A Short RNN talk

Juhani Luotolahti FinMT 2016
mjluot@utu.fi

Optimizing Recurrent Neural Network Models

- So, what's the topic of this presentation?
- Optimizing Recurrent Neural Networks
  - Could mean many things:
  - Maybe optimizing the performance of these models on let's say GPU?
    - Those matrix calculation tricks used by the deep-learning libraries?
    - Nope, don't know much about that
    - But here's a nice link from NVidia for those interested: https://devblogs.nvidia.com/parallelforall/optimizing-recurrent-neural-networks-cudnn-5/
  - So it must be about hyperparameter optimization then, right?
    - Well, yes.
    - Complex theoretical methods to solve all problems related to model selection and tuning?
    - No, sorry.

Optimizing Recurrent Neural Networks #2

- So what will be presented?
- 1) A short introduction
- 2) TurkuNLP system for shared task on cross-lingual pronoun prediction
  - The task
  - The model
  - How the model came to be
  - How much does its architecture affect its performance
  - How much do a few hyperparameters affect its performance
- 3) A few closing words

The TurkuNLP Pronoun system

- TurkuNLP System for Cross-Lingual Pronoun Prediction (Luotolahti, Kanerva, Ginter, 2016)
- Took part in WMT16 Shared Task on CrossLingual Pronoun Prediction
- MT-related task
- A deep recurrent neural network architecture
- Achieves the best macro recall (official task metric) on all four language pairs
- Margin to the next best system ranges between less than 1pp and almost 12pp depending on the language pair

The Pronoun Prediction Task #1

- Translating pronouns across language poses problems for machine translation systems
- How is a pronoun, for example ‘That’ translated to another language given a circumstance?
- Can be seen as a subtask of machine translation
- The task contained four language pairs:
  - En-Fr
  - Fr-En
  - De-En
  - En-De
- And sufficiently parallel data for each one of the pairs

The Pronoun Prediction Task #2

- The task setting simulating a processing step in a machine translation system
- The data, for each sentence contains:
  - The original, tokenized, source sentence
  - Target sentence, lemmatized with POS-tags
  - REPLACE placeholder tokens for pronouns to be predicted in the target sentence
  - Alignments between source and target sequences
- The data is based on three different datasets:
  - The Europarl dataset (Koehn, 2005)
  - news commentary corpora (IWSLT15, NCv9),
  - TED corpus

The Pronoun Prediction Task #3

Typical training example:

Source:

That ’s how they like to live .

Target:

ce|PRON etre ˆ |VER comme|ADV cela|PRON que|PRON **REPLACE_3** aimer|VER vivre|VER .|.

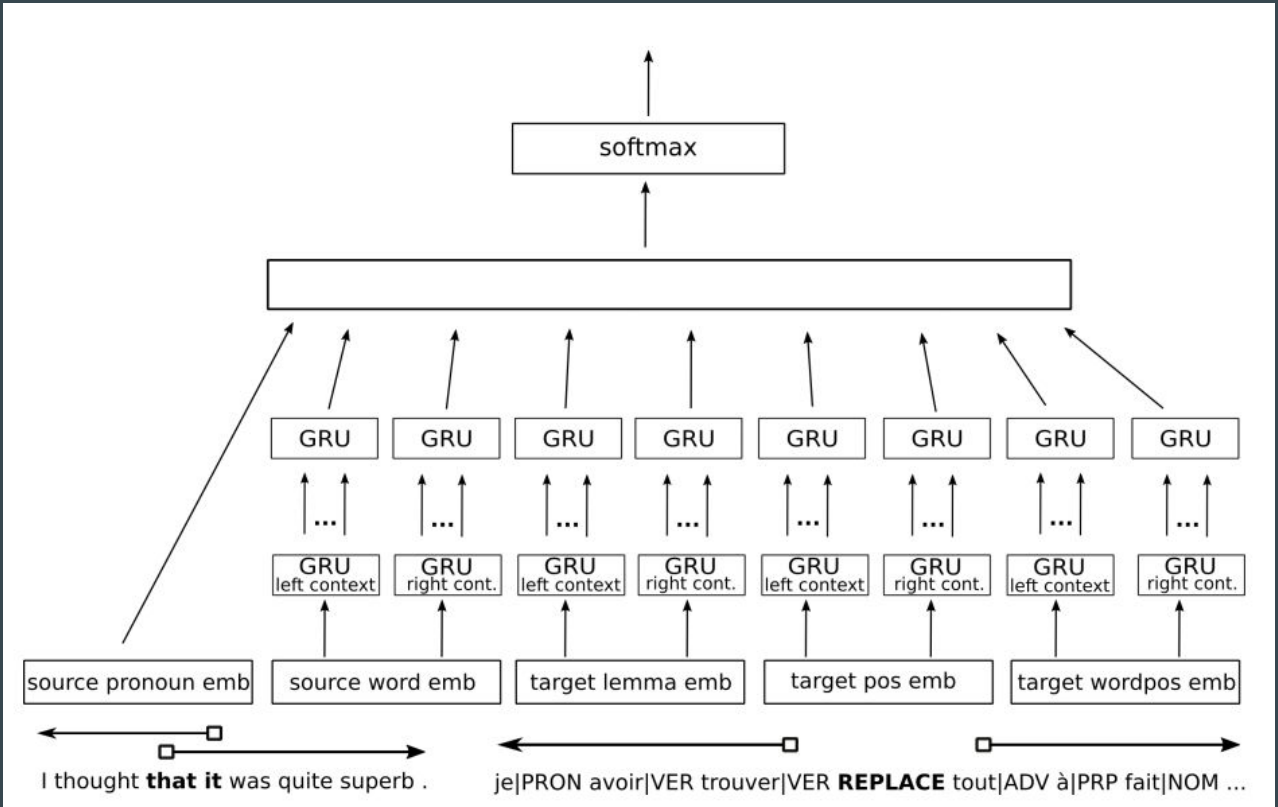
Our System

- Basically a deep recurrent neural network model with learned token-level embeddings, two layers of Gated Recurrent Units (GRUs), a dense network layer with rectified linear unit (ReLU) activation, and a softmax layer
- Altogether 16 GRUs
- Embeddings for:
  - Source Pronoun,
  - Source Token,
  - Target Lemma
  - Target POS-tag
  - Target wordpos

Our System #2

- The network has 9 inputs
  - Source sentence token context to the right of the Pronoun and separately to the left
  - Target sentence lemma context to the right of the Pronoun to be predicted and separately to the left
  - Target sentence POS context to the right and to the left of the Pronoun to be predicted
  - Target sentence WordPOS context to the right and to the left and to the right of the Pronoun to be predicted
  - The source token aligned with the Pronoun to be predicted

Our system #3


softmax
GRU
GRU left context
GRU right cont.
source pronoun emb
source word emb
target lemma emb
target pos emb
target wordpos emb
I thought **that it** was quite superb .
je|PRON avoir|VER trouver|VER **REPLACE** tout|ADV à|PRP fait|NOM ...

Building The System

- While the model relies on learned embeddings instead of predefined set of features, a process similar to feature engineering takes place while designing the system architecture
- Design choices were made in a greedy manner and mostly the system was built additively, testing new features and adding the promising ones to the final system
- Not all design choice combinations were properly tested during the system development
- That is because of the huge search space, and also because of the fact we had limited time in our hands
- But let's look at a few on the next slide
- Implemented in Keras
    - http://keras.io

Training and Tuning The System

- Only the training data provided by the shared task organizers was used
- Our primary submission is trained to optimize Macro Recall, the official metric for the task
  - Done by weighting the loss of the training examples relative to the frequencies of the classes, so that misclassifying a rare class is seen by the network as more serious mistake than misclassifying a common class.
  - Decision to do this was made on the last minute
  - We also delivered a contrastive system without
- We decided, mainly for congruency related reasons, to not carry out any language specific optimization for the models in the final submission
  - So the model parameters were the same for all language pairs in our submission

Training and Tuning The System #2

- The final model used for test set prediction was selected simply by its performance on the development set
- The final architecture of the system and the way training data was used was also selected on its performance on the development set
- All models were trained in the fantastic resource offered by CSC, the taito GPU cluster

The Effect of A Few Hyperparameters on the Model

- To illustrate the effect of the hyperparameter values on the model a few parameters were tested
- To use hyperbolic tangent activation instead of relu on the final dense layer decreases system performance on all language pairs by average of 5 pp
- The effect of modifying the size of the said last dense layer between yields differences on the order of 5 pp on all language pairs
- Swapping the GRU for LSTM generally decreases performance on all language pairs except French-English
- So, just these few simple hyperparameters affect the system performance quite dramatically

Feature Evaluation #1

	De-En		En-De		Fr-En		En-Fr	
Architecture	Macro R	Micro F	Macro R	Micro F	Macro R	Micro F	Macro R	Micro F
primary	**73.91**	75.36	64.41	71.54	**72.03**	80.79	65.70	70.51
no stacking	65.63	75.98	61.84	73.37	68.84	77.74	70.00	74.26
only in-domain	59.18	75.36	50.72	66.06	57.80	74.09	58.09	65.15
short context	61.29	73.50	**65.66**	71.80	65.84	79.59	69.27	70.51
cross-sentence	60.76	70.81	46.91	49.61	60.46	78.05	61.33	69.17
contrastive	72.60	**80.54**	58.39	72.85	66.54	**85.06**	61.46	72.39
no stacking	65.35	79.30	59.71	**76.76**	61.23	81.71	**70.88**	**77.75**

Oh, okay

- So, this was to demonstrate that on a relatively typical neural model, much like those seen in MT-tasks, these choices matter surprisingly much
- And these were just a very few example parameters / design-choices
- There's plenty:
  - Dropouts for RNNs
  - Dropout layers
  - Embedding sizes
  - Layer sizes
  - Regularization parameters
  - Batch sizes
  - ...

On Building Models

- As mentioned earlier, our modus operandi for building the system was a simple greedy, explorative and additive development
- That simply means we started small and simple, tried something and if it worked it was added in
- Parameters such as vector sizes were tested with very small scale grid searches during development
- Everything was tested against development set and overfitting was tried to be avoided by comparing loss with training and dev sets
- Very Simple Stuff!
- I have to admit that when you put it out there, it sounds very messy, raw and not too theoretically sound
    - Part of the messiness of the process is explained by our relative hurry to get the system done

On Building Models #2

- So, in a nutshell; for architectural types of decisions we just tried things, and for numerical parameters we basically did small grid searches
- While the approach just described doesn't sound like something to write home about and most certainly isn't a silver bullet for anything, something like this is a very typical tool of the trade
- Is that it? Is there nothing being done about this?
    - Yes, there is

Hyperparameter Optimization Methods for NNs

- Methods for optimizing parameters which are not part of the parameters to be learned by the model
- Grid Search
  - Brute force search through all relevant data points. Computationally intensive and subject to limitations on the feature space
- Random Search
  - Instead of doing a full search, search feature space by sampling. Uses fraction of the computational time for similar performance when compared to grid search (Bergstra & Bengio, 2012).
- Bayesian Optimization
  - Application of generic bayesian black-box optimization methods for neural network hyperparameters. (Snoek, 2012)
- Gradient based optimization methods
  - (Maclaurin et al., 2015)

What about architectonic choices?

- What about network architecture?
  - An insane idea!
  - Endless search spaces
- Previous work exists:
  - The gradient method (Maclaurin et al., 2015) is able to optimize parts of network architecture
  - Genetic algorithms and reinforcement learning for creating neural networks (Stanley et al., 2002)
  - Genetic algorithms to create recurrent neural network topologies (Blanco et al., 2000)
- Of course none of these is able to create the modern fine-tuned monster networks
- Intuition, experience & experimentation still lives
- But maybe, just maybe, in the future

Thanks!

http://bionlp.utu.fi