Report from Dagstuhl Seminar 17042

From Characters to Understanding Natural Language (C2NLU): Robust End-to-End Deep Learning for NLP

Organized by

Phil Blunsom[1], Kyunghyun Cho[2], Chris Dyer[3], and Hinrich Schütze[4]

1 **University of Oxford, GB,** phil.blunsom@cs.ox.ac.uk
2 **New York University, US,** kyunghyun.cho@nyu.edu
3 **Carnegie Mellon University - Pittsburgh, US,** cdyer@cs.cmu.edu
4 **LMU München, DE,** hs2016@cislmu.org

Edited by

Heike Adel[5] and Yadollah Yaghoobzadeh[6]

5 **LMU München, DE,** heike@cis.lmu.de
6 **LMU München, DE,** yadollah@cis.lmu.de

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 17042 "From Characters to Understanding Natural Language (C2NLU): Robust End-to-End Deep Learning for NLP". The seminar brought together researchers from different fields, including natural language processing, computational linguistics, deep learning and general machine learning. 31 participants from 22 academic and industrial institutions discussed advantages and challenges of using characters, i.e., "raw text", as input for deep learning models instead of language-specific tokens. Eight talks provided overviews of different topics, approaches and challenges in current natural language processing research. In five working groups, the participants discussed current natural language processing/understanding topics in the context of character-based modeling, namely, morphology, machine translation, representation learning, end-to-end systems and dialogue. In most of the discussions, the need for a more detailed model analysis was pointed out. Especially for character-based input, it is important to analyze what a deep learning model is able to learn about language – about tokens, morphology or syntax in general. For an efficient and effective understanding of language, it might furthermore be beneficial to share representations learned from multiple objectives to enable the models to focus on their specific understanding task instead of needing to learn syntactic regularities of language first. Therefore, benefits and challenges of transfer learning were an important topic of the working groups as well as of the panel discussion and the final plenary discussion.

Seminar January 22–27, 2017 – http://www.dagstuhl.de/17042
1998 ACM Subject Classification I.2 Artificial Intelligence, I.2.7 Natural Language Processing
Keywords and phrases Natural Language Understanding, Artificial Intelligence, Deep Learning, Natural Language Processing, Representation Learning
Digital Object Identifier 10.4230/DagRep.7.1.129

1 Executive Summary

Phil Blunsom
Kyunghyun Cho
Chris Dyer
Hinrich Schütze
Yadollah Yaghoobzadeh

Deep learning is currently one of most active areas of research in machine learning and its applications, including natural language processing (NLP). One hallmark of deep learning is *end-to-end learning*: all parameters of a deep learning model are optimized *directly for the learning objective*; e.g., for the objective of accuracy on the binary classification task: is the input image the image of a cat? Crucially, the set of parameters that are optimized includes "first-layer" parameters that connect the raw input representation (e.g., pixels) to the first layer of internal representations of the network (e.g., edge detectors). In contrast, many other machine learning models employ hand-engineered features to take the role of these first-layer parameters.

Even though deep learning has had a number of successes in NLP, research on true end-to-end learning is just beginning to emerge. Most NLP deep learning models still start with a hand-engineered layer of representation, the level of tokens or words, i.e., the input is broken up into units by manually designed tokenization rules. Such rules often fail to capture structure both within tokens (e.g., morphology) and across multiple tokens (e.g., multi-word expressions). Given the success of end-to-end learning in other domains, it is likely that it will also be widely used in NLP to alleviate these issues and lead to great advances.

The seminar brought together researchers from deep learning, general machine learning, natural language processing and computational linguistics to develop a research agenda for the coming years. The goal was to combine recent *advances in deep learning architectures and algorithms* with *extensive domain knowledge about language* to make *true end-to-end learning for NLP* possible.

Our goals were to make progress on answering the following research questions.

- C2NLU approaches so far fall short of the state of the art in cases where token structures can easily be exploited (e.g., in well-edited newspaper text) compared to word-level approaches. What are promising avenues for developing C2NLU to match the state of the art even in these cases of text with well-defined token structures?
- Character-level models are computationally more expensive than word-level models because detecting syntactic and semantic relationships at the character-level is more expensive (even though it is potentially more robust) than at the word-level. How can we address the resulting challenges in scalability for character-level models?
- Part of the mantra of deep learning is that domain expertise is no longer necessary. Is this really true or is knowledge about the fundamental properties of language necessary for C2NLU? Even if that expertise is not needed for feature engineering, is it needed to design model architectures, tasks and training regimes?
- NLP tasks are diverse, ranging from part-of-speech tagging over sentiment analysis to question answering. For which of these problems is C2NLU a promising approach, for which not?

- More generally, what characteristics make an NLP problem amenable to be addressed using tokenization-based approaches vs. C2NLU approaches?
- What specifically can each of the two communities involved – natural language processing and deep learning – contribute to C2NLU?
- Create an NLP/deep learning roadmap for research in C2NLU over the next 5–10 years.

2 Table of Contents

3 Introduction: C2NLU

Phil Blunsom (University of Oxford, GB), Kyunghyun Cho (New York University, US), Chris Dyer (Carnegie Mellon University – Pittsburgh, US), and Hinrich Schütze (LMU München, DE)

This section contains the motivation for the seminar given in the proposal submitted by the organizers in 2015 to the Schloss Dagstuhl – Leibniz-Zentrum für Informatik. It has been slightly edited.

3.1 Introduction

Deep learning is currently one of most active areas of research in machine learning and its applications, including natural language processing (NLP). One hallmark of deep learning is *end-to-end learning*: all parameters of a deep learning model are optimized *directly for the learning objective*; e.g., for the objective of accuracy on the binary classification task: is the input image the image of a cat? Crucially, the set of parameters that are optimized includes "first-layer" parameters that connect the raw input representation (e.g., pixels) to the first layer of internal representations of the network (e.g., edge detectors). In contrast, many other machine learning models employ hand-engineered features to take the role of these first-layer parameters.

Even though deep learning has had a number of successes in NLP, research on true end-to-end learning is just beginning to emerge. Most NLP deep learning models still start with a hand-engineered layer of representation, the level of tokens or words, i.e., the input is broken up into units by manually designed tokenization rules. Such rules often fail to capture structure both within tokens (e.g., morphology) and across multiple tokens (e.g., multi-word expressions). Given the success of end-to-end learning in other domains, it is likely that it will also be widely used in NLP to alleviate these issues and lead to great advances.

3.2 Goals of the Seminar

The seminar brought together researchers from deep learning, general machine learning, natural language processing and computational linguistics to develop a research agenda for the coming years. The goal was to combine recent *advances in deep learning architectures and algorithms* with *extensive domain knowledge about language* to make *true end-to-end learning for NLP* possible.

Our goals were to make progress on answering the following research questions.

- C2NLU approaches so far fall short of the state of the art in cases where token structures can easily be exploited (e.g., in well-edited newspaper text) compared to word-level approaches. What are promising avenues for developing C2NLU to match the state of the art even in these cases of text with well-defined token structures?
- Character-level models are computationally more expensive than word-level models because detecting syntactic and semantic relationships at the character-level is more expensive (even though it is potentially more robust) than at the word-level. How can we address the resulting challenges in scalability for character-level models?

- Part of the mantra of deep learning is that domain expertise is no longer necessary. Is this really true or is knowledge about the fundamental properties of language necessary for C2NLU? Even if that expertise is not needed for feature engineering, is it needed to design model architectures, tasks and training regimes?
- NLP tasks are diverse, ranging from part-of-speech tagging over sentiment analysis to question answering. For which of these problems is C2NLU a promising approach, for which not?
- More generally, what characteristics make an NLP problem amenable to be addressed using tokenization-based approaches vs. C2NLU approaches?
- What specifically can each of the two communities involved – natural language processing and deep learning – contribute to C2NLU?
- Create an NLP/deep learning roadmap for research in C2NLU over the next 5–10 years

3.3 Detailed Description of the Topic

C2NLU, i.e., for approaches to end-to-end deep learning in which either the input or the output or both are character streams.

The arguments for C2NLU we present below are closely related and can be thought of as different perspectives on the same underlying problems of token-based approaches.

3.3.1 Robustness against noise

Human natural language processing (NLP) is robust in the sense that small perturbations of the input do not affect processing negatively. Such perturbations include letter insertions, deletions, substitutions and transpositions and the insertion of spaces ("guacamole" → "gua camole") and the deletion of spaces ("ran fast" → "ranfast"). Such perturbations can cause complete failure of token-based processing. C2NLU has the potential of being robust against this type of noise.

3.3.2 Robust morphological processing

There currently does not exist an approach to morphological processing that handles both inflectional and derivational morphology and works across languages with typologically different systems of morphology. If we give up the notion that a token is an opaque symbol and instead model the sequence of characters it is made up of, then we can in principle learn all morphological regularities: inflectional and derivational regularities as well as a wide typological range of morphological processes such as vowel harmony, agglutination, reduplication and nonconcatenativity (as in Arabic and Hebrew). Further, this information can be integrated at the sequence level to permit the learning of agreement and other morpho-syntactic phenomena.

Two caveats are in order. First, it is clear that C2NLU has the potential of successful acquisition of morphology, but whether it can do so in practice is a big question. Second, adopting C2NLU does not mean that linguistically informed models will be replaced with "linguistically ignorant" models. Instead, C2NLU can be a complement to traditional morphological processing in computational linguistics, for example, in a system combination approach. C2NLU is also a promising framework for incorporating linguistic knowledge about morphology as an inductive bias into statistical models. The latter has been done with only limited success in standard statistical NLP models.

3.3.3 Orthographic productivity

It is clear that the truth is between two extreme positions: (i) the character sequence of a token is arbitrary and uninformative and (ii) the character sequence of a token perfectly and compositionally predicts its linguistic properties. Morphology is the most important phenomenon of limited predictability that lies between these two extremes. But there are many other less prominent phenomena that taken together have the potential of improving NLP models and performance of NLP systems considerably if they could be handled at the level of human competence.

- Properties of names predictable from character patterns, e.g., "Yaghoobazadeh" is identifiable as a Farsi surname, "Darnique" and "Delonda" are identifiable as names of girls (in a US context), "osinopril" is most likely a medication
- Orthographic blends and modifications of existing words, e.g., "staycation", "Obamacare", "mockumentary", "dramedy", "cremains"
- Non-morphological orthographic productivity in certain registers, domains and genres: character repetition in tweets ("cooooooooooool"), shm-reduplication ("fancy-shmancy"), the pseudo-derivational suffix "-gate" signifying "scandal" ("Watergate", "Irangate", "Dieselgate")
- Sound symbolism, phonesthemes, e.g., "gl-" ("glitter", "gleam", "glint", "glisten", "glow")
- Onomatopoeia, e.g., "oink", "sizzle", "tick tock"

3.3.4 OOV analysis

One big advantage of character models is that the problem of out-of-vocabulary (OOV) words disappears.[1] Of course, if the NLP system encounters a character string that was never observed before and that would be opaque even to a human reader, then the situation is not better than it would be for a token-based model encountering an OOV. However, as discussed above, in many cases a great deal can be predicted from the character string of an OOV.

So this argument – character models are a promising approach to OOV analysis – is a summary of the last three arguments. Character models are more robust against noise (which may lead to OOVs), have the potential for more robust morphological processing (many OOVs are due to inflection and derivation) and can handle orthographic productivity better ("Yaghoobazadeh", "osinopril" and "staycation" are relevant examples for words that a token-based system may not have observed in the training set).

3.3.5 OOV generation

There is currently no principled and general way for token-based end-to-end systems to generate tokens that are not part of the training vocabulary. Since a token is represented as a vocabulary index and parameters governing system behavior affecting this token are referring to this vocabulary index, a token that does not have a vocabulary index cannot easily be generated in end-to-end systems.

One application in which this is a critical problem is the generation of names in end-to-end machine translation. In the simplest case, a name like "Obama" must be copied from source sentence to target sentence. More complicated cases involve transliteration (English "Putin"

[1] However, there will occasionally be out-of-alphabet characters if special cases like emoji and rare diacritics occur in the input.

becomes French "Poutine") and number variation ("4.12 million" may become "4 million" in a summary). In token-based systems, these cases are often handled separately with external mechanisms, such as an external transliteration system coupled with named-entity detection. However, this is inherently limited as it requires a separate system, which is not jointly tuned together with the main system, for each and every special case.

Character-based systems do not have a problem with OOV generation in principle. However, as with other potential advantages, the problem of practical feasibility arises: specific proposals as to how C2NLU systems can learn to accurately generate OOVs are needed.

3.3.6 Tokenization-free models

One of the drawbacks of token-based models is that they usually tokenize text early on and it is difficult to correct these early tokenization decisions later on. While it is theoretically possible to generate all possible tokenizations and pass any tokenization ambiguity through the entire NLP pipeline (e.g., by using lattices), this is inefficient and often incompatible with the requirements of subsequent processing modules. For this reason, text-to-text machine translation systems usually only consider a single tokenization of source and target.

Tokenization causes limited damage in English although even in English there are difficult cases like "Yahoo!", "San Francisco-Los Angeles flights", "The Iowa campaign manager was selected 'The Apprentice'-style." and hashtags like "#starwars". In other languages, tokenization is even more problematic. In Chinese, tokens are not separated by spaces or other typesetting conventions. For most NLP applications, German compounds should be split. Tokens in agglutinative languages like Turkish are difficult to process as unanalyzed symbols.

3.3.7 Direct models of the data

Traditional machine learning and especially statistical natural language processing rely heavily on feature engineering. In contrast, the philosophy of deep learning is to use any knowledge about the domain for careful design of the architecture of models and for task definitions and training regimes that optimally exploit available data. This careful modeling should then obviate the need for manual feature engineering.

One motivation for this approach is the view that a good set of features can best be found by training a well designed model in a well designed experimental setup. In contrast, manual feature design is prone to errors and omissions.

The success of deep learning in speech, vision and machine translation demonstrates the potential of giving models direct access to the data as opposed to through the intermediary of human-designed features.

However, in natural language processing – and even in neural machine translation – there is still one set of manually designed features that is almost universally used: tokens. Given the success of the raw-data approach for other applications, an approach that takes it to the extreme – character-based models – seems worth investigating for natural language processing. Character-based models are the most faithful to deep learning philosophy: they model the data as it comes in without any alteration through manually designed features. It is an open question whether character-based models will ultimately turn out to be superior to token-based models, but it is an important question that we want to investigate in this workshop.

4 Overview of Talks

4.1 C2NLU: An Overview

Heike Adel (LMU München, DE)

Natural language processing (NLP) and natural language understanding (NLU) often build on a pipeline of different preprocessing modules, such as tokenization, sentence segmentation as well as syntactic and semantic analysis. This pipeline is prone to subsequent errors. However, recovering from those errors is often impossible or not feasible. Alternatively, NLU models could directly model the raw input data, i.e., sequences of characters. Especially with deep learning models, it is possible that the models learn their own representation of the data without the need of tokenizing the character sequence into words. In fact, character-based features have a long history in NLP/NLU, especially in information retrieval, grapheme-to-phoneme conversion or language identification. Recently, more and more studies use characters as input to (hierarchical) neural networks.

This talk provides an overview of character-based NLP/NLU systems. It motivates the usage of characters as features for machine learning systems and/or as input to deep learning models. Furthermore, it presents existing studies on character-based NLU, clustering them into three categories: tokenization-based models, bag-of-n-gram models and end-to-end models.

4.2 Should Model Architecture Reflect Linguistic Structure?

Chris Dyer (Carnegie Mellon University – Pittsburgh, US)

Sequential recurrent neural networks (RNNs) over finite alphabets are remarkably effective models of natural language. RNNs now obtain language modeling results that substantially improve over long-standing state-of-the-art baselines, as well as in various conditional language modeling tasks such as machine translation, image caption generation, and dialogue generation. Despite these impressive results, such models are a priori inappropriate models of language. One point of criticism is that language users create and understand new words all the time, challenging the finite vocabulary assumption. A second is that relationships among words are computed in terms of latent nested structures rather than sequential surface order (Chomsky, 1957; Everaert, Huybregts, Chomsky, Berwick, and Bolhuis, 2015).

In this talk I discuss two models that explore the hypothesis that more (a priori) appropriate models of language will lead to better performance on real-world language processing tasks. The first composes sub word units (bytes, characters, or morphemes) into lexical representations, enabling more naturalistic interpretation and generation of novel word forms. The second, which we call recurrent neural network grammars (RNNGs), is a new generative model of sentences that explicitly models nested, hierarchical relationships among words and phrases. RNNGs operate via a recursive syntactic process reminiscent of probabilistic context-free grammar generation, but decisions are parameterized using RNNs that condition on the entire (top-down, left-to-right) syntactic derivation history, greatly relaxing context-free independence assumptions. Experimental results show that RNNGs obtain better results in generating language than models that don't exploit linguistic structures.

4.3 From Bayes Decision Rule to Neural Networks for Human Language Technology (HLT)

Hermann Ney (RWTH Aachen, DE)

The last 40 years have seen dramatic progress in machine learning and statistical methods for speech and language processing like speech recognition, handwriting recognition and machine translation. Most of the key statistical concepts were originally developed for speech recognition. Examples of such key concepts are the Bayes decision rule for minimum error rate and probabilistic approaches to acoustic modeling (e.g., hidden Markov models) and language modeling. Recently, the performance of HLT systems for speech recognition were improved significantly by the use of artificial neural networks, such as deep feedforward multi-layer perceptrons and recurrent neural networks (including long short-term memory extension). We will discuss these approaches in detail and how they fit into the probabilistic approach to HLT.

4.4 Cross-Token Character N-Gram Modeling: The Other Shoe To Drop?

Hinrich Schütze (LMU München, DE) and Yadollah Yaghoobzadeh (LMU München, DE)

Representation learning in NLP is usually performed on the level of tokens, which requires segmentation or tokenization of input sentences. Here, we introduce an alternative that is completely nonsymbolic: random segmentation of the input (within-token or cross-token) into character n-grams. This applies to training the parameters of the model on a training corpus as well as to applying it when computing the representation of a new text. We give linguistic motivation as well as reporting experimental results, hypothesizing that this way of representation learning could be effective and overcome shortcomings of other methods.

4.5 Modeling Multiple Sequences: Explorations, Consequences and Challenges

Orhan Firat (Middle East Technical University – Ankara, TR)

Deep (recurrent) neural networks have been shown to successfully learn complex mappings between arbitrary length input and output sequences in varying input granularity, such as words, sub-words, characters or even unicode bytes. We investigate extensions of this effective framework, encoder-decoder networks, that handle multiple sequences at the same time. This reduces to the problem of multi-lingual machine translation (MLNMT), as we explore and discuss the applicability and benefits of using finer tokens. Further we discuss future directions that are enabled by using finer tokens, from multi-modal processing and multi-task learning perspectives. We finally discuss observed problems and future challenges for multi-sequence modeling with finer tokens.

4.6 Robsut Wrod Reocginiton via semi-Character Recurrent Neural Network

Kevin Duh (Johns Hopkins University – Baltimore, US)

The Cmabrigde Uinervtisy (Cambridge University) effect from the psycholinguistics literature has demonstrated a robust word processing mechanism in humans, where jumbled words (e.g., Cmabrigde / Cambridge) are recognized with little cost. Inspired by the findings from the Cmabrigde Uinervtisy effect, we propose a word recognition model based on a semi-character level recursive neural network (scRNN). In our experiments, we demonstrate that scRNN has significantly more robust performance in word spelling correction (i.e., word recognition) compared to existing spelling checkers. Furthermore, we demonstrate that the model is cognitively plausible by replicating a psycholinguistics experiment about human reading difficulty using our model. (This is joint work with Keisukue Sakaguchi, Matt Post, and Ben Van Durme)

4.7 Inducing Morpho-syntactic Lexicons and Morphological Inflections

Manaal Faruqui (Carnegie Mellon University – Pittsburgh, US)

Morphology of a word can help determine different aspects of its meaning such as tense, mood, voice, aspect, person, gender, number and case. Such morpho-syntactic information about word meaning provides crucial information while training models for downstream NLP tasks. In this talk we are going to discuss two different problems involving morphology. In the first part of the talk I will show how morphological information can be used to construct large-scale morpho-syntactic lexicons for a large number of languages. In the second part of the talk I will show how different possible inflected forms of a word can be generated using encoder-decoder neural network models in a language-independent manner.

4.8 (Neural) Graphical Models Over Strings

Ryan Cotterell (Johns Hopkins University – Baltimore, US)

Natural language processing must sometimes consider the internal structure of words, e.g., in order to understand or generate an unfamiliar word. Unfamiliar words are systematically related to familiar ones due to linguistic processes such as morphology, phonology, abbreviation, copying error, and historical change. We will show how to build joint probability models over many strings. These models are capable of predicting unobserved strings, or predicting the relationships among observed strings. However, computing the predictions of these models can be computationally hard. We outline approximate algorithms based on Markov chain Monte Carlo, expectation propagation, and dual decomposition. We give results on some NLP tasks.

5 Working groups

5.1 Morphology

Kristina Toutanova (Google – Seattle, US)

Joint work of Fabienne Cap, Ryan Cotterell, Chris Dyer, Kevin Duh, Manaal Faruqui, Vladimir Golkov, Adam Lopez, Laura Rimmel, Kristina Toutanova, François Yvon

5.1.1 Introduction

This is a report that summarizes the discussions and project ideas that originated in the morphology working group. Our group had productive discussions on several topics that deal with character-level and subword-level modeling of natural language.

The discussion started with "what is morphology?" or rather what is a universally accepted theory of how words are formed. Most people agreed that probably at some level there exists a morpheme and then inflection generation on the morpheme leads to the creation of new words. However, we did not delve deep into whether this is correct theory or not, as we wanted to discuss more practical problems.

A common agreement in the team was that derivational morphology is often ignored in modeling, even though around 50% of words in English are constructed through derivational processes. An example of this phenomenon is: *unquaffability*, which is composed of *un* + *quaff* + *able* + *ity*. In derivational morphology since the semantic meaning of the word changes, it is more important to have segmentation of the word that can reveal its composition. Derivational morphology is also different from inflectional morphology in the sense that we often need to extract knowledge from the sentence context to be able to analyze the word.

An advantage of character-level models over word or morpheme-level models is that it allows the generation and analysis of out-of-vocabulary words. Using character-level models of course raises the question whether morphology is useful at all practically or it is something that can be ignored. To people whose interest lies in understanding how language works, this is an interesting question. But for those who only care about whether information within a word can help NLP systems, just proceeding with a character-level model seems like the best route.

5.1.2 Research Questions

Now we provide a list of different topics that we came up with for future research. We will describe one particular model in detail in Section 5.1.4.

- **RQ1:** Is there a relation between traditional morphology and character-level representations?
- **RQ2:** Does morphological modeling help downstream tasks?
- **RQ3:** How do we incorporate morphology in neural machine translation?
- **RQ4:** How do we know if the characters of segments in a word are compositional?
- **RQ5:** How do we analyze what segments/characters of the word is the model selecting?
- **RQ6:** How does character-level NLP handle compounding and multi-word expressions?
- **RQ7:** How does character-level NLP handle non-standard morphological processes (e.g., word analogies, phonesthemes (glow-glitter-glide), apocope, portmanteau words (Oxbridge, Bennifer, Brexit), diminutives, etc.
- **RQ8:** Given a word form, can we predict its morpho-syntactic properties?
- **RQ9:** Can we use data with perturbed character sequences within a word to generate more training data and improve model performance?
- **RQ10:** Do character and segment-level models learn the exact same thing?

5.1.3 Analyzing distributed representations

Character-based models deliver distributed representations for words, which are varying dependent on the context of their occurrences. An interesting issue is to try and diagnose the kind of morphological knowledge that is (or not) encoded in these representations.

Expectations regarding these representations are that they should for instance encode:

- Ambiguity (in inflection and derivation): This is important because ambiguity is pervasive in NLP, and can sometimes have some systematic (i.e., in conjugation) nature. Therefore, looking at how ambiguity is encoded in distributional representations has a bearing on what we can expect from continuous representations.
- Another issue is with exceptional patterns: if we look at string patterns, we see regular and exceptional / coincidental patterns, e.g., dearly is dear+ly but early is not ear+ly. Can we detect regularity vs. exceptions in distributed representations? In a similar fashion, can we detect transparent vs. opaque derivatives based on these representations?

In order to evaluate these models and representations, a set of systematic tests, applicable to a wide variety of languages and morphological systems needs to be designed. Possible tasks, along the lines of the recent work by [1, 2] are the following:

- Predict the correct agreement / case markings, for increasing complex (remote) dependencies. This could be implemented in a number of ways, such as compare the likelihood of correct vs. incorrect forms.

  Another interesting test would a be akin to the "wug" test. Given a prefix sentence and the prefix of a word, see whether the model can generate the right suffix. For instance:

  I think that writing this short example really wug?

  If the model can also generate POS tags, we can make the task more interesting by also providing POS information (in the former example how to generate given .... (past,wug)). This kind of exercise should be designed for more inflectionally interesting languages than English.

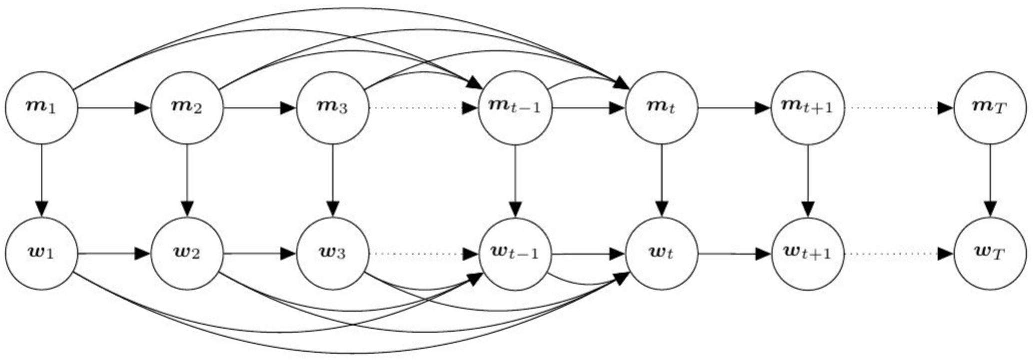

Figure 1 A graphical represention of the joint model.

- Testing derivational knowledge is more challenging. A possible source of inspiration is the definition generation task, which provides a way to explicitly stimulate lexical creativity. Alternatively, one could also try sentence completion exercises with contexts that should both constrain a given POS and a given derivative (if it exists)

  This model is really [Markovian / *Markovable / Markovist / *?Markoving / *Markovability / Markovesque / *Markovism] etc.

  and test phenomena such as morphotactics, morpho-phonological processes (or rather their orthographical expression), such as e.g., vowel harmony, vowel reduction, consonant assimilation, etc).

As regards the evaluation of these representations, various ideas have popped up, such as:

- Correlate the distribution of representation of a word with the number of morphological analyses;
- Evaluate the ability of the representations to identify derivational families;
- Contrast character-based vs. word-based embeddings and see whether they can be made to agree[2]

The next step, if these properties are not satisfied and actually matter for downstream applications, is to try to enforce these properties by jointly learning word sequences and their morphology. Such ideas are developed in the following section.

5.1.4 Proposed Model

In this section, we sketch out a language model that jointly models words ($\vec{w}$) and morphological features ($\vec{m}$). Intuitively, each token has a set of morphological features, and the generative process first selects the morphological features at time t and then, conditional on that and the history of words, selects a word. A graphical representation is in Figure 1.

[2] This might be useful in the following setting: (1) learn word based embeddings for frequent words; (2) train a character-based model to reproduced these; (3) use the resulting character embeddings to generate embeddings for unknown words. This could also be used as a way to speed up the training of character-level embeddings.

$$\begin{aligned}
p(\vec{w}) &= \sum_{\boldsymbol{m}} p(\boldsymbol{w}, \boldsymbol{m}) \\
p(\boldsymbol{w}, \boldsymbol{m}) &= \prod_{t=1}^{N} p(w_t, m_t \mid \boldsymbol{w}_{<t}, \boldsymbol{m}_{<t}) \\
p(\vec{w}) &= \sum_{\vec{m}} \prod_{i=1}^{N} p(w_t, m_t \mid \boldsymbol{w}_{<t}, \boldsymbol{m}_{<t}) \\
p(w_t, m_t \mid \boldsymbol{w}_{<t}, \boldsymbol{m}_{<t}) &= p(m_t \mid \boldsymbol{w}_{<t}, \boldsymbol{m}_{<t}) \times p(w_t \mid \boldsymbol{w}_{<t}, m_t) && (1) \\
&= p(m_t \mid \boldsymbol{m}_{<t}) \times p(w_t \mid \boldsymbol{w}_{<t}, m_t) && (2) \\
&= p(m_t \mid m_{t-1}) \times p(w_t \mid \boldsymbol{w}_{<t}, m_t) && (3) \\
&= p(m_t \mid \boldsymbol{w}_{<t}, m_{t-1}) \times p(w_t \mid \boldsymbol{w}_{<t}, m_t) && (4)
\end{aligned}$$

In Equations (1) and (2), supervised training is tractable, but marginalizing $\boldsymbol{m}$ is intractable. However, the models given by Equations (3) and (4) are reasonable and the forward algorithm can be used to marginalize $\boldsymbol{m}$ which can be used for likelihood evaluation or unsupervised training.

$$\begin{aligned}
p(w_t = k \mid \boldsymbol{w}_{<t}, m_t) &= \text{softmax}_k \left(\mathbf{U} \tanh \left[\boldsymbol{\varphi}_{m_t}; \mathbf{h}^w_{t-1}\right]\right) \\
p(m_t = \ell \mid \boldsymbol{w}_{<t}, \boldsymbol{m}_{<t}) &= \text{softmax}_\ell \left(\mathbf{V} \tanh \left[\mathbf{h}^m_{t-1}; \mathbf{h}^w_{t-1}\right]\right)
\end{aligned}$$

Possible extensions of the baselines:

1. generate words with a character model (instead of the softmax layer). It would be especially interesting to see whether the additional morphological "tier" helps to improve the morphological tasks;
2. add an attention model over past words / past morphs / both;

$$\begin{aligned}
p(w_t = k \mid \boldsymbol{w}_{<t}, m_t) &= \text{softmax}_k \left(\mathbf{U} \tanh \left(\boldsymbol{\varphi}_{m_t} + \sum_{i=1}^{t-1} \beta_i^m \mathbf{h}_i^w\right)\right) \\
p(m_t = \ell \mid \boldsymbol{w}_{<t}, \boldsymbol{m}_{<t}) &= \text{softmax}_\ell \left(\mathbf{V} \tanh \left(\sum_{i=1}^{t-1} (\alpha_i^m \mathbf{h}_i^m + \beta_i^m \mathbf{h}_i^w)\right)\right)
\end{aligned}$$

3. use this as an additional model on the target side for MT tasks.

5.1.5 Comparing Characters and Words

Character-level modeling allows sharing of statistical strength among words that have common character sequences. Word-level modeling integrates an inductive bias towards architectures that construct and reason with representations of words. Prior work has built architectures using these different units of modeling, but has not isolated their impact in controlled comparative studies.

We propose several experiments to evaluate the capacity and practical performance of character and word-based architectures. A simple experiment is to start with two architectures for word representations used in a language modeling task for a simple n-gram feed-forward language model. Architecture W embeds all words using a one-hot representation of words and a second architecture C uses a bi-directional character-level RNN with one or more layers to derive a word embedding as the last layer.

Given a model trained on a given dataset D using architecture W and word embedding size d, theory suggests that an architecture of type C exists that can obtain the same optimal value of the training loss function. Theory does not provide guidance on the dimensionality or number of hidden layers in C and similarly, nothing is known about the training time and difficulty of the two optimization problems.

We suggest to experiment with architectures of growing size in C, to discover patterns in the relationship between dimensionality d in W and dimensionality and architectures in C. We will experiment with multiple layers and hidden unit size in C, a final layer mapping to dimensionality d. CNN in addition to RNN architectures will be interesting to evaluate. For every candidate architecture, we will measure the loss function value obtained, as well as dimensionalities, time, and learning curve.

An alternative to training character-level architectures from scratch is to first train a word-level architecture and then train a character level sub-model which for each word w aims to fit its word embedding v via the character-level sub-network. In theory, this approach would result in a similar value of the loss function (if the word embeddings can be approximated very well), but would be much faster to train since type-level as opposed to token-level updates will be required in training. The authors of [3] have already shown that character-level models are able to fit word vectors well, but have not studied the relative dimensionality and efficiency of the two representations.

References

1 Tal Linzen, Emmanuel Dupoux, Yoav Goldberg: Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. TACL 2016.

2 Rico Sennrich: How Grammatical is Character-level Neural Machine Translation? Assessing MT Quality with Contrastive Translation Pairs. EACL 2017.

3 Ryan Cotterell, Hinrich Schütze, Jason Eisner: Morphological Smoothing and Extrapolation of Word Embeddings. ACL 2016.

5.2 Machine Translation

Parnia Bahar (RWTH Aachen, DE)

Joint work of Parnia Bahar, Ondrej Bojar, Desmond Elliot, Orhan Firat, Alex Fraser, Georg Heigold, Jan Niehues, Laura Rimell, Helmut Schmid

Machine translation (MT) is perhaps the biggest success of deep learning in NLP. This working group was concerned with research questions and challenges for character-level MT. In particular, the working group dealt with three issues in Neural Machine Translation (NMT) [1, 2]: appropriate units for an NMT system; efficiency of NMT with different units; and how to treat multilinguality and multiple modalities.

5.2.1 Appropriate Units for NMT

Since the vocabulary size needs to be shortened to the most frequent tokens in word-level NMT, it suffers from out-of-vocabulary (OOV) problems. Possible solutions are subword units or characters as the atomic units of translation.

Byte Pair Encoding (BPE) [3] is a prominent approach which segments words into sequences of subwords. The most frequent words are kept in their original forms while the rare words are divided into pieces. This appproach is more effective than using words. However, it requires purely offline segmentation in advance and the number of merges needs to be predefined. An alternative idea would be to learn the merge operations inside the network for machine translation. Another solution which does not require explicit segmentation or word boundaries is character-level NMT.

Although using character-level NMT reduces the computational complexity in the softmax operation and can handle the OOV problem, several requirements and challenges need to be considered. If the source and target sentences are encoded in characters, the sequence length is longer. Hence, for each target token, the decoder needs to attend to all characters in the source sequence. This can lead to an extreme computational time in the attention layer. One solution to overcome this problem is to have a shorter source representation. The authors of [4], for example, propose a convolutional neural network with different filter sizes and max-pooling to have a shorter representation in the encoder. In particular, a representation for every five character positions is created and then forwarded to a highway network and the encoder, which is a bidirectional gated recurrent neural network. Thus, the network calculates features on character n-grams including cross-token n-grams. The working group agreed that it should be possible to get slightly better results without max pooling if there is enough training data and time available.

Character-based models provide advantages for multilingual models. However, they require a shared idea of characters. It is, for example, not clear if the model of [4] would work for both German and Chinese input. Coming up with a good BPE inventory for multilingual input/output is in general difficult.

5.2.2 Efficiency of NMT with Different Units

An advantage of character-based models is the possibility of attending to characters and, therefore, implicitly splitting compounds or words from agglutinative languages. However besides translation quality, efficiency of training and memory plays an important role. Based on experimental results reported in the literature, character-level NMT is three times slower than BPE-NMT. Therefore, utilizing larger strides in max pooling, reducing the number of computations of attention weights, faster convergence by means of momentum-based optimization algorithms and finally reducing the precision could speed up training and make memory usage more efficient. The decoder can be sped up by recomputing attention only after every three characters or by only computing attention for a few relevant positions. For this purpose, the working group proposed to run a recurrent neural network over the input embeddings to compute a relevance score (sigmoid output) for each position. Closely located high relevance scores are penalized (for example, by adding a sum of pairwise products of scores to the loss function). This additional constraint should force the model to identify a small set of highly relevant positional embeddings. The machine translation decoder can then only attend on the relevant positions. Having only a small number of relevant positions speeds up the decoder attention considerably.

5.2.3 Multilinguality and Multi-Modality

In the last session, the working group discussed how to use two input modalities, such as a character-level text and an image. An example for image-to-text decoding is the image captioning task. Because sharing information between character-level text, like the plural morpheme “+s” and a picture of two cats is difficult, the group first considered two-source character-level translation as in multilingual systems. The BLEU score of an NMT system which is trained on two sources (French and Spanish) for translating into English is considerably higher than the BLEU scores of two single-source systems. The working group assumed that this implies that the NMT systems learns a shared representation (something like an interlingua). Alternatively, it might just be an ensembling effect of two co-existing systems, leading to improvements because more target side data was seen effectively. To test whether anything is shared or not, the working group created two setups. The first one is to train two separate NMT systems: French→English and Spanish→English on the same set of English sentences and apply multi-source translation at test time (by ensemble decoding). In the second setup, there are two encoders (one for French and one for Spanish) and one shared decoder for English, trained by alternating samples. If the second scenario works better than the first one, there is some sharing in the joint decoder. This would be beneficial, because it means that one can train two separate encoders (for different modalities, such as character-level text and pictures) with a single shared output decoder. For such a multi-modal system, the same test could be applied to figure out whether there is sharing between the two modalities. An alternative approach for training shared representations is an adversarial setup which includes a second objective in order to make it hard to tell from which encoder the representation came.

5.2.4 Research Questions

The working group came up with the following research directions:

- **RQ1:** What is a proper translation unit in terms of performance, efficiency and availability of reasonable computational resources?
  - Possible units: characters, BPE, subwords, words, phrases, ...
  - Investigation of BPE (offline segmentation) vs. character-based (online segmentation) models.
    The space of possible offline segmentations is huge.
  - Reduction of BPE size towards characters to get a better generalization.
    Sennrich, for example, showed that a 2-gram character consecutive segmentation plus a word short list work well. [3]
  - For efficiency, it might be better to have a hybrid representation instead of a fully character-based one.
    An important question is what sort of hybrid representations are possible. A possible hybrid would be BPE + characters in an encoder-decoder network with two attention layers.
  - Comparison of character-based word representations and extra-token character-n-gram representations
  - Is it possible to predict bytes or even bits?
  - Investigation of multiple encoders at different levels (on words, BPE, characters).
    A two-level attention mechanism might be used which first decides which encoder to use. Another possibility is the concatenation of the results of the different encoders. Techniques from multi-source translations might be employed as well.
- **RQ2:** How can we inspect models?
  - An interesting analysis would be the comparison of attention at character level and attention at segment level.
  - Is it possible to use attention on character-to-character models to get BPE-like units (for evaluating character-to-character models)?
- **RQ3:** Can we use large amounts of monolingual source data to improve the encoder?
  One idea is to have multiple decoders with a shared encoder in a multi-task setting.

- **RQ4:** How to investigate whether there is sharing between a character-level text encoder and a picture encoder for image caption translation?
  Is there a good representation like the red box proposed by the representation-learning working group?

References

1 Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio: Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015.
2 Ilya Sutskever, Oriol Vinyals, Quoc V. Le: Sequence to Sequence Learning with Neural Networks. NIPS 2014.
3 Rico Sennrich, Barry Haddow, Alexandra Birch: Neural Machine Translation of Rare Words with Subword Units. ACL 2016.
4 Jason Lee, Kyunghyun Cho, Thomas Hofmann: Fully Character-Level Neural Machine Translation without Explicit Segmentation. arXiv 2016.
5 Junyoung Chung, Kyunghyun Cho, Yoshua Bengio: A character-level decoder without explicit segmentation for neural machine translation. ACL 2016.
6 Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, Koray Kavukcuoglu: Neural Machine Translation in Linear Time. arXiv 2017.
7 Jonas Gehring, Michael Auli, David Grangier, Yann N. Dauphin: A Convolutional Encoder Model for Neural Machine Translation. ICLR 2017.

5.3 Representation Learning

Yadollah Yaghoobzadeh (LMU München, DE)

Joint work of Heike Adel, Phil Blunsom, Fabienne Cap, Vera Demberg, Chris Dyer, Jan Hajicč, Karl-Moritz Hermann, Adam Lopez, Sien Moens, Martin Schmitt, Hinrich Schütze, Yadollah Yaghoobzadeh

5.3.1 Introduction

Starting from scratch and doing end-to-end learning for each individual task is not effective. In end-to-end NLP, we need to start from characters/bytes and therefore lots of training data is required. However, sparsity is always a reality. Therefore, we need to somehow transfer our knowledge (specifically, the generic knowledge) from one task to others. Representation learning is one way to do so: we learn a generic representation of the input and use that in the end tasks. We argue that segmentation of inputs to the sequence of characters/bytes makes representation learning ineffective.

A representation should contain all general linguistic knowledge, including morphological, syntactic, semantic, sound/phonetic, orthographic, distributional and discourse knowledge.[3] Then, for the specific NLP tasks, mappings are learned on top of these general representations. The advantage of representations is that less training data is necessary for a target task. However, there are also some caveats, e.g., how to fit coreference resolution and discourse dependencies, even though these require general linguistic knowledge.

There are some important (unanswered) questions about representation learning in NLP: (i) What is the level of granularity? (From morphemes to idioms). (ii) Are overlapping units

[3] The possibility of extraction of world knowledge from text is limited; e.g., “black sheep” is far more common than “white sheep” in text.

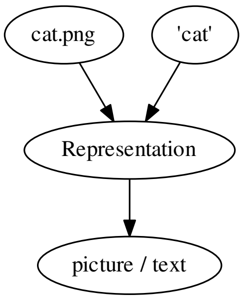

Figure 2 Multimodal representation.

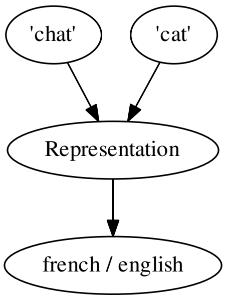

Figure 3 Multilingual representation.

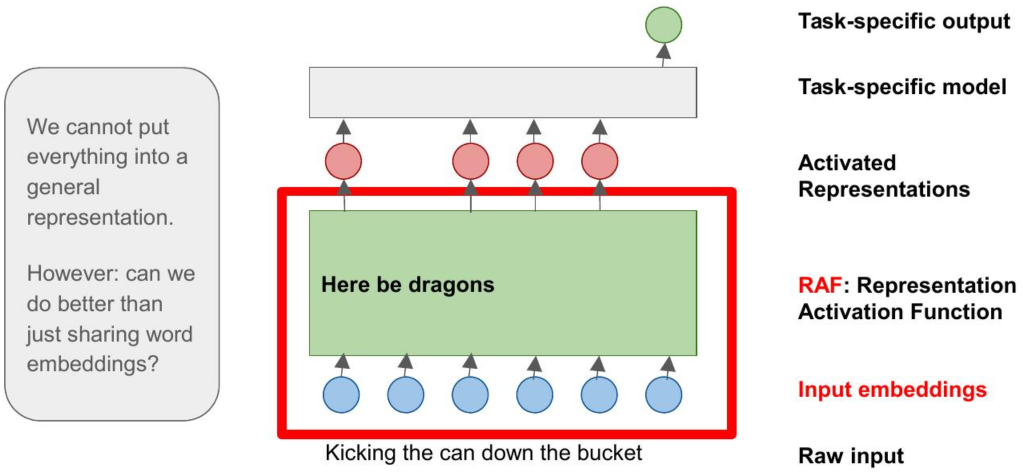
We cannot put everything into a general representation.
However: can we do better than just sharing word embeddings?
Here be dragons
Kicking the can down the bucket
Task-specific output
Task-specific model
Activated Representations
RAF: Representation Activation Function
Input embeddings
Raw input

Figure 4 Schematic diagram of The Red-Box.

fine? Example: "seven", "heaven" and "seventh heaven". (iii) What should be the content of the representations? Example types of content: phonetic similarity, visual similarity (radicals, strokes, pixels, bits), Affordances, physical properties, (iv) Are the contents of representations based on the target use case? (v) Should we use better learning objectives for representations? (vi) How can we learn disentangled representations? (vii) How to trade off memorization against generalization? (Saussure vs. Frege).

5.3.2 The Red-Box

The motivation and schematic diagram of the model we proposed for representation learning is shown in Figure 4.

There are some key considerations in modeling the Red-Box: (i) Activated representations (AR) = standard embeddings + some context + some syntax + some semantics. (ii) The aim is to facilitate learning task-specific models. (iii) Is the representation activation function (RAF) equivalent to the NLP pipeline? (iv) Training: multi-task setup with joint core up to AR. (v) RAF evaluation objective: minimize sample complexity for downstream tasks. (vi) Duplication / Parameter Sharing: "In seventh heaven" is a unit that we can have an embedding for. But to understand "the highest of seven heavens according to Islam

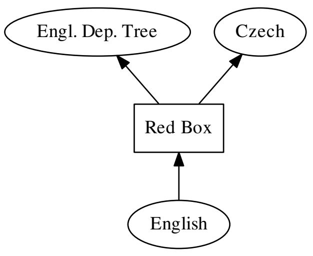

Figure 5 Example tasks for the Red-Box: dependency parsing and machine translation.

and Judaism" we need access to the notion of seventh heaven downstream. In traditional approaches, all relevant info is in one place (the lexicon). In contrast in the Red-Box model, knowledge is distributed over two places: embeddings & RAF. How is knowledge shared between embeddings and RAF?

5.3.3 Research Questions

Apart from theoretical considerations, there are also some practical challenges and research questions that need to be addressed:

- **RQ1: How to train the Red-Box?** (i) on which tasks? (ii) with which kind of weighting regime and (iii) how to avoid catastrophic forgetting?
  We need to choose the tasks that we want to train the Red-Box on. Should these tasks be multi-modal, or just text-based?
  - Example tasks A (see Figure 5). Task 1: dependency parsing, an English sentence as input and the English dependency parse of the sentence as output. Task 2: machine translation, an English sentence as input and the Czech translation as output. The Red-Box possibly benefits in this scenario as follows: Task 1 helps identify "grandma" as object of "take care" in "he took care of grandma". Task 2 (transfer task) can then learn more easily that "grandma" needs to be generated in the accusative case.
  - Example tasks B: Task 1: machine translation, a Finnish sentence as input and the English translation as output. Task 2 (transfer task): coreference resolution, a Finnish sentence as input and the resolved coreference chains as output. The Red-Box possibly benefits in this scenario as follows: Task 1 should learn coreference resolution since it is required for correct Finnish-to-English translation. Task 2 just has to tap into what Task 1 has already learned.

  To avoid catastrophic forgetting, we might need to include some sort of memory in the Red-Box – Human memory prevents forgetting.
- **RQ2: What information should be in the output of the Red-Box?** How to draw the line between core linguistic competence and non-linguistic information? What about: common-sense knowledge, world knowledge, logical inference? And: are vectors going to be the representation of outputs? Vectors are the "universal language" of deep learning, so an easy combination of modules is possible with them. However, they are hard to interpret.
- **RQ3: How can the Red-Box and its output be interpreted?** There are two questions in this regard: (i) Which parts of the sentence are responsible for a particular output? (ii) What is the interpretation proper of the output? There are some possible ways: (i) Nearest neighbors; (ii) Attention; (iii) Train special "task" modules that produce an

interpretable output; (iv) If we train the Red-Box on machine translation, how do we find out what it has learned? Maybe it has not learned anything interesting?

- **RQ4: For which tasks can the Red-Box be used?** There are more open research questions that need to be discussed: (i) Which tasks can a Red-Box trained on machine translation perform well for? Question answering, textual entailment, inference? (ii) Which combination of tasks should the Red-Box be trained on, so that it is usable across a broad range of applications? (similar to the classical NLP pipeline).

5.4 End-to-End

Heike Adel (LMU München, DE)

Joint work of Heike Adel, Jan Hajič, Thomas Hofmann, Hang Li, Hermann Ney, Ngoc Thang Vu

5.4.1 Introduction

There are several interpretations of the term "end-to-end", such as "trainability of the system as a whole", "joint training of all system components" or "training without explicit feature design". While the term "end-to-end" is probably young, the concept of end-to-end learning is more than 20 years old although back then people refered to it rather as a joint learning of modules. While the computational complexity of such a system was too high in the past, current studies are getting closer to building "real" end-to-end systems.

5.4.2 What are the "Two Ends"?

One characteristic of end-to-end systems is that the whole system is trained as a whole – from the processing of the input to the prediction of the output.

The typical input to end-to-end systems is raw data. While in speech, this corresponds to the acoustic signal or spectral features, and computer vision uses the pixels of an image, the question in NLP is whether to use words, characters or bytes. Mixtures of those are also possible. The working group agreed that using pre-trained embeddings to represent the input does not contradict the notion of end-to-end. One particular possibility is the joint (hybrid) training of embeddings and the final task. Especially due to continuous vector representations in neural network and end-to-end training, it is also possible to create representations for inputs of different modalities in the same continuous space. However, this increases the complexity of the model. Slightly different to multi-modal is cross-modal training. In this case, the modality of the input is different to the modality of the output. An example is the generation of an output in a different modality (such as image captioning). One research question in this area is how to integrate a prior of the target modality into the end-to-end model (such as language model probabilities for image captioning).

For training end-to-end models, the correct choice of the objective function is crucial. The objective function should be related to the end performance / final application to be able to train every system component jointly. However, the end performance might be hard to express mathematically or might not be differentiable. An example is a machine translation system that should be used by human beings. The reaction of humans to the system and its output can differ a lot from typical machine translation metrics, such as the BLEU score. In training end-to-end systems, there might also be multiple objectives. For example, an intermediate layer could be trained to predict part-of-speech (POS) tags

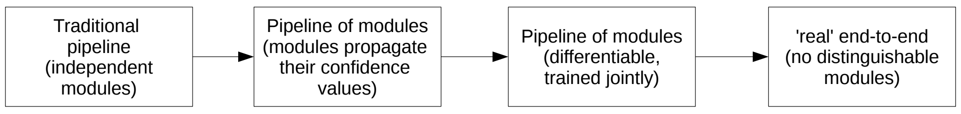
Traditional pipeline (independent modules)
Pipeline of modules (modules propagate their confidence values)
Pipeline of modules (differentiable, trained jointly)
'real' end-to-end (no distinguishable modules)

Figure 6 Evolution from pipeline systems to end-to-end systems.

using a side-objective in order to bias the model to learn POS tag-like features. This allows integrating linguistic or domain knowledge in a differentiable way into an end-to-end system. It can improve the final performance and lead to a faster convergence of the system.

5.4.3 End-to-End Systems

End-to-end learning refers to the joint training of individual system components. There is an important tradeoff between end-to-end training and model interpretability. Between traditional pipeline systems and "real" end-to-end systems, there are other hybrid possibilities. Figure 6 shows the evolution from pipeline systems to end-to-end systems. In traditional pipeline systems, each system component (module) is independent from the other modules. In a first hybrid step, the pipeline modules are still independent but propagate their confidence values to the following modules. Between these types of systems and "real" end-to-end systems which might not have distinguishable modules any more, there are pipeline systems with different modules but all modules are differentiable and can be trained jointly.

Due to the complexity of end-to-end systems, a good implementation is necessary. There is statistical efficiency (i.e., how many training examples a system needs until it gets reasonable results) and computational complexity (i.e., how and after which training time the optimal parameters are found). In contrast to the past, it is now possible to apply backpropagation training through the whole end-to-end system.

The working group discussed specific deep learning models and their strengths and weaknesses in terms of learning long-distance dependencies (which is a crucial challenge in text processing, especially when the input is a character sequence). While recurrent neural networks (RNNs and especially LSTMs) are more powerful, also for modeling long-range dependencies, they are also harder to train and computationally more expensive. While LSTMs actively manage their memory (hidden layers), attention is a recently introduced mechanism that avoids the need of a memory. Convolutional neural networks (CNNs), on the other hand, are good at (low-level) feature extraction (they are usually applied in combination with fully connected layers afterwards) and computationally more efficient but they have limitations in capturing long range dependencies.

Hierarchical models have both advantages and disadvantages: Since documents are also built hierarchically, hierarchical models might be appropriate for modeling documents. However, processing a document without a hierarchical structure could factor out syntax and help focus on the semantic meanings.

5.4.4 Research Questions

The working group proposed the following set of research questions:

- **RQ1: Interpretation of results:** When papers report improvements for a specific task, it is central to show whether those improvements are also reproducible on other tasks and/or domains. It is important to see whether the improvement comes from a superior model or rather from a different (and maybe larger) training data set. In fact, systematic errors in the model or architecture might only be recognizable if the amount of training

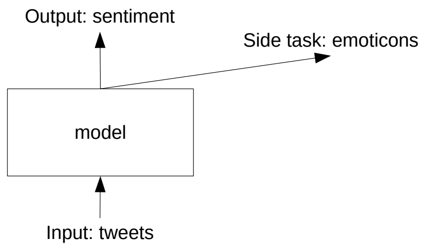

Figure 7 Example for data augmentation.

data is very large (infinite). This means that a more careful design of experiments as well as a detailed error analysis is necessary. A possibility to judge end-to-end models might be shared tasks because in shared tasks the competing systems are trained and evaluated on exactly the same dataset. Thus, direct comparisons of models are possible.

- **RQ2: Understanding the models:** Deep learning models and especially end-to-end models are often hard to interpret. Understandable models might be structured in a way that intermediate representations have meanings. For example, a model could have an intermediate layer that is trained to learn POS tag-like features. Similar to analyzing the results and errors of a model (see RQ1), the model itself should be inspected, too. Attention provides a straightforward way of visualizing attention weights. The question is whether this is enough to understand the model. Would it be possible, e.g., to visualize connections between words that are far away from each other in the input? In LSTMs, it is possible to analyze how the gradients flow through the different intermediate layers. In CNNs, most analysis focuses on extracting the most important n-gram features. An open question is whether these n-gram features are generalizable to other tasks and domains. In general, visualizing models or model components always comes with the theoretical question of what we actually expect.
- **RQ3: Integrating knowledge into end-to-end systems:** When training end-to-end systems, the system should learn everything by itself, i.e., without explicit feature design. Nevertheless, by designing the model, we implicitly use (and should use) knowledge about the task: For designing the input, it is crucial to know which information is needed for the task. When creating the model structure, we use our knowledge by deciding how the different modules should interact. For coming up with an appropriate objective function, it is important to know which function is most closely related to the application. That means although there is no explicit feature design in end-to-end learning, the models still reflect domain knowledge. Especially end-to-end systems are hard to train with little data. On the other hand, getting/annotating data can be very expensive. A cheap way of obtaining additional data for end-to-end systems in order to train them more robustly, is data augmentation. The concept is similar to transfer learning of domain information or multitask learning. Figure 7 shows how a side task can be used to better train a model for an application. In this example, the input to the model is tweets and the task is the prediction of sentiment. Beside the labeled data for the task (which might be little), there is a huge amount of unlabeled tweets available. Therefore, the model is extended with an additional output layer which predicts the emoticons in the unlabeled tweets. This allows a more robust training of the model part that creates a representation of the input tweet. The data augmentation process is another way of integrating domain knowledge since the selection of the side task needs a good understanding of the main task and domain.
- **RQ4: Adaptivity of end-to-end systems:** The adaptivity of end-to-end systems, e.g., to new domains is a very important challenge. An example would be self-driving cars in a

new city. A particular problem of end-to-end models is that they are very specialized to a particular task and domain. Furthermore, it is arguable whether there are still individual modules inside an end-to-end system that can be re-trained or exchanged with modules for the new domain (similar to "plug-and-play"). Another question that arises is how efficient adaptation can be ensured. A possibility might be defining and measuring an "adaptation distance". When re-training the whole end-to-end system for the new domain, it would be interesting to analyze which parts of the network have changed because of the adaptation (see RQ2).

5.5 Dialogue

Heike Adel (LMU München, DE)

Joint work of Heike Adel, Karl Moritz Hermann, Hang Li, Hermann Ney, Kristina Toutanova

5.5.1 Introduction

There are different variants of dialogue systems. The most important categorizations differ between task-independent vs. task-dependent systems, between single-turn or multi-turn dialogues and between retrieval-based vs. generation-based systems. Examples for task-independent models are chat bots. Task-dependent dialogue systems are, for example, question answering systems. The answer can be either found in structured data, e.g., templates, databases, knowledge graphs, or in unstructured data, e.g., a forum.

Character-based models are a great opportunity to improve dialogue systems since they provide the possibility of generating unseen words.

There are different data sets available in different languages [1]. STC [2], for example, consists of Chinese single-turn dialogue data from Weibo. Ubuntu [3] is an English multi-turn dialogue data set from a chat room. Reddit [4] contains English comment data from Reddit. Unfortunately, there is usually not enough data to train robust systems.

5.5.2 Research Topics

The working group formulated the following research topics:
- Sample Complexity
- Robust Question Answering
- Multi-source Question Answering
- Inference in Question Answering

References
1 Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, Joelle Pineau: A survey of available corpora for building data-driven dialogue systems. arXiv preprint (2017). arXiv:1512.05742
2 http://61.93.89.94/Noah_NRM_Data
3 https://github.com/rkadlec/ubuntu-ranking-dataset-creator
4 https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicy_available_reddit_comment

6 Panel discussions

6.1 Panel discussion: Character-based models

Heike Adel (LMU München, DE)

Moderator: Manaal Faruqui
Participants: Ryan Cotterell, Kristina Toutanova, Chris Dyer, Jan Hajič, Phil Blunsom

The decision whether to model words, characters, bytes or even bits is often an engineering question depending on the bias of the models and the amount of available data. While characters are a part of our language, bytes have been defined by humans. Since there is a deterministic mapping between bytes and characters, it is a purely engineering decision whether to learn that mapping or not. While it might be possible to capture some additional signals or regularities when modeling bytes (e.g., capital letters have a fixed distance to their lowercase counterpart), going beyond that (and, e.g., modeling bits) might not be advantageous in general but could help for special tasks like pronunciation modeling.

Since we cannot give our models access to everything, it is useful to help them with linguistic information. (There might be an analogy to the epicycle history: Before the observation that the sun was in the center, it was very hard to predict the planet movements.) Linguistic features could, for example, be learned with multitask learning (as a side-knowledge for the main task) and they might also be useful to better interpret the model after training. However, it is an important question which kind of linguistics the model should learn: linguistic theories? Some notions of morphology? Or just something like tokenization? A promising solution could be to focus on phenomena described in most linguistic theories, i.e., phenomena most linguists agree on.

Modeling morphology is an important part of natural language processing/understanding. Sequences of characters do not capture all information. So, external knowledge about morphology can help the model to solve its task. While morphology can be useful from the empirical and engineering point of view, it is also an important theoretical topic (for example, to answer questions like how children learn morphemes).

For short-term artificial intelligence, end-to-end models are the quickest solution. However, for long-term solutions, multitask and transfer learning techniques will get more important because there is not enough data for end-to-end learning in general. Transfer learning is similar to how children learn: They learn something from task A and then leverage it for task B. Therefore, it is essential to separate representation learning and language understanding from specific applications. An example for successes of transfer learning are zero-shot translations. While deep learning facilitates the transfer of representations (even across modalities), getting it to work is not straightforward. For example, it is not clear which tasks we should use to learn to understand language and to figure out whether the models have learned anything about language or linguistics. For example, if the models just follow the traditional NLP pipeline, they do not learn anything about language but just about our own understanding of it. In fact, models should be able to induce such kind of knowledge as in, e.g., alignment learning in neural machine translation.

The prospects of sharing character-level encoders among tasks might be task dependent. However, at the character or morpheme level it should be possible to learn something general and task-independent. A character-based model that is trained across tasks can benefit from a large amount of training data. Examples are the success of multilingual models.

6.2 Plenary discussion: Where do we go from here?

Heike Adel (LMU München, DE)

Moderator: Adam Lopez

So far, we only know empirically that transfer learning works (example: word embeddings) but we do not have a theoretical basis for this (yet). It is an open question whether it is provable at all. There are similarites to semi-supervised learning for which we know criteria when it works and when it does not work. Transfer learning could be analyzed in a similiar way. A reason why transfer or multitask learning works in machine translation is that it prevents the decoder from only doing language modeling: Instead, the focus shifts more towards considering the source language. As a result, the system learns when to consider the source and when to consider the target language.

Transfer or multitask learning can cause "catastrophic forgetting". To mitigate this problem, examples from every task should be put into each batch and the weights should be updated based on the gradients gathered from all of them. This heuristic is used in machine translation research.

An open question is how to find out whether a model has gathered linguistic knowledge. In contrast to vision, analyzing the internal features of a neural network is hard for language. Possible ways to answer this question could be (i) to do intrinsic queries (e.g., the model should have learned that "glasses" refers to a single object although it is a plural word), (ii) to generate text based on the internal representations, (iii) to alter the input in a systematic way, (iv) to permute the predictions and differentiate them with the input (to see what is forcing the prediction and where the gradients come from), or (v) to break down a big task (such as neural machine translation) into smaller incremental tasks and analyze the neural network on those.

In general, there is a lot of data available. Rather than needing more data, we need more tasks and automatic evaluation methods.

Especially in the context of end-to-end learning, there are different objectives, such as language engineering vs. gaining linguistic insights, or building intelligent agents vs. understanding the human brain / language / behavior.

Participants

- Heike Adel
LMU München, DE
- Parnia Bahar
RWTH Aachen, DE
- Phil Blunsom
University of Oxford, GB
- Ondřej Bojar
Charles University – Prague, CZ
- Fabienne Cap
Uppsala University, SE
- Ryan Cotterell
Johns Hopkins University – Baltimore, US
- Vera Demberg
Universität des Saarlandes, DE
- Kevin Duh
Johns Hopkins University – Baltimore, US
- Chris Dyer
Carnegie Mellon University – Pittsburgh, US
- Desmond Elliott
University of Amsterdam, NL
- Manaal Faruqui
Carnegie Mellon University – Pittsburgh, US
- Orhan Firat
Middle East Technical University – Ankara, TR
- Alexander M. Fraser
LMU München, DE
- Vladimir Golkov
TU München, DE
- Jan Hajič
Charles University – Prague, CZ
- Georg Heigold
DFKI – Kaiserslautern, DE
- Karl Moritz Hermann
Google DeepMind – London, GB
- Thomas Hofmann
ETH Zürich, CH
- Hang Li
Huawei Technologies – Hong Kong, HK
- Adam Lopez
University of Edinburgh, GB
- Marie-Francine Moens
KU Leuven, BE
- Hermann Ney
RWTH Aachen, DE
- Jan Niehues
KIT – Karlsruher Institut für Technologie, DE
- Laura Rimell
University of Cambridge, GB
- Helmut Schmid
LMU München, DE
- Martin Schmitt
LMU München, DE
- Hinrich Schütze
LMU München, DE
- Kristina Toutanova
Google – Seattle, US
- Ngoc Thang Vu
Universität Stuttgart, DE
- Yadollah Yaghoobzadeh
LMU München, DE
- François Yvon
LIMSI – Orsay, FR