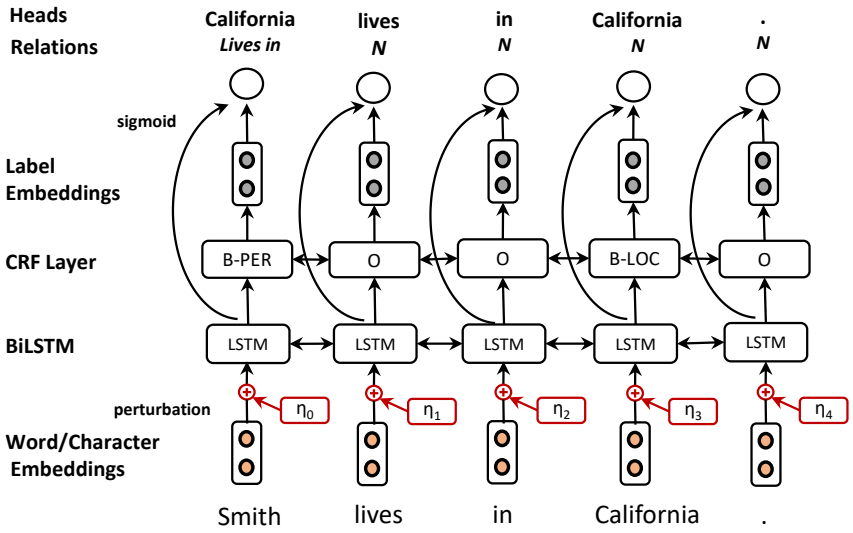

Figure 4B.1: Our model for joint entity and relation extraction with adversarial training (AT) comprises (i) a word and character embedding layer, (ii) a BiLSTM layer, (iii) a CRF layer and (iv) a relation extraction layer. In AT, we compute the worst-case perturbations η of the input embeddings.

4B.2 Related work

Joint entity and relation extraction: Joint models [14, 15] that are based on manually extracted features have been proposed for performing both the named entity recognition (NER) and relation extraction subtasks at once. These methods rely on the availability of NLP tools (e.g., POS taggers) or manually designed features leading to additional complexity. Neural network methods have been exploited to overcome this feature design issue and usually involve RNNs and CNNs [5, 16]. Specifically, Miwa and Bansal [5] as well as Li et al. [10] apply bidirectional tree-structured RNNs for different contexts (i.e., news, biomedical) to capture syntactic information (using external dependency parsers). Gupta et al. [9] propose the use of various manually extracted features along with RNNs. Adel and Schütze [11] solve the simpler problem of entity classification (EC, assuming entity boundaries are given), instead of NER, and they replicate the context around the entities, feeding entity pairs to the relation extraction layer. Katiyar and Cardie [12] investigate RNNs with attention without taking into account that relation labels are not mutually exclusive. Finally, Bekoulis et al. [13] use LSTMs in a joint model for extracting just one rela-

tion at a time, but increase the complexity of the NER part. Our baseline model enables simultaneous extraction of multiple relations from the same input. Then, we further extend this strong baseline using adversarial training.

Adversarial training (AT) [7] has been proposed to make classifiers more robust to input perturbations in the context of image recognition. In the context of NLP, several variants have been proposed for different tasks such as text classification [8], relation extraction [17] and POS tagging [18]. AT is considered as a regularization method. Unlike other regularization methods (i.e., dropout [19], word dropout [20]) that introduce random noise, AT generates perturbations that are variations of examples easily misclassified by the model.

4B.3 Model

4B.3.1 Joint learning as head selection

The baseline model, described in detail in [21], is illustrated in Fig. 4B.1. It aims to detect (i) the type and the boundaries of the entities and (ii) the relations between them. The input is a sequence of tokens (i.e., sentence) $w = w_1, ..., w_n$. We use character level embeddings to implicitly capture morphological features (e.g., prefixes and suffixes), representing each character by a vector (embedding). The character embeddings are fed to a bidirectional LSTM (BiLSTM) to obtain the character-based representation of the word. We also use pre-trained word embeddings. Word and character embeddings are concatenated to form the final token representation, which is then fed to a BiLSTM layer to extract sequential information.

For the **NER task**, we adopt the BIO (Beginning, Inside, Outside) encoding scheme. In Fig. 4B.1, the B-*PER* tag is assigned to the beginning token of a 'person' (*PER*) entity. For the prediction of the entity tags, we use: (i) a softmax approach for the entity classification (EC) task (assuming entity boundaries given) or (ii) a CRF approach where we identify both the *type* and the boundaries for each entity. During decoding, in the softmax setting, we greedily detect the entity *types* of the tokens (i.e., independent prediction). Although independent distribution of *types* is reasonable for EC tasks, this is not the case when there are strong correlations between neighboring tags. For instance, the BIO encoding scheme imposes several constraints in the NER task (e.g., the B-*PER* and I-*LOC* tags cannot be sequential). Motivated by this intuition, we use a linear-chain CRF for the NER task [2]. For decoding, in the CRF setting, we use the Viterbi algorithm. During training, for both EC (softmax) and NER tasks (CRF), we

minimize the cross-entropy loss $\mathcal{L}_{\text{NER}}$. The entity tags are later fed into the relation extraction layer as label embeddings (see Fig. 4B.1), assuming that knowledge of the entity *types* is beneficial in predicting the relations between the involved entities.

We model the **relation extraction task** as a multi-label head selection problem [1, 21]. In our model, each word w_i can be involved in multiple relations with other words. For instance, in the example illustrated in Fig. 4B.1, "Smith" could be involved not only in a *Lives in* relation with the token "California" (head) but also in other relations simultaneously (e.g., *Works for, Born In* with some corresponding tokens). The goal of the task is to predict for each word w_i, a vector of heads $\hat{y}_i$ and the vector of corresponding relations $\hat{r}_i$. We compute the score $s(w_j, w_i, r_k)$ of word w_j to be the head of w_i given a relation label r_k using a single layer neural network. The corresponding probability is defined as: $\Pr(w_j, r_k \mid w_i; \theta) = \sigma(s(w_j, w_i, r_k))$, where $\sigma(.)$ is the sigmoid function. During training, we minimize the cross-entropy loss $\mathcal{L}_{\text{rel}}$ as:

$$\sum_{i=0}^{n} \sum_{j=0}^{m} -\log \Pr(y_{i,j},\, r_{i,j} \mid w_i; \theta) \tag{4B.1}$$

where m is the number of associated heads (and thus relations) per word w_i. During decoding, the most probable heads and relations are selected using threshold-based prediction. The final objective for the joint task is computed as $\mathcal{L}_{\text{JOINT}}(w; \theta) = \mathcal{L}_{\text{NER}} + \mathcal{L}_{\text{rel}}$ where θ is a set of parameters. In the case of multi-token entities, only the last token of the entity can serve as head of another token, to eliminate redundant relations. If an entity is not involved in any relation, we predict the auxiliary "N" relation label and the token itself as head. An additional batch normalization layer could have been used, however we do not include it since it has not been used in previous studies (see [2, 5, 22]).

4B.3.2 Adversarial training (AT)

We exploit the idea of AT [7] as a regularization method to make our model robust to input perturbations. Specifically, we generate examples which are variations of the original ones by adding some noise at the level of the concatenated word representation [8]. This is similar to the concept introduced by [7] to improve the robustness of image recognition classifiers. We generate an adversarial example by adding the worst-case perturbation η_{adv} to the original embedding w that maximizes the loss function:

$$\eta_{adv} = \underset{\|\eta\| \leq \epsilon}{\text{argmax}}\, \mathcal{L}_{\text{JOINT}}(w + \eta; \hat{\theta}) \tag{4B.2}$$

where $\hat{\theta}$ is a copy of the current model parameters. Since Eq. (4B.2) is intractable in neural networks, we use the approximation proposed in [7] defined as: $\eta_{adv} = \epsilon g / \|g\|$, with $g = \nabla_w \mathcal{L}_{\text{JOINT}}(w; \hat{\theta})$, where ϵ is a small bounded norm treated as a hyperparameter. Similar to [18], we set ϵ to be $\alpha\sqrt{D}$ (where D is the dimension of the embeddings). We train on the mixture of original and adversarial examples, so the final loss is computed as: $\mathcal{L}_{\text{JOINT}}(w; \hat{\theta}) + \mathcal{L}_{\text{JOINT}}(w + \eta_{adv}; \hat{\theta})$.

4B.4 Experimental setup

We evaluate our models on four datasets, using the code as available from our Github codebase.[1] Specifically, we follow the 5-fold cross-validation defined by [5] for the ACE04 [23] dataset. For the CoNLL04 [24] EC task (assuming boundaries are given), we use the same splits as in [9, 11]. We also evaluate our models on the NER task similar to [15] in the same dataset using 10-fold cross validation. For the Dutch Real Estate Classifieds, DREC [25] dataset, we use train-test splits as in [13]. For the Adverse Drug Events, ADE [26], we perform 10-fold cross-validation similar to [10]. To obtain comparable results that are not affected by the input embeddings, we use the embeddings of the previous works. We employ early stopping in all of the experiments. We use the Adam optimizer [27] and we fix the hyperparameters (i.e., α, dropout values, best epoch, learning rate) on the validation sets. The scaling parameter α is selected from $\{5 \times 10^{-2}, 1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}\}$. Larger values of α (i.e., larger perturbations) lead to consistent performance decrease in our early experiments. This can be explained from the fact that adding more noise can change the content of the sentence as also reported by [17].

We use three types of evaluation, namely: (i) *S(trict)*: we score an entity as correct if both the entity boundaries and the entity *type* are correct (ACE04, ADE, CoNLL04, DREC), (ii) *B(oundaries)*: we score an entity as correct if only the entity boundaries are correct while the entity *type* is not taken into account (DREC) and (iii) *R(elaxed)*: a multi-token entity is considered correct if at least one correct *type* is assigned to the tokens comprising the entity, assuming that the boundaries are known (CoNLL04), to compare to previous works. In all cases, a relation is considered as correct when both the relation *type* and the argument entities are correct.

[1] https://github.com/bekou/multihead_joint_entity_relation_extraction

Table 4B.1: Comparison of our method with the state-of-the-art in terms of F_1 score. The proposed models are: (i) baseline, (ii) baseline EC (predicts only entity classes) and (iii) baseline (EC) + AT (regularized by AT). The ✓and ✗ symbols indicate whether the models rely on external NLP tools. We include different evaluation types (*S*, *R* and *B*). Note that confidence intervals for each dataset are reported in Fig. 4B.2.

	Settings	Features	Eval.	Entity	Relation	Overall
ACE 04	Miwa & Bansal (2016) [5]	✓	*S*	81.8	48.4	65.1
	Katiyar and Cardie (2017) [12]	✗	*S*	79.6	45.7	62.6
	baseline	✗	*S*	81.1	47.1	64.1
	baseline + AT	✗	*S*	**81.6**	**47.4**	**64.5**
CoNLL 04	Gupta et al. (2016) [9]	✓	*R*	92.4	69.9	81.1
	Gupta et al. (2016) [9]	✗	*R*	88.8	58.3	73.6
	Adel & Schütze [11]	✗	*R*	82.1	62.5	72.3
	baseline EC	✗	*R*	**93.2**	67.0	80.1
	baseline EC + AT	✗	*R*	93.0	**67.9**	**80.5**
	Miwa & Sasaki (2014) [15]	✓	*S*	80.7	61.0	70.8
	baseline	✗	*S*	83.0	61.0	72.0
	baseline + AT	✗	*S*	**83.6**	**61.9**	**72.7**
DREC	Bekoulis et al. (2018a) [13]	✗	*B*	79.1	49.7	64.4
	baseline	✗	*B*	82.3	52.8	67.5
	baseline + AT	✗	*B*	**82.9**	**53.8**	**68.4**
	baseline	✗	*S*	81.3	52.2	66.8
	baseline + AT	✗	*S*	**82.0**	**53.1**	**67.5**
ADE	Li et al. (2016) [28]	✓	*S*	79.5	63.4	71.4
	Li et al. (2017) [10]	✓	*S*	84.6	71.4	78.00
	baseline	✗	*S*	86.4	74.5	80.4
	baseline + AT	✗	*S*	**86.7**	**75.5**	**81.1**

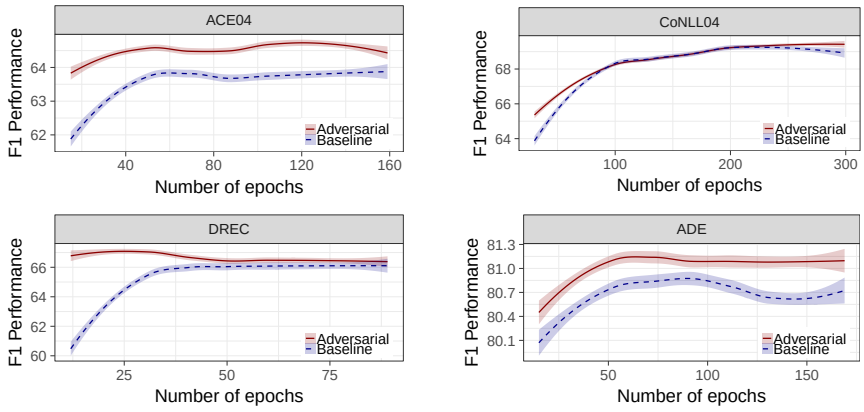

Figure 4B.2: F_1 performance of the baseline and the AT models on the validation sets from 10-30 epochs onwards depending on the dataset. The smoothed lines (obtained by LOWESS smoothing) model the trends and the 95% confidence intervals.

4B.5 Results

Table 4B.1 shows our experimental results. The name of the dataset is presented in the first column while the models are listed in the second column. The proposed models are the following: (i) *baseline*: the baseline model shown in Fig. 4B.1 with the CRF layer and the sigmoid loss, (ii) *baseline EC*: the proposed model with the softmax layer for EC, (iii) *baseline (EC) + AT*: the baseline regularized using AT. The final three columns present the F_1 results for the two subtasks and their average performance. Bold values indicate the best results among models that use only automatically extracted features.

For ACE04, the baseline outperforms [12] by $\sim$2% in both tasks. This improvement can be explained by the use of: (i) multi-label head selection, (ii) CRF-layer and (iii) character level embeddings. Compared to Miwa and Bansal [5], who rely on NLP tools, the baseline performs within a reasonable margin (less than 1%) on the joint task. On the other hand, Li et al. [10] use the same model for the ADE biomedical dataset, where we report a 2.5% overall improvement. This indicates that NLP tools are not always accurate for various contexts. For the CoNLL04 dataset, we use two evaluation settings. We use the *relaxed* evaluation similar to [9, 11] on the EC task. The baseline model outperforms the state-of-the-art models that do not rely on manually extracted features ($>$4% improvement for both tasks), since we directly model the whole sentence, instead of just considering pairs of entities. Moreover, compared to the model of [9] that relies on complex features, the baseline model performs within a margin of 1% in terms of overall F_1 score. We also report NER results on the same dataset and improve overall F_1 score with $\sim$1% compared to [15], indicating that our automatically extracted features are more informative than the handcrafted ones. These automatically extracted features exhibit their performance improvement mainly due to the shared LSTM layer that learns to automatically generate feature representations of entities and their corresponding relations within a single model. For the DREC dataset, we use two evaluation methods. In the *boundaries* evaluation, the baseline has an improvement of $\sim$3% on both tasks compared to [13], whose quadratic scoring layer complicates NER.

Table 4B.1 and Fig. 4B.2 show the effectiveness of the adversarial training on top of the baseline model. In all of the experiments, AT improves the predictive performance of the baseline model in the joint setting. Moreover, as seen in Fig. 4B.2, the performance of the models using AT is closer to maximum even from the early training epochs. Specifically, for ACE04, there is an improvement in both tasks as well as in the overall F_1 perfor-

mance (0.4%). For CoNLL04, we note an improvement in the overall F_1 of 0.4% for the EC and 0.8% for the NER tasks, respectively. For the DREC dataset, in both settings, there is an overall improvement of $\sim$1%. Figure 4B.2 shows that from the first epochs, the model obtains its maximum performance on the DREC validation set. Finally, for ADE, our AT model beats the baseline F_1 by 0.7%.

Our results demonstrate that AT outperforms the neural baseline model consistently, considering our experiments across multiple and more diverse datasets than typical related works. The improvement of AT over our baseline (depending on the dataset) ranges from $\sim$0.4% to $\sim$0.9% in terms of overall F_1 score. This seemingly small performance increase is mainly due to the limited performance benefit for the NER component, which is in accordance with the recent advances in NER using neural networks that report similarly small gains (e.g., the performance improvement in [22] and [2] on the CoNLL-2003 test set is 0.01% and 0.17% F_1 percentage points, while in the work of [18], a 0.07% F_1 improvement on CoNLL-2000 using AT for NER is reported). However, the relation extraction performance increases by $\sim$1% F_1 scoring points, except for the ACE04 dataset. Further, as seen in Fig. 4B.2, the improvement for CoNLL04 is particularly small on the evaluation set. This may indicate a correlation between the dataset size and the benefit of adversarial training in the context of joint models, but this needs further investigation in future work. Thus, we hypothesize that in a smaller dataset, less adversarial examples are being generated compared to the adversarial examples generated for the rest of datasets leading to a smaller increase in the F_1 score.

4B.6 Conclusion

We proposed to use adversarial training (AT) for the joint task of entity recognition and relation extraction. The contribution of this study is twofold: (i) investigation of the consistent effectiveness of AT as a regularization method over a multi-context baseline joint model, with (ii) a large scale experimental evaluation. Experiments show that AT improves the results for each task separately, as well as the overall performance of the baseline joint model, while reaching high performance already during the first epochs of the training procedure.

Acknowledgments

We would like to thank the anonymous reviewers for the time and effort they spent in reviewing our work, and for their valuable feedback.

References

[1] X. Zhang, J. Cheng, and M. Lapata. *Dependency Parsing as Head Selection*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 1, Long Papers), pages 665–676, Valencia, Spain, 3–7 Apr. 2017. doi:10.18653/v1/e17-1063.

[2] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. *Neural Architectures for Named Entity Recognition*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270, San Diego, California, 12–17 Jun. 2016. doi:10.18653/v1/n16-1030.

[3] C. dos Santos, B. Xiang, and B. Zhou. *Classifying Relations by Ranking with Convolutional Neural Networks*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 626–634, Beijing, China, 26–31 Jul. 2015. doi:10.3115/v1/p15-1061.

[4] D. Bahdanau, K. Cho, and Y. Bengio. *Neural machine translation by jointly learning to align and translate*. In Proceedings of the International Conference for Learning Representations, San Diego, USA, 7–9 May 2015.

[5] M. Miwa and M. Bansal. *End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1105–1116, Berlin, Germany, 7–12 Aug. 2016. doi:10.18653/v1/p16-1105.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. *Intriguing properties of neural networks*. In Proceedings of the International Conference on Learning Representations, Banff, Canada, 14–16 Apr. 2014.

[7] I. Goodfellow, J. Shlens, and C. Szegedy. *Explaining and Harnessing Adversarial Examples*. In Proceedings of the International Conference on Learning Representations, San Diego, USA, 7–9 May 2015. Available from: http://arxiv.org/abs/1412.6572.

[8] T. Miyato, A. M. Dai, and I. Goodfellow. *Adversarial training methods for semi-supervised text classification*. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 Apr. 2017.

[9] P. Gupta, H. Schütze, and B. Andrassy. *Table filling multi-task recurrent neural network for joint entity and relation extraction*. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2537–2547, 2016.

[10] F. Li, M. Zhang, G. Fu, and D. Ji. *A neural joint model for entity and relation extraction from biomedical text*. BMC Bioinformatics, 18(1):1–11, 2017. doi:10.1186/s12859-017-1609-9.

[11] H. Adel and H. Schütze. *Global Normalization of Convolutional Neural Networks for Joint Entity and Relation Classification*. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi:10.18653/v1/d17-1181.

[12] A. Katiyar and C. Cardie. *Going out on a limb: Joint Extraction of Entity Mentions and Relations without Dependency Trees*. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, Canada, 2017. doi:10.18653/v1/p17-1085.

[13] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *An attentive neural architecture for joint segmentation and parsing and its application to real estate ads*. Expert Systems with Applications, 102:100–112, 2018. doi:10.1016/j.eswa.2018.02.031.

[14] Q. Li and H. Ji. *Incremental Joint Extraction of Entity Mentions and Relations*. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 402–412, Baltimore, USA, 23–25 Jun. 2014. doi:10.3115/v1/p14-1038.

[15] M. Miwa and Y. Sasaki. *Modeling Joint Entity and Relation Extraction with Table Representation*. In Proceedings of the 2014 Conference

on Empirical Methods in Natural Language Processing, pages 1858–1869, Doha, Qatar, 25–29 Oct. 2014. Association for Computational Linguistics. doi:10.3115/v1/d14-1200.

[16] S. Zheng, Y. Hao, D. Lu, H. Bao, J. Xu, H. Hao, and B. Xu. *Joint entity and relation extraction based on a hybrid neural network*. Neurocomputing, 257:59–66, 2017. doi:10.1016/j.neucom.2016.12.075.

[17] Y. Wu, D. Bamman, and S. Russell. *Adversarial Training for Relation Extraction*. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1778–1783, Copenhagen, Denmark, 2017. Association for Computational Linguistics. Available from: http://aclweb.org/anthology/D17-1187, doi:10.18653/v1/d17-1187.

[18] M. Yasunaga, J. Kasai, and D. Radev. *Robust Multilingual Part-of-Speech Tagging via Adversarial Training*. In Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, USA, 2018. doi:10.18653/v1/n18-1089.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[20] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III. *Deep Unordered Composition Rivals Syntactic Methods for Text Classification*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1681–1691, Beijing, China, July 2015. Association for Computational Linguistics. doi:10.3115/v1/p15-1162.

[21] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Joint entity recognition and relation extraction as a multi-head selection problem*. Expert Systems with Applications, 114:34–45, 2018. Available from: http://www.sciencedirect.com/science/article/pii/S095741741830455X, doi:10.1016/j.eswa.2018.07.032.

[22] X. Ma and E. Hovy. *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1064–1074, Berlin, Germany, 7–12 Aug. 2016. doi:10.18653/v1/p16-1101.

[23] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. *The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation*. In Proceedings of the Fourth International Conference on Language Resources and Evaluation, volume 2, page 1, Lisbon, Portugal, 2004.

[24] D. Roth and W.-t. Yih. *A Linear Programming Formulation for Global Inference in Natural Language Tasks*. In HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004), pages 1–8, Boston, USA, 2004. Association for Computational Linguistics. Available from: http://www.aclweb.org/anthology/W04-2401.

[25] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Reconstructing the house from the ad: Structured prediction on real estate classifieds*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 2, Short Papers), pages 274–279, Valencia, Spain, 3–7 Apr. 2017. doi:10.18653/v1/e17-2044.

[26] H. Gurulingappa, A. M. Rajput, A. Roberts, J. Fluck, M. Hofmann-Apitius, and L. Toldo. *Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports*. Journal of Biomedical Informatics, 45(5):885–892, 2012. doi:10.1016/j.jbi.2012.04.008.

[27] D. Kingma and J. Ba. *Adam: A method for stochastic optimization*. In Proceedings of the International Conference on Learning Representations, San Diego, USA, 2015.

[28] F. Li, Y. Zhang, M. Zhang, and D. Ji. *Joint Models for Extracting Adverse Drug Events from Biomedical Text*. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pages 2838–2844, New York, USA, 9–15 Jul. 2016. IJCAI/AAAI Press.

5

Sub-event detection from Twitter streams as a sequence labeling problem

Previous chapters discussed our newly introduced task of real estate information extraction: Chapter 2 defined the extraction problem at hand, i.e., recover the tree-like structured representation of the property tree, and Chapters 3-4B proposed new neural network methods for the joint task of entity recognition and relation extraction. In this chapter however, we will focus on a considerably different problem: we present improved methods for sub-event detection in social media streams. We frame the task as a sequence labeling problem (which is inherently similar to the named entity recognition task described in Chapters 2-4B). This way, we are able to resolve several shortcomings identified in previous works (see Section 5.2). Specifically, our model is able to (i) take into account the chronological order of consecutive tweets while (ii) exploiting information from previous tweets for predicting the presence and the type of a sub-event. Experimental results indicate the benefit of sequence labeling for sub-event detection in sports Twitter streams in all of the examined architectures.

⋆⋆⋆

G. Bekoulis, J. Deleu, T. Demeester and C. Develder

In Proceedings of 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019.

Abstract This paper introduces improved methods for sub-event detection in social media streams, by applying neural sequence models not only on the level of individual posts, but also directly on the stream level. Current approaches to identify sub-events within a given event, such as a goal during a soccer match, essentially do not exploit the sequential nature of social media streams. We address this shortcoming by framing the sub-event detection problem in social media streams as a sequence labeling task and adopt a neural sequence architecture that explicitly accounts for the chronological order of posts. Specifically, we (i) establish a neural baseline that outperforms a graph-based state-of-the-art method for binary sub-event detection (2.7% micro-F_1 improvement), as well as (ii) demonstrate superiority of a recurrent neural network model on the posts sequence level for labeled sub-events (2.4% bin-level F_1 improvement over non-sequential models).

5.1 Introduction

Social media allow users to communicate via real-time postings and interactions, with Twitter as a notable example. Twitter user posts, i.e., tweets, are often related to events. These can be social events (concerts, research conferences, sports events, etc.), emergency situations (e.g., terrorist attacks) [1], etc. For a single event, multiple tweets are posted, by people with various personalities and social behavior. Hence, even more so than (typically more neutral) traditional media, this implies many different perspectives, offering an interesting aggregated description.

Given this continuous and large stream of (likely duplicated) information in Twitter streams, and their noisy nature, it is challenging to keep track of the main parts of an event, such as a soccer match. Automating such extraction of different sub-events within an evolving event is known as sub-event detection [2]. For tracking each of the sub-events, the timing aspect is an important concept (i.e., consecutive tweets in time). Thus, a sequential model could successfully exploit chronological relations between the tweets in a Twitter stream as an informative feature for sub-event detection.

Several methods have been proposed for sub-event detection: clustering methods [3], graph-based approaches [4], topic models [5] and neural network architectures [6]. None of these studies exploits the chronological relation between consecutive tweets. In contrast, our work does take

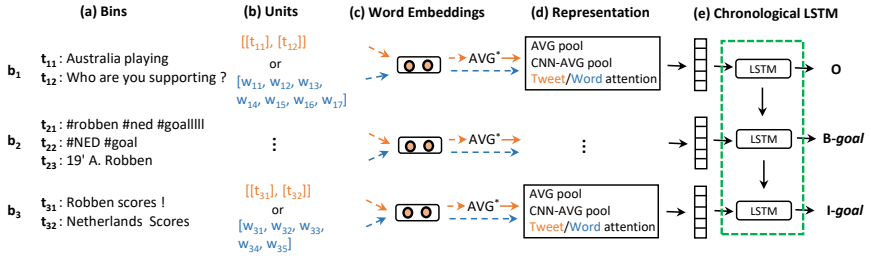

Figure 5.1: Our sub-event detection model comprises: (a) a bin layer, (b) a unit layer, (c) a word embeddings layer, (d) a representation layer and (e) a **chronological LSTM** layer to model the natural flow of the sub-events within the event. We represent each bin using either (i) a **tweet**- or (ii) a **word**-level representation. The AVG* represents an average pool operation, performed either directly on the embeddings or on the tweet's LSTM representation.

into account the chronological order of the Twitter stream and we predict the presence and the type of a sub-event exploiting information from previous tweets. Specifically, we (i) propose a new neural baseline model that outperforms the state-of-the-art performance on the binary classification problem of detecting the presence/absence of sub-events in a sports stream, (ii) establish a new reasonable baseline for also predicting the sub-event *types*, (iii) explicitly take into account chronological information, i.e., the relation among consecutive tweets, by framing sub-event detection as a sequence labeling problem on top of our baseline model, and (iv) perform an experimental study, indicating the benefit of sequence labeling for sub-event detection in sports Twitter streams.

5.2 Related work

Twitter streams have been extensively studied in various contexts, such as sentiment analysis [7], stock market prediction [8] and traffic detection [9]. Specifically, for sub-event detection in Twitter, several approaches have been tried. *Unsupervised methods* such as clustering aim to group similar tweets to detect specific sub-events [3, 10] and use simple representations such as tf-idf weighting combined with a similarity measure. Other unsupervised algorithms use topic modeling approaches, based on assumptions about the tweets' generation process [5, 11]. Several methods [2, 12, 13] assume that a sub-event happens when there is a 'burst', i.e., a sudden increase in the rate of tweets on the considered event, with many people commenting on it. Recently, neural network methods have used

more complicated representations [6, 14]. Also *supervised methods* have been applied [15, 16] to the sub-event detection task. These methods usually exploit graph-based structures or tf-idf weighting schemes. We believe to be the first to (i) exploit the chronological order of the Twitter stream in the context of sub-event detection and take into account its sequential nature, and (ii) frame the sub-event detection problem as a sequence labeling task.

5.3 Model

5.3.1 Task definition

The goal is, given a main event (i.e., soccer match), to identify its core sub-events (e.g., goals, kick-off, yellow cards) from Twitter streams. Specifically, we consider a *supervised* setting, relying on annotated data [16].

5.3.2 Word- vs tweet-level representations

Similar to previous works, we split a data stream into time periods [16]: we form bins of tweets posted during consecutive time intervals. E.g., for a soccer game, one-minute intervals (bins) lead to more than 90 bins, depending on the content before and after the game, halftime, stoppage time, and possibly some pre-game and post-game buffer. Thus, for each bin, we predict either the presence/absence of a sub-event (Section 5.3.3) or the most probable sub-event type (Section 5.3.4), depending on the evaluation scenario.

We consider representing the content of each bin either at (i) word-level or (ii) tweet-level (see Fig. 5.1). Formally, we assume that we have a set of n bins $b_1, ..., b_n$, where each bin b_i consists of m_i tweets and k_i words (i.e., all words of tweets in bin b_i). Then, the *tweet-level representation* of bin b_i is symbolized as $t_{i1}, ..., t_{im_i}$, where t_{im_i} is the m_i^{th} tweet of bin b_i. In the *word-level representation*, we chronologically concatenate the words from the tweets in the bin: $w_{i1}, .., w_{ik_i}$, where w_{ik_i} is the k_i^{th} word of bin b_i.

5.3.3 Binary classification baseline

To compare with previous work [16], we establish a simple baseline for binary classification: presence/absence of a sub-event. For this case, we use as input the word-level representation of each bin. To do so, we use word embeddings (randomly initialized) with average (AVG) pooling [17] in combination with a multilayer perceptron (MLP) for binary classification, i.e., presence/absence of a sub-event. Note that we experimented

with pre-trained embeddings as well as max-pooling, but those early experiments led to performance decrease compared to the presented baseline model. We found that training based on average bin representations works substantially better than with max-pooling, and we hypothesize that this is related to the noisy nature of the Twitter stream.

5.3.4 Sequence labeling approach

Building on the baseline above, we establish a new architecture that is able to capture the sub-event types as well as their duration. We phrase sub-event detection in Twitter streams as a sequence labeling problem. This means we assume that the label of a bin is not independent of neighboring bin labels, given the chronological order of bins of the Twitter stream, as opposed to independent prediction for each bin in the binary classification baseline above. For instance, when a *goal* is predicted as a label for bin b_i, then it is probable that the label of the next bin b_{i+1} will also be *goal*. Although a sub-event may occur instantly, an identified sub-event in a Twitter stream can span consecutive bins, i.e., minutes: users may continue tweeting on a particular sub-event for relatively long time intervals. For this reason, we apply the well-known BIO tagging scheme [18] for the sub-event detection problem. For example, the beginning of a *goal* sub-event is defined as B-*goal*, while I-*goal* (inside) is assigned to every consecutive bin within the same sub-event, and the O tag (outside) to every bin that is not part of any sub-event. To propagate chronological information among bins, we adopt an LSTM on the sequence of bins as illustrated in Fig. 5.1, layer (e). Note that this tagging approach assumes that sub-events do not overlap in time, i.e., only at most one is ongoing in the Twitter stream at any point in time.

5.4 Experimental setup

We evaluated our system[1] on the dataset from [16], with tweets on 20 soccer matches from the 2010 and 2014 FIFA World Cups, totalling over 2M pre-processed tweets filtered from 6.1M collected ones, comprising 185 events. The dataset includes a set of sub-events, such as *goal*, *kick-off*, *half-time*, etc. To compare our binary classification *baseline system* to previous methods (Table 5.1), we use the same train/test splits as [16], where 3 matches are used for training and 17 matches as test set. In this setting, we predict only the presence/absence of a sub-event. Similar to previous work, we count a sub-event as correct if at least one of its compris-

[1] https://github.com/bekou/subevent_sequence_labeling

ing bins has been classified as a sub-event. For the experimental study of our proposed *sequence labeling approach* for sub-event detection, where sub-event types are predicted, we have randomly split the test set into test (10 matches) and development (7 matches) sets. We use the development set to optimize the F_1 score for tuning of the model parameters, i.e., the word/tweet embedding representation size, LSTM hidden state size, dropout probability. We adopt 2 evaluation strategies. The first one, referred to as *relaxed* evaluation, is commonly used in entity classification tasks [19–21] and similar to the binary classification baseline system evaluation: score a multi-bin sub-event as correct if at least one of its comprising bin types (e.g., *goal*) is correct, assuming that the boundaries are given. The second evaluation strategy, *bin-level*, is stricter: we count each bin individually, and check whether its sub-event type has been predicted correctly, similar to the token-based evaluation followed in [22].

5.5 Results

5.5.1 Baseline results

Table 5.1 shows the experimental results of our baseline model. The Burst baseline system is based on the tweeting rate in a specific time window (i.e., bin) and if a threshold is exceed, the system identifies that a sub-event has occurred. We report evaluation scores as presented in [16]. The second approach is the graph-based method of [16]. We observe that our baseline system (Section 5.3.3) has a 1.2% improvement in terms of macro-F_1 and 2.7% improvement in terms of micro-F_1, compared to the graph-based model from [16], mainly due to increased precision, and despite the recall loss.

Table 5.1: Comparing our neural network binary classification baseline model to state-of-the-art (P = precision, R = recall).

Settings	Macro			Micro		
	P	R	F_1	P	R	F_1
Burst	78.0	54.0	64.0	72.0	54.0	62.0
Meladianos et al. (2018) [16]	76.0	75.0	75.0	73.0	74.0	73.00
Our binary classif. baseline	89.7	69.9	**76.1**	83.6	69.0	**75.6**

Table 5.2: Comparison of our baseline methods in terms of micro *bin-level* and *relaxed* F_1 score with and without chronological LSTM (see Fig. 5.1). The ✓and ✗ indicate whether the model uses a tweet-level LSTM (TL).

	Model	Bin-level				Relaxed			
		TL	P	R	F_1	TL	P	R	F_1
without chronol. LSTM	Word-tf-idf	-	49.4	52.0	50.6	-	56.1	56.1	56.1
	Word-AVG	-	51.4	45.9	48.5	-	56.1	56.1	56.1
	Word-CNN-AVG	-	56.9	56.0	56.4	-	75.6	75.6	75.6
	Word-attention	-	52.9	58.7	55.6	-	86.5	86.5	**86.5**
	Tweet-AVG	✓	49.0	45.9	47.4	✓	62.1	62.1	62.1
	Tweet-attention	✓	51.9	42.3	46.6	✗	80.4	80.4	80.4
	Tweet-CNN	✗	58.8	51.1	54.7	✗	70.7	70.7	70.7
with chronol. LSTM	Word-AVG	-	58.1	58.3	58.2	-	71.9	71.9	71.9
	Word-CNN-AVG	-	60.8	56.1	58.4	-	60.9	60.9	60.9
	Word-attention	-	52.9	42.9	47.4	-	60.9	60.9	60.9
	Tweet-AVG	✗	57.4	60.3	**58.8**	✗	64.6	64.6	64.6
	Tweet-attention	✓	48.2	52.2	50.1	✗	67.0	67.0	67.0
	Tweet-CNN	✗	65.3	49.7	56.4	✗	60.9	60.9	60.9

5.5.2 Sequence labeling results

Table 5.2 illustrates the predictive performance of our proposed model (i.e., using the chronological LSTM) compared to models making independent predictions per bin. The upper part of Table 5.2 contains models without the chronological LSTM. Our experiments study both *word-level* and *tweet-level* bin representations (see Fig. 5.1), as reflected in the 'Word' vs. 'Tweet' prefix, respectively, in the Model column of Table 5.2.

The simplest *word-level* representation uses the tf-idf weighting scheme (as in [3]) followed by an MLP classifier. For the other word-level models, we exploit several architectures: AVG pooling [17], a CNN followed by AVG pooling [23] and hierarchical word-level attention [24].

For *tweet-level* representations, we adopt similar architectures, where the AVG, CNNs and attention are performed on sentence level rather than on the word-level representation of the bin. In this scenario, we have also exploited the usage of sequential LSTMs to represent the tweets. When comparing models with and without tweet-level LSTMs, we report the strategy that yields the best results, indicated by ✓ and ✗ in the tweet-level LSTM (TL) columns of Table 5.2. We do not present results for applying sequential LSTMs on the word-level bin representation, because of slow training on the long word sequences.

Benefit of chronological LSTM: The bottom part of Table 5.2 presents the results of the same models followed by a chronological LSTM to capture

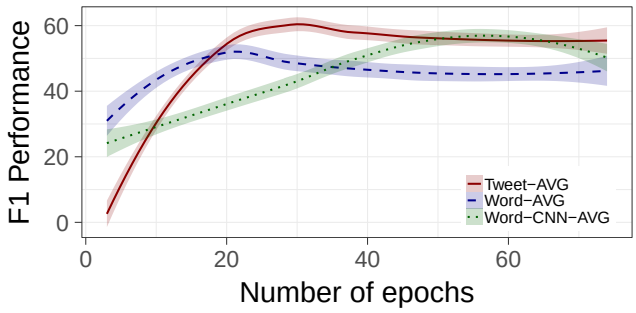

Figure 5.2: *Bin-level* F_1 performance of the three best performing models on the validation set with respect to the number of epochs. The smoothed lines (obtained by LOWESS smoothing) model the trends and the 95% confidence intervals.

the natural flow of the stream as illustrated in Fig. 5.1. We report results as described in Section 5.4, using the micro F_1 score with the two evaluation strategies (*bin-level* and *relaxed*). We observe that when using the chronological LSTM, the performance in terms of *bin-level* F_1 score is substantially improved for almost every model. Note that the best model using the chronological LSTM (Tweet-AVG) achieves 2.4% better F_1 than the best performing model without the use of chronological LSTM (Word-CNN-AVG). In most cases there is also a consistent improvement for both the precision and the recall metrics, which is thanks to the sequential nature of the upper level LSTM capturing the flow of the text.

Limitations of *relaxed* evaluation: On the other hand, using the *relaxed* evaluation strategy, we observe that the best models are those without the chronological LSTM layer. Yet, we consider the *relaxed* evaluation strategy flawed for our scenario, despite the fact that it has been used for entity classification tasks [19, 20]. Indeed, it is not able to properly capture sub-events which are characterized by duration: e.g., if a model assigns a different label to each of the bins that together constitute a single sub-event, then this sub-event counts as a true positive based on the *relaxed* evaluation strategy (similar to the evaluation proposed by [16] and followed in Table 5.1). Thus, in this work, we propose to use the *bin-level* evaluation, since it is a more natural way to measure the duration of a sub-event in a supervised sequence labeling setting. Note that due to the noisy nature of Twitter streams, a tweet sequence spanning a particular sub-event is likely to contain also tweets that are not related to the given sub-event: a given bin inside the event may contain only a minority of tweets discussing the event. Therefore, we consider the standard sequence labeling evaluation

(requiring to have types as well as boundaries correct) to be not applicable in sub-event detection.

Performance comparison of the top-3 models: Figure 5.2 shows the performance of our three best performing models in terms of *bin-level* F_1 score on the validation set. The best performing model is the Tweet-AVG model since it attains its maximum performance even from the first training epochs. The Word-AVG model performs well from the first epochs, showing similar behavior to the Tweet-AVG model. This can be explained by the similar nature of the two models. The word-level CNN model attains maximum performance compared to the other two models in later epochs. Overall, we propose the use of the chronological LSTM with the Tweet-AVG model since this model does not rely on complex architectures and it gives consistent results.

5.6 Conclusion

In this work, we frame the problem of sub-event detection in Twitter streams as a sequence labeling task. Specifically, we (i) propose a binary classification baseline model that outperforms state-of-the-art approaches for sub--event detection (presence/absence), (ii) establish a strong baseline that additionally predicts sub-event *types*, and then (iii) extend this baseline model with the idea of exchanging chronological information between sequential posts, and (iv) prove it to be beneficial in almost all examined architectures.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive feedback. Moreover, we would like to thank Christos Xypolopoulos and Giannis Nikolentzos for providing (i) the Twitter dataset (tweet ids) and (ii) instructions to reproduce the results of their graph-based approach.

References

[1] C. Castillo. *Big Crisis Data: Social Media in Disasters and Time-Critical Situations*. Cambridge University Press, 2016. doi:10.1017/CBO9781316476840.

[2] J. Nichols, J. Mahmud, and C. Drews. *Summarizing Sporting Events Using Twitter*. In Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces, pages 189–198, New York, NY, USA,

2012. ACM. Available from: http://doi.acm.org/10.1145/2166966.2166999, doi:10.1145/2166966.2166999.

[3] D. Pohl, A. Bouchachia, and H. Hellwagner. *Automatic Sub-event Detection in Emergency Management Using Social Media*. In Proceedings of the 21st International Conference on World Wide Web, pages 683–686, New York, NY, USA, 2012. ACM. Available from: http://doi.acm.org/10.1145/2187980.2188180, doi:10.1145/2187980.2188180.

[4] P. Meladianos, G. Nikolentzos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis. *Degeneracy-Based Real-Time Sub-Event Detection in Twitter Stream*. In Proceedings of the 9th International AAAI Conference on Web and Social Media, pages 248–257. AAAI Press, 2015.

[5] C. Xing, Y. Wang, J. Liu, Y. Huang, and W.-Y. Ma. *Hashtag-based Sub-event Discovery Using Mutually Generative LDA in Twitter*. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pages 2666–2672. AAAI Press, 2016. Available from: http://dl.acm.org/citation.cfm?id=3016100.3016274.

[6] Z. Wang and Y. Zhang. *A Neural Model for Joint Event Detection and Summarization*. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, pages 4158–4164. AAAI Press, 2017. Available from: http://dl.acm.org/citation.cfm?id=3171837.3171867.

[7] E. Kouloumpis, T. Wilson, and J. Moore. *Twitter sentiment analysis: The good the bad and the omg!* In Proceedings of the Fifth International AAAI conference on weblogs and social media, pages 538–541, 2011.

[8] T. H. Nguyen and K. Shirai. *Topic Modeling based Sentiment Analysis on Social Media for Stock Market Prediction*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1354–1364. Association for Computational Linguistics, 2015. Available from: http://aclweb.org/anthology/P15-1131, doi:10.3115/v1/P15-1131.

[9] E. D'Andrea, P. Ducange, B. Lazzerini, and F. Marcelloni. *Real-Time Detection of Traffic From Twitter Stream Analysis*. IEEE Transactions on Intelligent Transportation Systems, 16(4):2269–2283, Aug 2015. doi:10.1109/TITS.2015.2404431.

[10] D. Abhik and D. Toshniwal. *Sub-event Detection During Natural Hazards Using Features of Social Media Data*. In Proceedings of the 22nd International Conference on World Wide Web, pages 783–788, New

York, NY, USA, 2013. ACM. Available from: http://doi.acm.org/10.1145/2487788.2488046, doi:10.1145/2487788.2488046.

[11] P. Srijith, M. Hepple, K. Bontcheva, and D. Preotiuc-Pietro. *Sub-story detection in Twitter with hierarchical Dirichlet processes*. Information Processing & Management, 53(4):989 – 1003, 2017. Available from: http://www.sciencedirect.com/science/article/pii/S0306457316300668, doi:https://doi.org/10.1016/j.ipm.2016.10.004.

[12] S. Zhao, L. Zhong, J. Wickramasuriya, and V. Vasudevan. *Human as real-time sensors of social and physical events: A case study of Twitter and sports games*. arXiv preprint arXiv:1106.4300, 2011.

[13] A. Zubiaga, D. Spina, E. Amigó, and J. Gonzalo. *Towards Real-time Summarization of Scheduled Events from Twitter Streams*. In Proceedings of the 23rd ACM Conference on Hypertext and Social Media, pages 319–320, New York, NY, USA, 2012. ACM. Available from: http://doi.acm.org/10.1145/2309996.2310053, doi:10.1145/2309996.2310053.

[14] G. Chen, N. Xu, and W. Mao. *An Encoder-Memory-Decoder Framework for Sub-Event Detection in Social Media*. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pages 1575–1578, New York, NY, USA, 2018. ACM. Available from: http://doi.acm.org/10.1145/3269206.3269256, doi:10.1145/3269206.3269256.

[15] T. Sakaki, M. Okazaki, and Y. Matsuo. *Earthquake shakes Twitter users: real-time event detection by social sensors*. In Proceedings of the 19th international conference on World wide web, pages 851–860. ACM, 2010.

[16] P. Meladianos, C. Xypolopoulos, G. Nikolentzos, and M. Vazirgiannis. *An Optimization Approach for Sub-event Detection and Summarization in Twitter*. In Proceedings of the 40th European Conference in Information Retrieval, pages 481–493. Springer International Publishing, 2018.

[17] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III. *Deep Unordered Composition Rivals Syntactic Methods for Text Classification*. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1681–1691. Association for Computational Linguistics, 2015. Available from: http://aclweb.org/anthology/P15-1162, doi:10.3115/v1/P15-1162.

[18] L. Ramshaw and M. Marcus. *Text Chunking using Transformation-Based Learning*. In Third Workshop on Very Large Corpora, 1995. Available from: http://aclweb.org/anthology/W95-0107.

[19] H. Adel and H. Schütze. *Global Normalization of Convolutional Neural Networks for Joint Entity and Relation Classification*. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1723–1729. Association for Computational Linguistics, 2017. Available from: http://aclweb.org/anthology/D17-1181, doi:10.18653/v1/D17-1181.

[20] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Adversarial training for multi-context joint entity and relation extraction*. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2830–2836. Association for Computational Linguistics, 2018. Available from: http://aclweb.org/anthology/D18-1307.

[21] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Joint entity recognition and relation extraction as a multi-head selection problem*. Expert Systems with Applications, 114:34 – 45, 2018. Available from: https://arxiv.org/abs/1804.07847.

[22] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *An attentive neural architecture for joint segmentation and parsing and its application to real estate ads*. Expert Systems with Applications, 102:100 – 112, 2018. Available from: https://arxiv.org/abs/1709.09590.

[23] Y. Kim. *Convolutional Neural Networks for Sentence Classification*. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751. Association for Computational Linguistics, 2014. Available from: http://aclweb.org/anthology/D14-1181, doi:10.3115/v1/D14-1181.

[24] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. *Hierarchical Attention Networks for Document Classification*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1480–1489. Association for Computational Linguistics, 2016. Available from: http://aclweb.org/anthology/N16-1174, doi:10.18653/v1/N16-1174.

6

Conclusions and Future Research

This chapter outlines the main conclusions for each of the chapters described in this thesis. Specifically, we describe the main contributions for every presented model and we conclude by presenting future research directions that alleviate some of the limitations of the proposed models.

⋆⋆⋆

6.1 Conclusions

6.1.1 Baseline methods for the real estate structured prediction problem

In Chapter 2 of this thesis, we defined the new real estate relation extraction problem [1] where the goal is to recover the tree structured representation of the property. Specifically, a comparative study has been conducted on the newly defined real estate relation extraction problem where we divided the problem into the sub-problems of sequence labeling and dependency parsing. We used (1) Conditional Random Fields (CRFs) for the real estate entity recognition subtask, (2) local and global graph-based, and transition-based algorithms for dependency parsing to predict the part-of rela-

tions between the identified entities, and (3) a maximum spanning tree algorithm for recovering the *property tree*. Experimental results illustrate that (i) the global graph-based approach outperforms other alternative methods when the gold entities are given while (ii) the locally based method combined with a maximum spanning tree algorithm performs better (i.e., improvement of 1% F_1 points) in our pipeline setting.

6.1.2 Neural joint model for the real estate structured prediction problem

In Chapter 3, we proposed a new joint model for the real estate structured prediction problem that unlike the pipeline models presented in Chapter 2 solves the subtasks of NER and dependency parsing simultaneously [2]. This way, we resolved the limitations of tackling the two tasks in a pipeline fashion: we (i) avoid the error propagation that would arise from the execution of consecutive subtasks, and (ii) exploit the interactions between the subtasks. We experimentally evaluated the performance of the proposed model over several pipeline methods that have been developed for this task (presented in Chapter 2) and we reported an improvement of over three percentage points in the overall edge F_1 score of the property tree. Moreover, we experimented with different attentive architectures over our basic joint model. The results indicate that exploiting attention mechanisms that encourage our model to focus on informative tokens, improves the model performance (increase of overall edge F_1 score with $\sim$2.1%) and increases the ability to form valid trees in the prediction phase (4% to 10% more valid trees for the two best scoring attention mechanisms) before applying the maximum spanning tree algorithm (see step (3) in Section 6.1.1).

6.1.3 General purpose neural joint model for NER and relation extraction

In Chapter 4A, we presented a joint neural model that reduces the quadratic complexity of the NER module introduced in Chapter 3. The model presented in Chapter 4A aims at simultaneously extracting entities and relations from textual data and comprises a CRF layer for the entity recognition task and a sigmoid layer for the relation extraction task [3]. The sigmoid layer facilitates the extraction of multiple relations per entity modeling the problem as multi-label head selection task. Unlike previous models on this joint task that rely on external NLP tools (i.e., POS taggers, dependency parsers), the proposed methodology produces automatically generated features and can be adapted into any language and context. Thus, the performance of our model is not affected by the accuracy of external or

hand-crafted features. Experimental results illustrate the effectiveness of our approach by conducting a large scale experimental study.

In Chapter 4B, we proposed to use adversarial training (AT) as a regularization method over the joint model for entity recognition and relation extraction [4] proposed in Chapter 4A. We showed how to apply adversarial perturbations to improve the robustness of our model. Results indicate that AT improves the results for each task separately and the overall performance of the joint model while it reaches high performance even for the first iterations of the training procedure.

6.1.4 Sub-event detection from Twitter streams as a sequence labeling problem

Unlike previous Chapters 2-4B, where we examined the tasks of NER and relation extraction, Chapter 5 focused on the sub-event detection task from Twitter streams [5]. Specifically, we framed the problem of sub-event detection in Twitter streams as a sequence labeling task (this is similar to the NER task examined in previous Chapters 2-4B). In particular, we (i) proposed a baseline model that outperforms state-of-the-art approaches for sub-event detection (presence/absence of a sub-event), (ii) established a strong baseline for predicting sub-event types, (iii) built upon this baseline model and introduce the idea of exchanging chronological information between sequential posts which (iv) is proven beneficial in almost any of the examined architectures.

6.2 Future Directions

In this thesis, we (i) defined the new real estate structure prediction problem, which we formulated as a pipeline of two subtasks (see Chapter 2), (ii) proposed a neural joint model that simultaneously extracts the entities and the relations for this application scenario (see Chapter 3), (iii) developed a general purpose joint model that can achieve state-of-the-art results in various settings and languages (see Chapter 4A) combined with adversarial perturbations as a regularization method (see Chapter 4B), and (iv) formulated the sub-event detection problem from Twitter streams as a sequence labeling problem to capture the chronological order of the tweets. Even though the proposed methodologies have achieved state-of-the-art performance in the respectively examined tasks, there are several promising research directions. In the next paragraphs, we describe future research directions for improving the models discussed in Chapters 3-5.

Auxiliary tasks: The proposed joint model (Chapters 3-4B) is currently trained on two specific tasks (i.e., NER and relation extraction). However, as mentioned in [6–8], studying tasks together with other auxiliary tasks in a multitask learning setting can lead to performance improvements. This idea has not been exploited yet in the current architecture. Thus, we propose to enhance our training scheme for the joint setting of entity and relation extraction with other auxiliary tasks. That way our model (i.e., the one for joint entity and relation extraction) will benefit from different tasks by using different representations of the data learned by these other auxiliary tasks without the need of additional training data. The information sharing is usually implemented by sharing layers in the neural architecture. To select appropriate auxiliary tasks to jointly consider, existing works have investigated [7, 8] the benefit of sharing information between particular tasks. For instance, we can (i) exploit the relatedness between sequence labeling tasks such as POS tagging, keyword detection, etc. as described in [7], and (ii) study additional tasks in an increasingly complex fashion as explained in [8].

Transfer learning: Another idea for improving the current joint model is to exploit the use of transfer learning methods. Transfer learning leads to high performance improvements in several NLP problems. Specifically, recent works such as ELMo [9], BERT [10] and ULMFiT [11] demonstrated significant performance gains on problems ranging from text classification to question answering. Thus, we can exploit the use of pre-trained language models for our joint architecture. This way the joint models (presented in Chapters 3-4B) will be trained on better representations of words. These representations using language modeling architectures can capture the meaning of the entire context of the word instead of just focusing on the meaning of the word based only on adjacent terms. This is extremely useful in cases where we have polysemous terms. For instance, in the biology domain where some genes and diseases share the same nomenclature as reported in [12], we can use such contextual word representations to obtain substantial improvements.

Few-shot learning: Finally, we can study the problem of entity and relation extraction in a few-shot learning setting. In this setting, we are interested in making predictions in cases where we have seen few training instances. The current neural network architecture for the joint model needs a large amount of training examples. This is a common characteristic of all the neural network architectures (i.e., they need a lot of data to generalize well). Although the idea of few-shot learning for relation extraction has been investigated in the work of [13] (where the relation extraction task

has been formulated as a question answering problem), the solution they propose introduces an additional annotation load and uses complex question answering systems to solve the task at hand (i.e., relation extraction). Thus, to address scenarios where there is a limited amount of training data in the task of entity and relation extraction, we propose to use alternative methods (similar to the work of [14]) which can be applied on top of the existing state-of-the-art joint models presented in Chapters 4A-4B.

Future research steps: Natural language understanding is a hard problem to solve even when we consider only the English language. However, the problem can be even more complicated when we consider that there are more than 7000 languages throughout the world and more than 90 languages with more than 10 million native speakers [15]. Although there is an increasing need for solving NLP tasks (e.g., text classification) in multiple languages, collecting a large amount of annotated data for each language is not realistic due to the increased annotation load. Nevertheless it is not straightforward to train a model for a specific task (e.g., text classification) in a particular language (e.g., English) and apply it directly on (another) language with limited training data (i.e., low-resource scenarios). Several methods have been proposed to tackle this type of cross-lingual problems (see [16, 17]). The best performing systems in these works rely on machine translation methods. Machine translation for solving cross-lingual tasks can be used by translating (i) the source to the target language and then training a model (e.g., classifier) for each target language, and (ii) each target language to the source language and use the model (e.g., classifier) of the source language. These methods have several shortcomings. For instance, using method (i) leads to maintaining one model for each target task while method (ii) leads to translating to the source language at inference time. An alternative solution is to use cross-lingual word embeddings [18] but this leads to performance decrease compared to standard monolingual word embeddings. Transfer learning and meta-learning methods are promising future directions that can further increase the performance of multi-lingual systems on difficult NLP tasks especially in low-resource scenarios (i.e., languages with limited training data). This is because transfer learning and meta-learning methods have been also proposed for few-shot learning settings. These approaches can be applied in the way that we described to the previous paragraphs of this section.

References

[1] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Reconstructing the house from the ad: Structured prediction on real estate classifieds*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 2, Short Papers), pages 274–279, Valencia, Spain, 3–7 Apr. 2017.

[2] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *An attentive neural architecture for joint segmentation and parsing and its application to real estate ads*. Expert Systems with Applications, 102:100–112, 2018. doi:10.1016/j.eswa.2018.02.031.

[3] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Joint entity recognition and relation extraction as a multi-head selection problem*. Expert Systems with Applications, 114:34–45, 2018.

[4] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Adversarial training for multi-context joint entity and relation extraction*. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2830–2836, 2018.

[5] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. *Sub-event detection from twitter streams as a sequence labeling problem*. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 745–750, Minneapolis, Minnesota, June 2019. doi:10.18653/v1/N19-1081.

[6] A. Søgaard and Y. Goldberg. *Deep multi-task learning with low level tasks supervised at lower layers*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, page 231, 2016.

[7] J. Bingel and A. Søgaard. *Identifying beneficial task relations for multi-task learning in deep neural networks*. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 164–169, 2017.

[8] K. Hashimoto, Y. Tsuruoka, R. Socher, et al. *A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks*. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1923–1933, 2017.

[9] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. *Deep Contextualized Word Representations*. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, 2018.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. *Bert: Pre-training of deep bidirectional transformers for language understanding*. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2019.

[11] J. Howard and S. Ruder. *Universal Language Model Fine-tuning for Text Classification*. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 328–339, 2018.

[12] W. Yoon, C. H. So, J. Lee, and J. Kang. *CollaboNet: collaboration of deep neural networks for biomedical named entity recognition*. In Proceedings of the ACM 12th International Workshop on Data and Text Mining in Biomedical Informatics, 2018.

[13] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer. *Zero-Shot Relation Extraction via Reading Comprehension*. In Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), pages 333–342, 2017.

[14] C. Finn, P. Abbeel, and S. Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. In Proceedings of the International Conference on Machine Learning (ICML 2017), pages 1126–1135, 2017.

[15] *Summary by language size*. https://www.ethnologue.com/statistics/size. Accessed: 2019-05-25.

[16] A. Conneau, R. Rinott, G. Lample, A. Williams, S. Bowman, H. Schwenk, and V. Stoyanov. *XNLI: Evaluating Cross-lingual Sentence Representations*. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2475–2485, 2018.

[17] A. Eriguchi, M. Johnson, O. Firat, H. Kazawa, and W. Macherey. *Zero-Shot Cross-lingual Classification Using Multilingual Neural Machine Translation*. arXiv preprint arXiv:1809.04686, 2018.

[18] S. Ruder, I. Vulic, and A. Søgaard. *A Survey of Cross-Lingual Word Embedding Models*. 2018.