Natural Language Processing (Almost) from Scratch

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, Pavel Kuksa (2011)

Presented by: Dibyanshu Kumar (5123766)
Bhumika Sadbhave (5123767)
Devadharshni Rajarajeshwari (5123749)

Introduction

Ideology and Goal of the Paper

Traditional NLP approach

- Extract rich set of hand-designed features (based on linguistic intuition, trial and error)
  - Task dependent
- Complex tasks (SRL) then require a large number of possibly complex features (eg: extracted from aparse tree)
  - Impacts the computational cost

Proposed System

- Task Specific Engineering
- Pre-process features as little as possible - Make it generalisable
- Single Learning System to discover adequate internal representations
- Use a multilayer neural network (NN) architecture trained in an end-to-end fashion

Benchmark Tasks

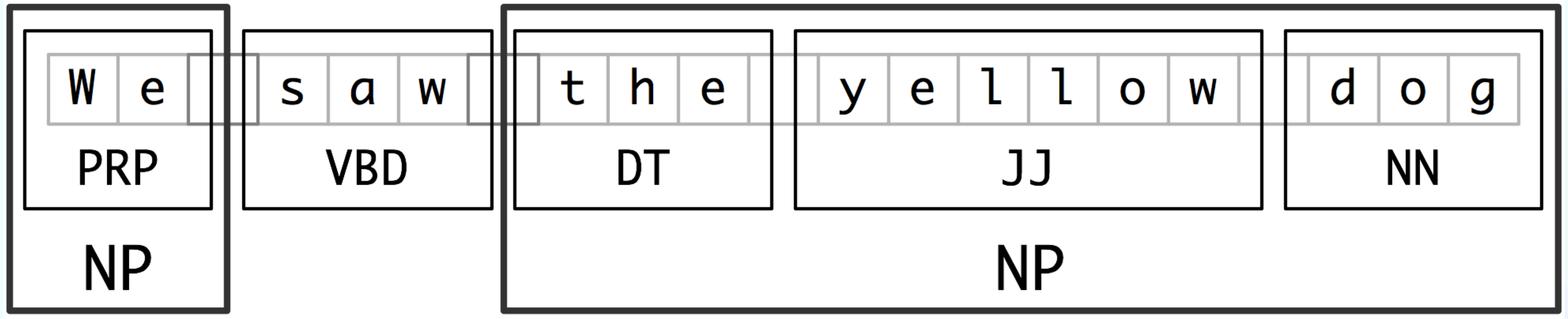
POS (Parts of Speech) Tagging

Benchmark Tasks

- Label word with syntactic tag (verb, noun, adverb...)
- Best POS classifiers:
  - Trained on windows of text, which are then fed to bidirectional decoding algorithm during inference
  - Features - previous and next tag context, multiple words (bigrams, trigrams. . .) context

Chunking

Benchmark Tasks

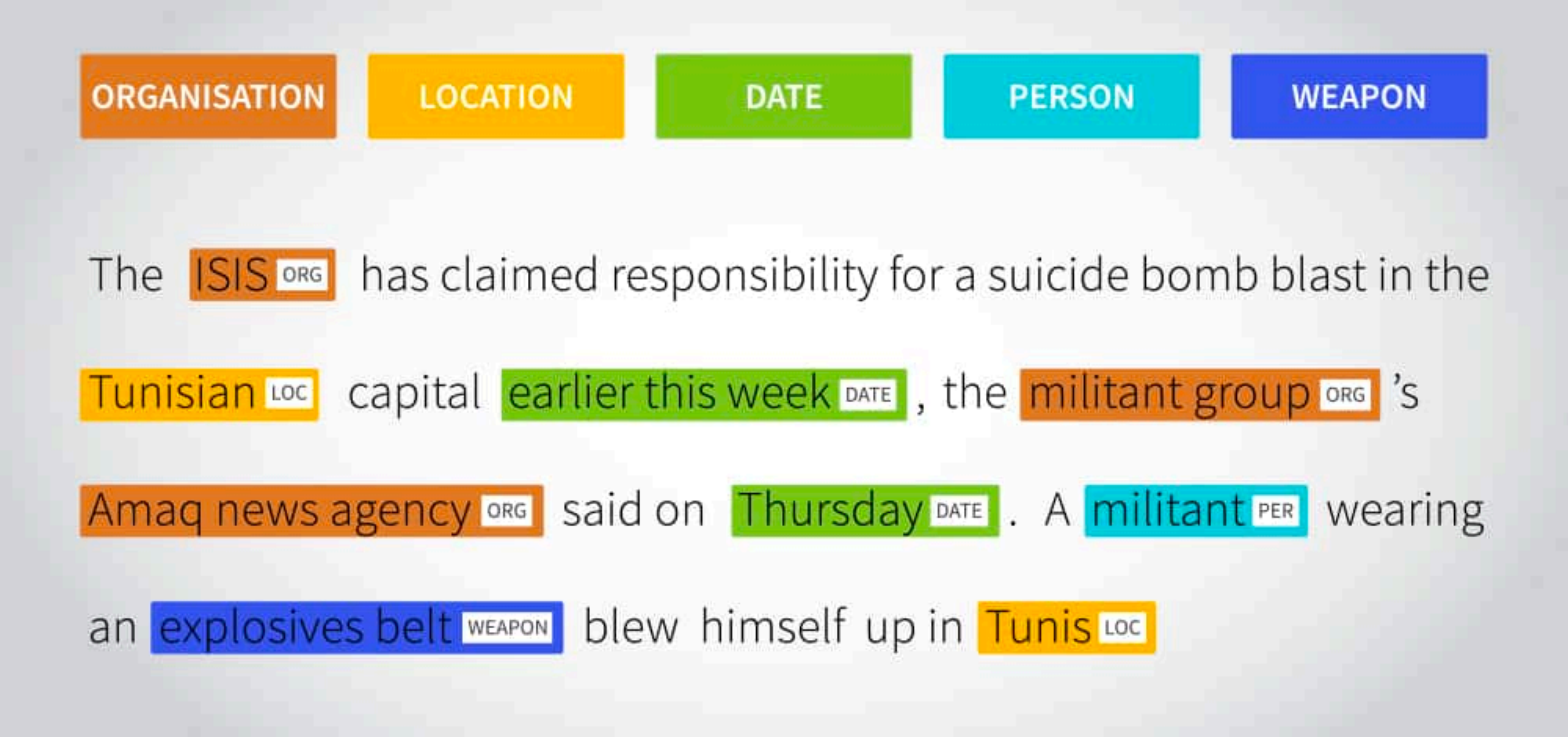


- Labelling segments of a sentence with syntactic constituents (NP or VP)
- Each word assigned only one unique tag, encoded as begin-chunk (B-NP) or inside-chunk tag (I-NP)
- Evaluated using CoNLL shared task

Named Entity Recognition

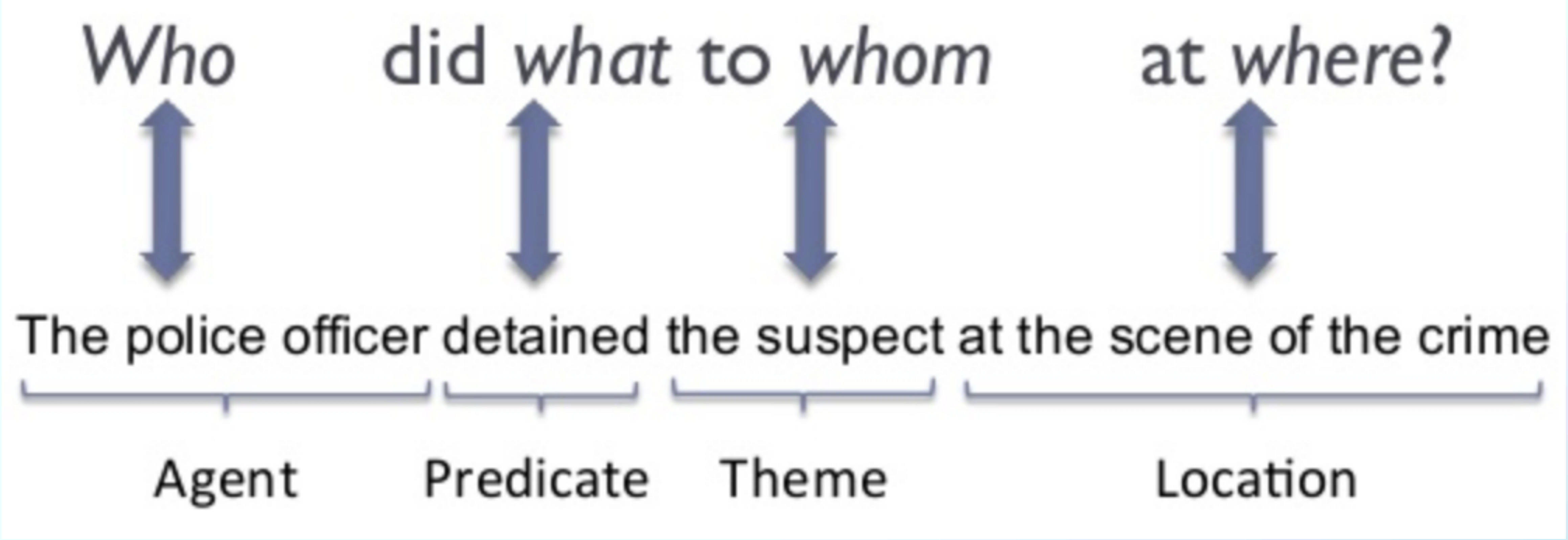
Benchmark Tasks

- Labelling atomic elements in the sentence into categories (“PERSON”, “LOCATION”)



Semantic Role Labelling

Benchmark Tasks



- Gives a semantic role to a syntactic constituent of a sentence
- State-of-the-art SRL systems:
  - Producing a parse tree
  - Identifying which parse tree nodes represent the arguments of a given verb
  - Classifying nodes to compute the corresponding SRL tags

The Networks

Transforming Words into Feature Vectors

The Networks

- For efficiency, words are fed as indices taken from a finite dictionary D
- **First layer:** maps each of these word indices into a feature vector, by a ***lookup table*** operation.
- Initialise the word lookup table with these representations (instead of randomly)
- For each word, an internal d-dimensional feature vector representation given by the lookup table layer LTW (·):

$$LT_W(w) = \langle W \rangle_w^1 ,$$

where W: Matrix of parameters to be learned, ⟨W⟩: wth column of W

- Given a sentence or any sequence of T words, the output matrix produced -

$$LT_W([w]_1^T) = \begin{pmatrix} \langle W \rangle_{[w]_1}^1 & \langle W \rangle_{[w]_2}^1 & \dots & \langle W \rangle_{[w]_T}^1 \end{pmatrix}$$

Window-based Approach

Extracting Higher Level Features

- Assumes the tag of a word depends on its neighbouring words
- Word feature window given by lookup table

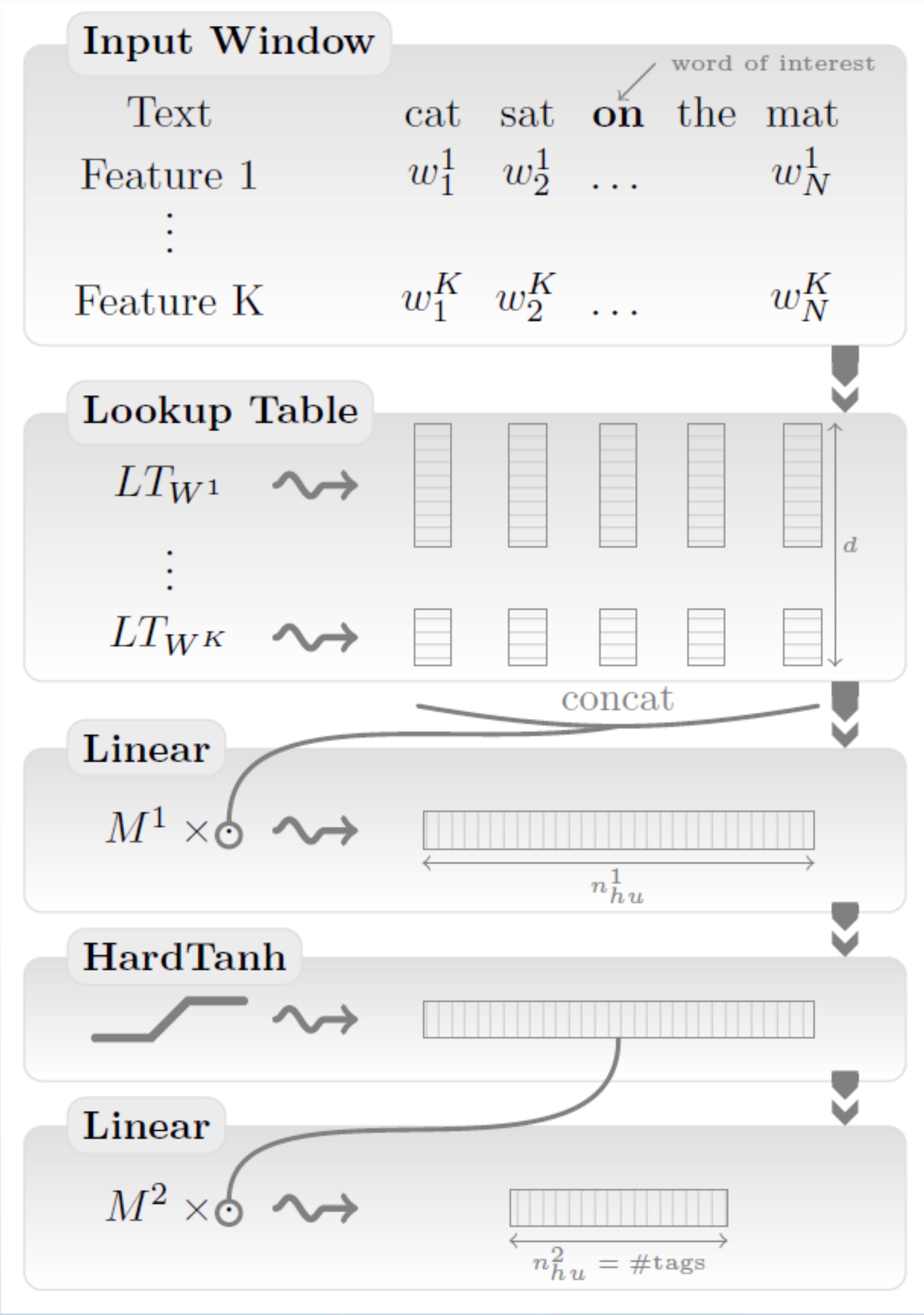
$$f_\theta^1 = \langle LT_W([w]_1^T)\rangle_t^{d_{win}} = \begin{pmatrix} \langle W\rangle^1_{[w]_{t-d_{win}/2}} \\ \vdots \\ \langle W\rangle^1_{[w]_t} \\ \vdots \\ \langle W\rangle^1_{[w]_{t+d_{win}/2}} \end{pmatrix}$$

- Linear Layer: $f_\theta^l = W^l f_\theta^{l-1} + b^l$
- HardTanh Layer:

$$\left[f_\theta^l\right]_i = \mathrm{HardTanh}\left(\left[f_\theta^{l-1}\right]_i\right)$$

$$\mathrm{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 <= x <= 1 \\ 1 & \text{if } x > 1 \end{cases}$$

- Scoring: size of number of tags with corresponding score
- Feature window is not well defined for words near the beginning or the end of a sentence - augment the sentence with a special “PADDING” akin to the use of “start” and “stop” symbols in sequence models.



Sentence-based Approach

Extracting Higher Level Features

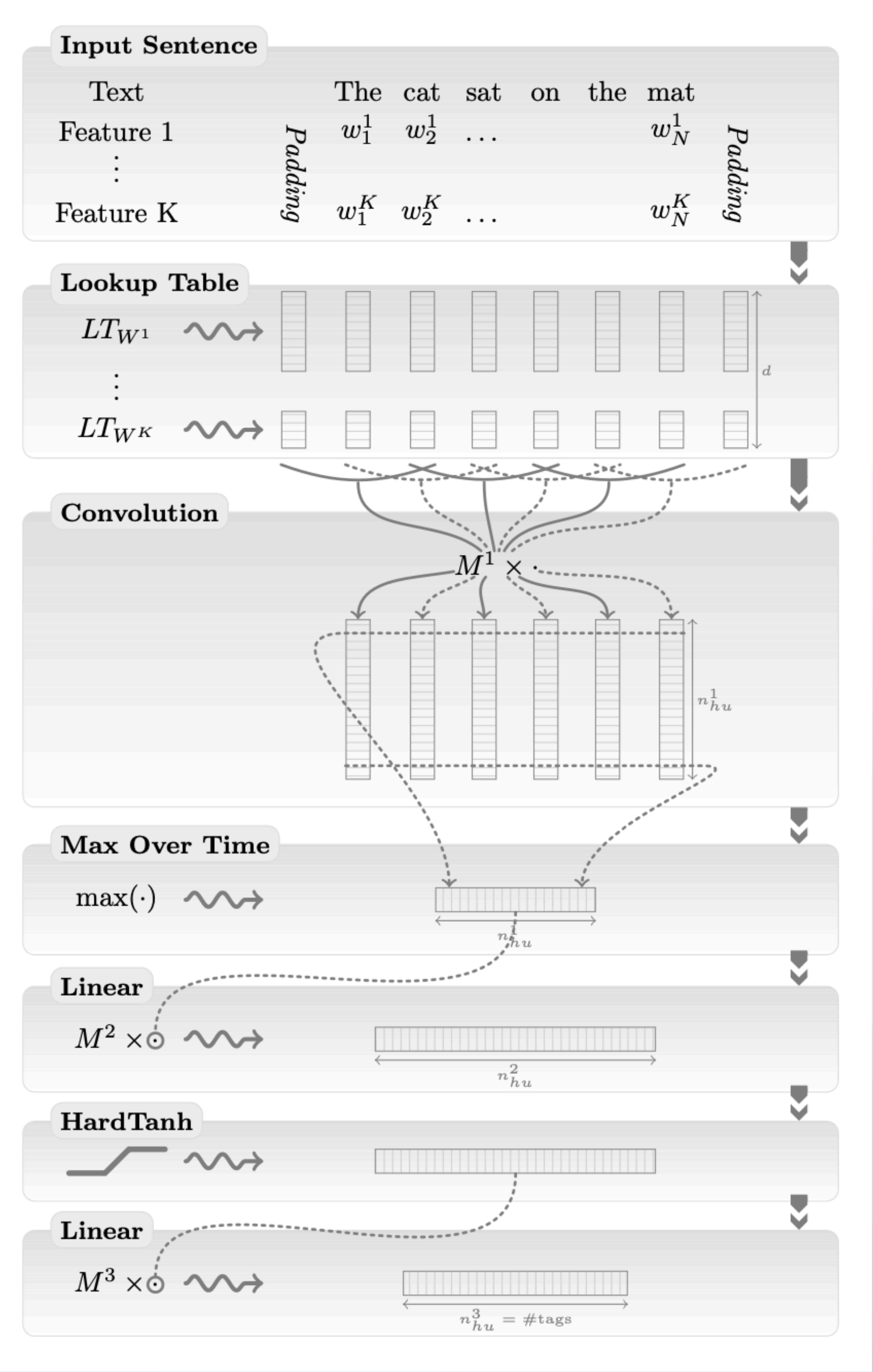
- Window approach fails with SRL, where the tag of a word depends on a verb chosen beforehand in the sentence

- **Convolutional Layer:** Generalisation of a window approach - for all windows t, output column of layer l

$$\langle f_\theta^l \rangle_t^1 = W^l \, \langle f_\theta^{l-1} \rangle_t^{d_{win}} + b^l \quad \forall t$$

- **Max Layer:**
  - Average operation does not make much sense - Most words in the sentence do not have any influence on the semantic role of a given word to tag
  - Max approach forces the network to capture the most useful local features

$$\left[f_\theta^l\right]_i = \max_t \left[f_\theta^{l-1}\right]_{i,t} \qquad 1 \leq i \leq n_{hu}^{l-1}$$



Tagging Schemes

The Networks

- **Window approach:** Tags apply to the word located in the centre of the window

- **Sentence approach:** Tags apply to the word designated by additional markers in the network input

Scheme	Begin	Inside	End	Single	Other
IOB	B-X	I-X	I-X	B-X	O
IOE	I-X	I-X	E-X	E-X	O
IOBES	B-X	I-X	E-X	S-X	O

- Each word in a segment labeled “X” is tagged with a prefixed label, depending of the word position in the segment (begin, inside, end)

- Words not in a labeled segment are labeled “O”. Variants of the IOB (and IOE) scheme exist, where the prefix B (or E) is replaced by I for all segments not contiguous with another segment having the same label “X”

Training the Networks

The Networks

- Trained by maximising a likelihood over the training data, using stochastic gradient ascent
- Likelihood function

$$\theta \mapsto \sum_{(x,y)\in\mathcal{T}} \log p(y|x,\theta)$$

- Stochastic gradient: maximisation is achieved by iteratively selecting a random example (x, y) and making a gradient step

$$\theta \longleftarrow \theta + \lambda \frac{\partial \log p(y|x,\theta)}{\partial\theta}$$

Word-Level Log Likelihood

Training Networks

- Each word in a sentence is considered independently
- conditional tag probability p(i | x, θ) by applying a softmax:

$$p(i\,|\,x,\theta) = \frac{e^{[f_\theta]_i}}{\sum_j e^{[f_\theta]_j}}.$$

- Defining the log-add operation as:

$$\log p(y\,|\,x,\theta) = [f_\theta]_y - \operatorname*{logadd}_j\,[f_\theta]_j$$

- log-likelihood for one training example (x, y):

$$\operatorname*{logadd}_i z_i = \log\Big(\sum_i e^{z_i}\Big)$$

Sentence-Level Log Likelihood

Training Networks

- Enforces dependencies between the predicted tags in a sentence (needed for NER or SRL)
- Introduce scores:
  - Transition score [A]ij : from i to j tags in successive words
  - Initial score [A]i0 : starting from the ith tag
- Score of sentence along a path of tags, using initial and transition scores

$$s([x]_1^T, [i]_1^T, \tilde{\theta}) = \sum_{t=1}^{T} \left([A]_{[i]_{t-1}, [i]_t} + [f_\theta]_{[i]_t, t} \right)$$

- Maximise this score
- Viterbi algorithm for inference $\underset{[j]_1^T}{\operatorname{argmax}} \, s([x]_1^T, [j]_1^T, \tilde{\theta})$

Supervised Benchmark Results

Training the Networks

Approach	POS (PWA)	Chunking (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99

Results are **behind** the benchmark results

FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869	MEGABITS 87025
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

- neighbouring words in the embedding space **do not seem to be semantically related**
- Word embeddings in the word lookup table trained with a dictionary of size **100,000**.
- Queried word is followed by: index in the dictionary and its **10 nearest neighbors** (using the Euclidean metric)

Performance Improvement

- Using Unlabelled data (Semi-supervised Learning)
- Multi-task Learning
- Task-specific Engineering

Using Unlabelled data

Performance Improvement

- Model performance hinges on initialisation of look-up table
- Use unlabelled data(with window architecture) to improve embedding for the initialisation
- First corpus - entire English Wikipedia - tokenise - **631 million words**
- Second corpus - **extra 221 million words** extracted from the Reuters data set.
- Extended the lookup table dictionary to **130,000 words** (added 30,000 most common words in Reuters)
- Use ranking criterion instead of entropy (SLL and WLL previously described) - higher score if the phrase is legal, lower score for incorrect
- Training: stochastic gradient minimisation of the ranking criterion

Performance vs Benchmark Results

Using Unlabelled Data

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99
NN+WLL+LM1	97.05	91.91	85.68	58.18
NN+SLL+LM1	97.10	93.65	87.58	73.84
NN+WLL+LM2	97.14	92.04	86.96	58.34
NN+SLL+LM2	97.20	93.63	88.67	74.15

NN: Neural Network

LM: Language Model

WLL: Word-level Log Likelihood

SLL: Sentence-level Log Likelihood

FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869	MEGABITS 87025
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Syntactic and Semantic properties of the neighbours are **clearly related** to those of the query word

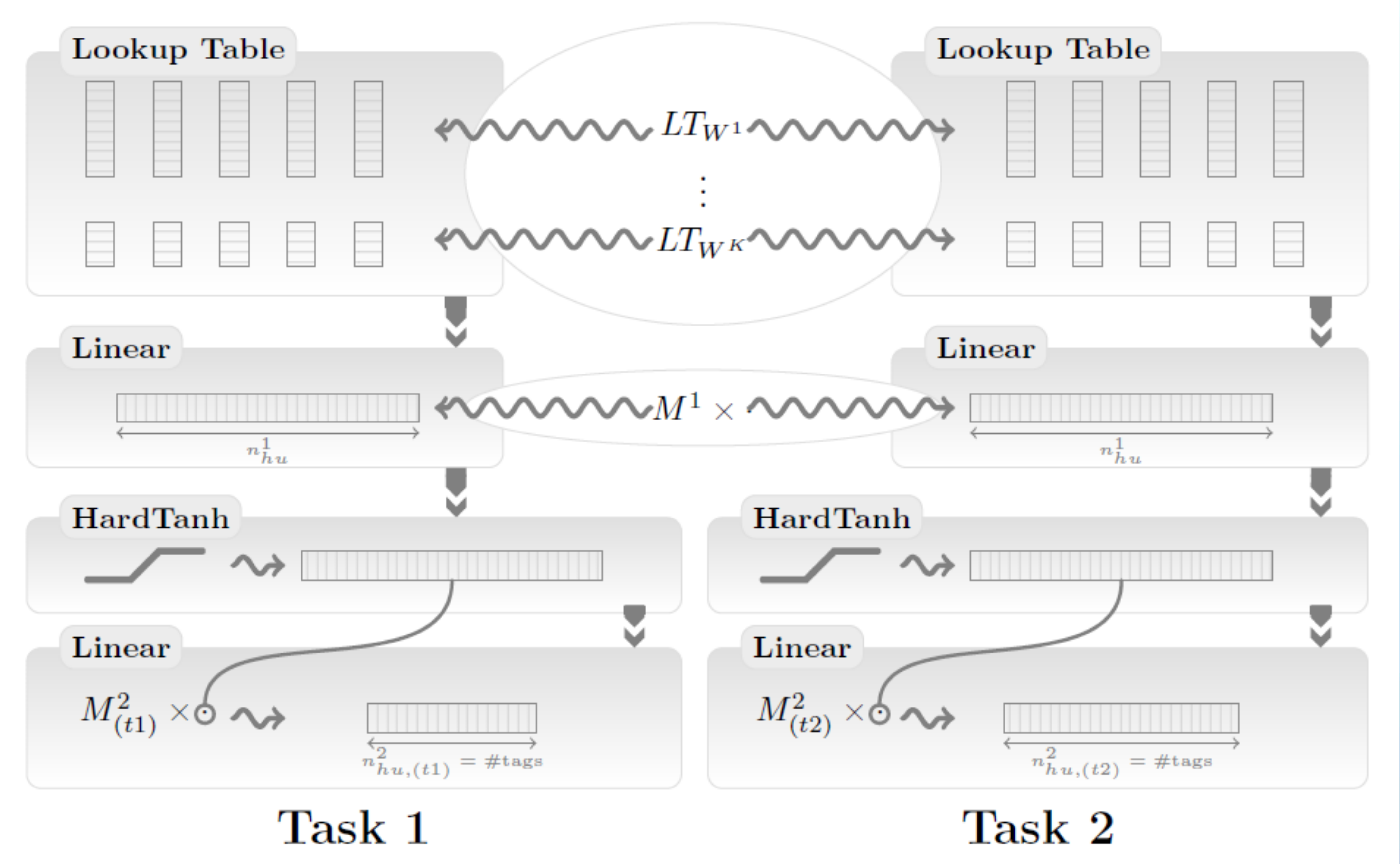
-**Initialising the word lookup tables** of the networks with the embeddings computed by the language models

- **significantly boosts** the generalisation **performance** of the supervised networks for each task

Multi-task Learning

Training Goal: minimising the loss averaged across all tasks

- Features trained for one task can be useful for related tasks
- Models for all tasks of interests are jointly trained
- Additional linkage between their trainable parameters
- **Joint Decoding:** Combining the predictions of independently trained models i.e decoding the outputs of different tasks together
- **Joint training:** Training sets for the individual tasks contain the same patterns with different labels. Multiple outputs for each pattern
- WLL: parameters of Linear layers shared
- SLL: parameters of Convolution Layer shared



Example of multitasking with NN (Window architecture)
Lookup tables, first hidden layer: Shared Last layer: Task specific

Multi-task Learning Performance

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
	Window Approach			
NN+SLL+LM2	97.20	93.63	88.67	–
NN+SLL+LM2+MTL	97.22	94.10	88.62	–
	Sentence Approach			
NN+SLL+LM2	97.12	93.37	88.78	74.15
NN+SLL+LM2+MTL	97.22	93.75	88.27	74.29

- POS, CHUNK, NER trained in a MTL way, for window and sentence network approaches
- SRL included in the sentence approach joint training
- sentence approach for the POS, Chunking, and NER tasks yields no performance improvement (or degradation) over the window approach
- Training was achieved by minimising the **loss averaged** across all tasks

Leads to **Marginal improvements** over using a separate network for each task

Task-specific Engineering

The Temptation

Brown Clusters?
builds hierarchical word clusters to calculate word embeddings(instead of using lookup tables)

- **Suffix features for POS Tagging:** Two character word suffixes, suffix dictionary size = **455**
- **Gazetteers for NER**: gazetteer provided by the CoNLL challenge, containing **8,000** locations, person names, organisations, and miscellaneous entities.
- **Cascading:** Training CHUNK and NER networks with additional POS features; and training the SRL network with additional CHUNK features
- **Ensembles:** Combining the outputs of multiple classifiers trained with different tagging conventions
- **Parsing:** Providing Charniak parse tree (with CoNLL 2005 data) as an additional input feature for SRL task
- **Word Representations:** Word representations derived from the Brown Clusters are provided as input feature - no significant improvement; embeddings at least as good as Brown Clusters
- **Engineering a Sweet Spot:** Standalone version of our architecture, written in the C language

Suffix Features, Gazetteer, Cascading

Task-specific Engineering

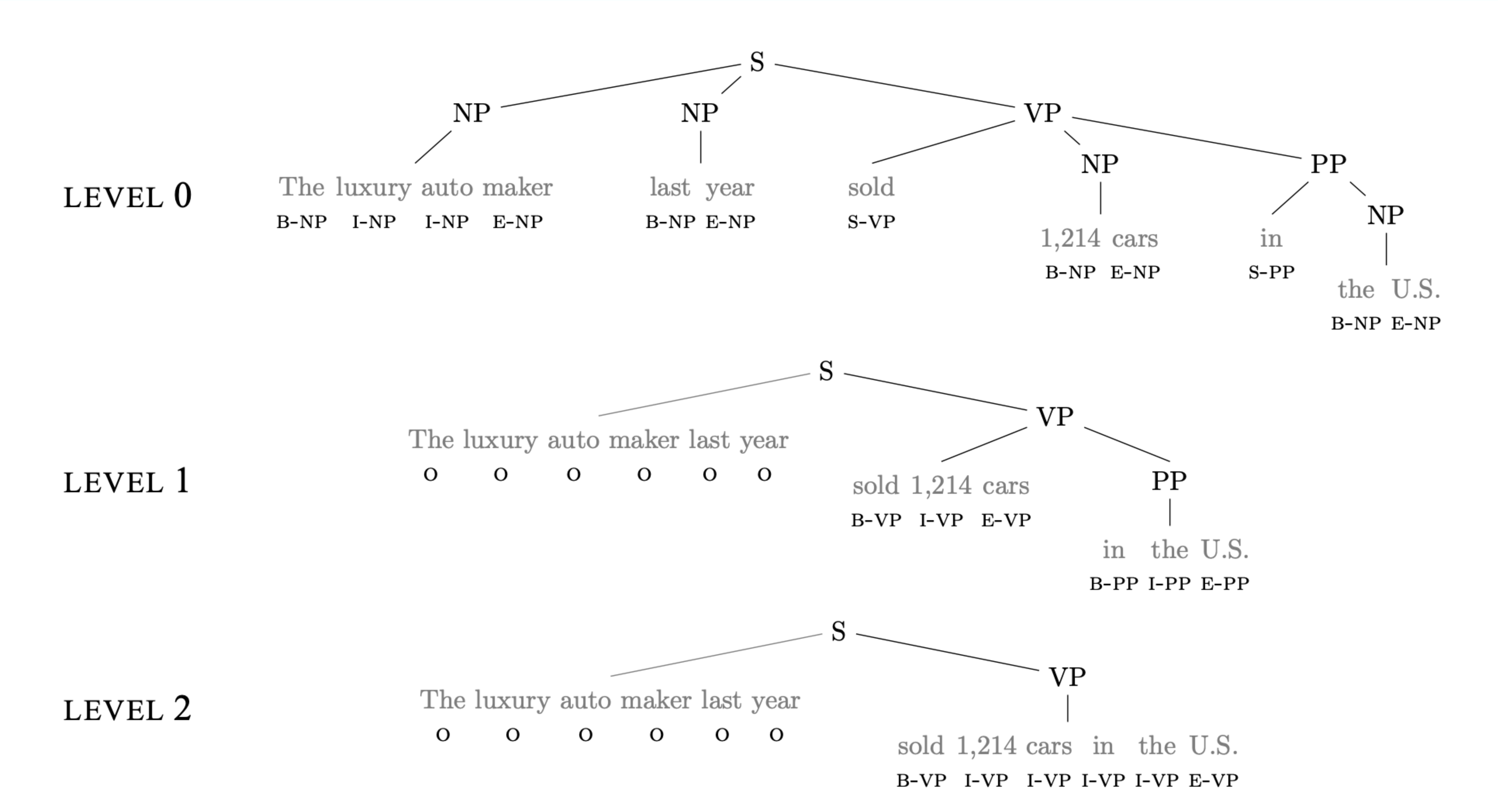
Approach	**POS** (PWA)	**CHUNK** (F1)	**NER** (F1)	**SRL**
Benchmark Systems	97.24	94.29	89.31	77.92
NN+SLL+LM2	97.20	93.63	88.67	74.15
NN+SLL+LM2+Suffix2	97.29	–	–	–
NN+SLL+LM2+Gazetteer	–	–	89.59	–
NN+SLL+LM2+POS	–	94.32	88.67	–
NN+SLL+LM2+CHUNK	–	–	–	74.72

- POS network: trained with two character word suffixes;
- NER network: trained using the small CoNLL 2003 gazetteer
- CHUNK and NER networks: trained with additional POS features
- SRL network: trained with additional CHUNK features.

- NER+Gazetteer and POS+Suffix2: **outperforms the baseline**
- Adding POS and chunk - **consistent moderate improvements**
- Lower than the benchmark for SRL

Parsing for SRL

Task-specific Engineering



Charniak parse tree for the sentence "The luxury auto maker last year sold 1,214 cars in the U.S.".

Performance

Approach	SRL (valid)	SRL (test)
Benchmark System (six parse trees)	77.35	77.92
Benchmark System (top Charniak parse tree only)	74.76	–
NN+SLL+LM2	72.29	74.15
NN+SLL+LM2+Charniak (level 0 only)	74.44	75.65
NN+SLL+LM2+Charniak (levels 0 & 1)	74.50	75.81
NN+SLL+LM2+Charniak (levels 0 to 2)	75.09	76.05
NN+SLL+LM2+Charniak (levels 0 to 3)	75.12	75.89
NN+SLL+LM2+Charniak (levels 0 to 4)	75.42	76.06
NN+SLL+LM2+CHUNK	–	74.72
NN+SLL+LM2+PT0	–	75.49

- Accuracy improved after adding levels 0 to 4
- Top performance reaches **76.06%** F1 score. This is not too far from the benchmark which uses six parse trees instead of one

"SENNA" (Semantic/Syntactic Extraction using a NN Architecture)

Standalone version of architecture, written in the C language

all tasks run on single 3GHz Intel core

Performance on the tagging tasks

Task		Benchmark	SENNA
Part of Speech (POS)	(Accuracy)	97.24 %	97.29 %
Chunking (CHUNK)	(F1)	94.29 %	94.32 %
Named Entity Recognition (NER)	(F1)	89.31 %	89.59 %
Parse Tree level 0 (PT0)	(F1)	91.94 %	92.25 %
Semantic Role Labeling (SRL)	(F1)	77.92 %	75.49 %

needs less than a ms per word to compute tags

Runtime speed and memory consumption comparison

POS System	RAM (MB)	Time (s)
Toutanova et al. (2003)	800	64
Shen et al. (2007)	2200	833
SENNA	32	4

SRL System	RAM (MB)	Time (s)
Koomen et al. (2005)	3400	6253
SENNA	124	51

Conclusion

Features used by SENNA

Task	Features
POS	Suffix of size 2
CHUNK	POS
NER	CoNLL 2003 gazetteer
PT0	POS
SRL	PT0, verb position

- Fast and efficient "all purpose" NLP tagger
- Avoids task-specific engineering as much as possible
- Relies on large unlabelled data sets and allows the training algorithm discover internal representations
- Milestone for solving NLP tasks using Neural Network Approach
- BONUS - SENNA link: https://ronan.collobert.com/senna/

Thank You