Research Statement

Ioannis Konstas

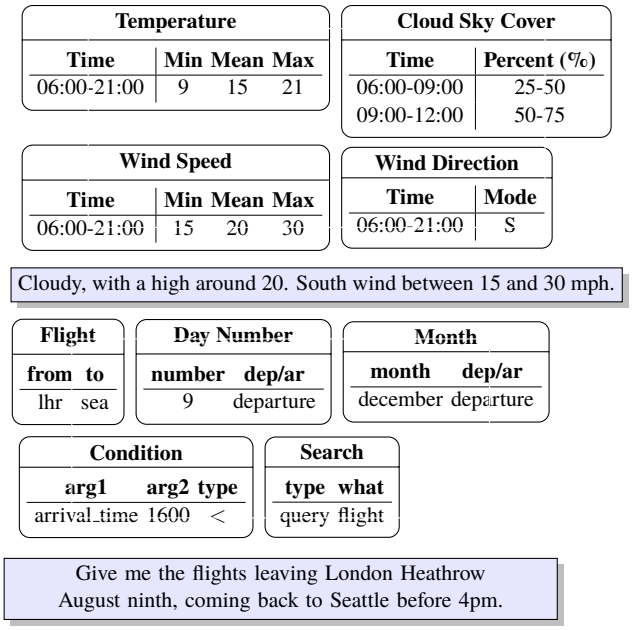
My research focuses on building programs that automatically generate text from computer-readable input. Current applications are restricted to tasks such as giving us the weather forecast, a list of items in our daily agenda, or trying to answer factoid questions. However, we also want the machine to give us instructions on troubleshooting our computer, and assist us in authoring source code by explaining the functionality of complicated snippets. We want to build assistive technologies for educators to automatically create engaging homework for students. *Ultimately*, we would like to create an intelligent machine, that can maintain an interesting and rich conversation with us, humans, and be able to author long, creative and captivating stories, movie scripts and poems.

The majority of existing Natural Langauge Generation (NLG) systems use hard-wired rules or templates in order to capture the input of every different application, rely on small manually annotated corpora, and make simplifying assumptions when generating multi-sentence documents. I advocate end-to-end NLG systems that can **adapt** to multiple domains, **scale** to large corpora and complex inputs and model **global coherence**. In my thesis [1], I proposed the first fully-trainable model that made no assumption on the input domain, did not rely on annotated corpora, and jointly captured the discourse structure of the generated text. In my postdoctoral research, I introduced new neural network architectures that scale on very large corpora and can handle complex inputs, such as source code and meaning representations; we have built the first code-to-language summarizer [2], trained on a large dataset collected from online forums. My work on NLG draws attention on:

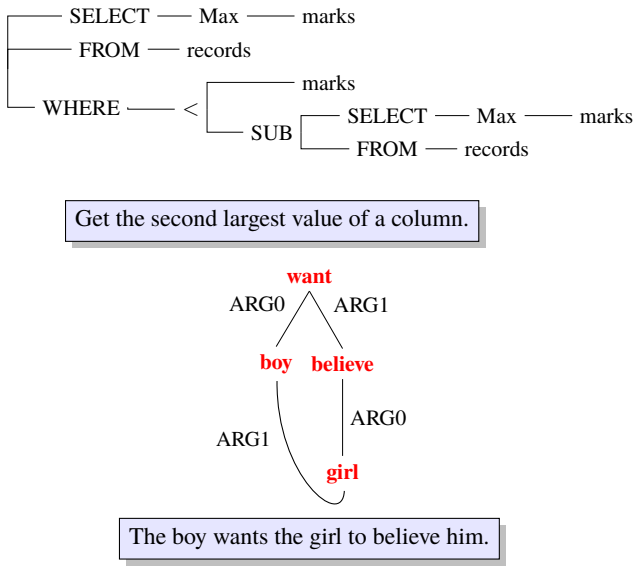
1. **Building adaptable systems using joint models trained on unannotated data and deep learning**. My contributions have emphasized on creating fully-trainable end-to-end models that capture the structure of the document, select parts of the input, and output text, jointly. Given a parallel corpus of input-output examples with no extra annotation, my systems learn to generate text for new domains, ranging from sportscasting and flight booking systems, to weather forecasts and troubleshooting guides. I have also designed neural network architectures that can generalize to complex inputs, such as source code and meaning representation graphs. In my work, I have shown that portability is a corner-stone feature for a low maintenance, highly-reusable NLG system, and have developed approaches to achieve it.
2. **Creating large scale datasets and algorithms that train efficiently**. To this end, I have developed models that are able to train on very large noisy corpora, e.g., collected from community-curated websites such as Stackoverflow or Yahoo! Answers. The datasets can contain input representations that are quite distant from the corresponding outputs, e.g., multiple lines of source code and the compact summary of their function. They can also have a complicated input structure, for example multiple nested subqueries in a SQL query, or re-entrant nodes in a meaning representation graph. In order to be able to adapt to challenging real-life domains, we need to design trainable models that can scale to large and heterogeneous corpora.
3. **Modeling global coherence of the generated document**. I explored the possibility of directly modeling the sentences in a document, and capture the inter- and intra-relations of parts of the input in them. I developed a model that extracted patterns of multiple input records that described, for example weather-related events, and generated a detailed multi-sentence weather forecast. I also investigated techniques that mimicked the structure of human-authored documents, by taking a story, such as a math word problem and rewriting it to a novel one using specific language from a theme such as Star Wars, so that it is more engaging to students, but crucially maintaining the relationship between entities mentioned in the original. Accounting for the discourse structure is crucial for generating output that departs from a simple concatenation of multiple sentences, and approaches a coherent document.

1 Adaptability: Trainable end-to-end Natural Language Generation systems

NLG input data can occur naturally in different forms, as shown in Figure 1. A simple approach is to design a custom algorithm for every domain; this has the advantage that we can then very accurately determine rules to pick parts of



(a) Tabular input for weather report and flight booking generation systems, and their outputs.



(b) Tree and graph-based inputs to a code-to-language and a meaning representation generation system, and their outputs.

Figure 1: Typical inputs and outputs to NLG systems.

the input, and templates to generate the resulting text. There are two main problems though: (a) for domains that have a similar input format type (Figure 1a), we would be unnecessarily wasting effort to fine tune different selection rules, e.g., for wind speed events, or the departure/arrival value of the month of a flight, and (b) for domains that have a nested open-vocabulary input space (Figure 1b), e.g., the Abstract Syntax Tree of a SQL query can contain nested sub-queries over an unbounded number of tables and fields, we would need to spend considerable amounts of time to construct general enough grammars, in order to achieve sufficient coverage.

Joint Modeling for Concept-to-Text Generation For domains that can be expressed in a tabular format as shown in Figure 1a, I developed a system [3] that models content selection and surface realization with a domain-agnostic trainable Probabilistic Context-Free Grammar (PCFG) that hierarchically captures the correspondences between the structure of the input tables and the facts therein, and the resulting text. The benefit of using a generic PCFG is that generation can then be recast as a common parsing problem, and be solved using existing techniques; I formally proposed a variant of the CKY algorithm for generation [4]. The generator exhibits considerable robustness across three domains, namely sportscasting for simulated soccer games (RoboCup), weather forecasts, and user utterances for flight booking systems (ATIS) [1], achieving 2-8% relative increase for two out of the three domains over previous approaches. My system easily allowed for further exploitation by introducing global features over the input [5], and long-range dependencies on the generated output text [6, 1], attaining further gains in performance. Crucially, I showed that using a grammar that essentially *describes* the generation process, we can re-train and generate for several domains.

Representing the input using Neural Networks Some domains inherently represent the input to an NLG system with a more nested structure (Figure 1b), and therefore would require considerable engineering to port them in a tabular format. I have been investigating deep embedding representations that attempt to directly encode the input with minimal intervention to the original structure, using Neural Network architectures. We have developed CODE-NN [2], a sequence-to-sequence neural generator with attention mechanism that takes as input raw code snippets and produces a short one-line summary. CODE-NN learns word embeddings for the input tokens, currently observing them as a bag of

words, while ignoring order and tree structure, and uses an attentive neural layer to bias the surface realization process towards the most important tokens, hence performing content selection at the same time. We tested our generator on two languages, namely C# and SQL, and achieved up to 45% relative increase compared to competitive baselines.

I am currently studying the problem of generating from open-vocabulary graph-based Meaning Representations, such as the Abstract Meaning Representation (AMR) corpus, which comprises sentences from multiple domains (e.g, newswire, forums) and their formal meaning representations (second example in Figure 1b). The architecture of the generator is also a sequence-to-sequence decoder using attention. Compared to source code, the context of every node in the graph plays a more significant role, hence I am exploring richer encodings of the input: I assume an in-order traversal of the graph, and feed every token in sequence through a multi-layer bidirectional Recurrent Neural Network (RNN); the hidden state for each token contains information from the neighboring tokens coming from both directions. This RNN-encoding outperforms a simple bag-of-words representation by 20%. Essentially, the model learns first to associate small sub-graphs rather than individual tokens in the input, with small utterances in the output, and then progressively compose and re-order them to generate the final sentence. Therefore, the model is able to generate complex and long sentences with many concepts intertwined, such as those found in news articles and literature.

2 Scalability: Wide-coverage Neural Natural Language Generation

So far I have been advocating the utility of general purpose learning approaches that can adapt across multiple domains and over arbitrarily complex input structures. Such models typically require significant amounts of training data. However, existing datasets for NLG are low-resourced with only a handful of manually-annotated parallel examples. Therefore we are posed with two challenges: (a) creating cost-effective large-scale parallel corpora when possible, or otherwise leveraging the utility of existing large cheap resources to deal with sparsity, (b) building robust algorithms that can efficiently scale on such large datasets.

Curating large datasets for NLG User-moderated forums are often a great source for multi-modal comparable data. In order to train CODE-NN [2], we created a new dataset using data gathered from Stackoverflow, a popular programming help website. We collected about 1 million posts with relevant tags for C# and SQL, filtered those without an accepted answer, created pairs of snippets of code from the accepted answer and the title of each post, and finally cleaned up pairs with irrelevant titles using a classifier trained on a small set of manually annotated pairs. The resulting corpus comprises of 60k snippets–titles pairs for C# and 33k for SQL, i.e., an order of magnitude more data compared to most previous NLG datasets (e.g., RoboCup has 1.5k examples, and ATIS has 5k examples). Interestingly, the corpus contains challenging pairs with structurally complex input snippets and a diverse output vocabulary.

Addressing sparsity via data augmentation Low-resource trained NLG systems usually suffer from sparsity, which translates to a very high out-of-vocabulary rate, and hence the generated output is producing lots of unknown words, is usually very repetitive and ungrammatical. A very effective approach to battle sparsity is data augmentation. The idea is to build a very large (and possibly noisy) synthetic dataset in order to *expose* the model to more input–output pairs. In particular for AMR generation (14k AMR-sentence pairs), I pretrained on up to 20 million randomly selected sentences from the Gigaword corpus, and attained 55% relative increase in performance compared to training on the original dataset. The approach can be generalized, provided we have access to a corpus which is close to the domain language of our original dataset. We can obtain the noisy input either by using an existing semantic parser, or by reversing the generation model, i.e., by feeding the text as input.

Scaling using Neural Networks My older work on concept-to-text generation [1] recast generation as a parsing problem, hence used a variant of inside-outside for training with EM, and an approximate k-best Viterbi search for decoding. Unfortunately, both algorithms have a cubic time complexity, hence have trouble scaling to bigger datasets and decoding very long inputs and outputs. In contrast, for CODE-NN and AMR generation I used neural network architectures, which usually train in linear time and take advantage of implementations on multiple GPUs that can

efficiently batch multiple examples; decoding is greedy, hence linear in time too, or uses a small beam of hypotheses that can also be parallelized.

3 Global Coherence: Modeling Document-level Structure

Current generation models often struggle to produce larger multi-sentence outputs such as paragraphs, narratives, and ultimately conversations, because of the complex interplay of concepts across sentences. NLG systems need to model document structure at least at one of the following levels of abstraction: (a) predict structural elements of the document, such as connective words between sentences, (b) capture the order of sentences of the document, and (c) keep track of events and entity mentions in order to correctly refer back to them in consequent sentences.

Learning document plans I incorporated the document structure for multi-sentence concept-to-text generation by learning a discourse grammar following two approaches: (a) directly extracting rules representing global patterns of record sequences within a sentence and among sentences from the data, (b) learning document plans based on Rhetorical Structure Theory (RST; [7]). Both techniques improved the previous document planning-agnostic model [3] on weather forecasts and troubleshooting guides by up to 9%. The model automatically learnt patterns of common global sequences of records and ways to express them in a full document, which eliminated unnecessary repetitions of information.

Theme rewriting Often educators need to create math word problems, which are coherent stories with clear cues between the plot and the entities narrated therein, and the mathematical operations and values in the underlying equation. At the same time the stories need to be interesting enough, to promote the engagement and hence increase the performance of the student in solving the problem. We developed a NLG system [8] that takes a human-authored problem, e.g., talking about owning horses, and *rewrites* it so that it adapts to a particular theme, e.g., blasting droids from the Star Wars universe. Our process progressively edits the original story, by introducing highly thematized concepts, but crucially maintaining the semantic congruity at the document level with the original problem. For example, if the original problem narrates a story on the concepts of buying chairs and tables, we would like to rewrite it to a story of buying starships and laser-guns. Our approach uses no theme-specific data, therefore can be easily ported to new themes; it achieves indistinguishable performance to humans in terms of thematicity, coherence and solvability.

Future Research

I have outlined my contributions on creating end-to-end trainable NLG systems that can **adapt** to multiple domains, **scale** to complex inputs and large datasets and capture **document-level structure**. I am also interested in pursuing novel areas of research:

Evaluation The majority of NLG systems are evaluated automatically, by measuring the lexical overlap between the generated output and (usually) a single gold text. One caveat is that often semantically correct outputs are unfairly penalized, because of little overlap with the original text. I am interested in designing metrics that aim to account for the semantic compatibility of the output with respect to the input. For example, given the reference text we could automatically create simple questions based on shallow semantic representations, such as "Who did what?", and crowdsource the answers as gold standard. Then we could investigate methods to automatically identify and measure the number of correct –potentially paraphrased– answers contained in the generated output.

Story Generation Most existing NLG systems output text that contains facts or describes some part of the input. Telling a non-factual story, on the other hand, poses many challenges than can lead to fascinating areas of research including: (a) modeling the structure of a narrative and the characters involved, (b) generate a story based on an existing plot or script, and ultimately (c) generating a plot which is interesting and novel, either as a first step or jointly with the resulting story. An interesting approach would be to introduce to the original sequence to sequence architecture an

extra RNN layer that will model directly the sequence of sentences; then we can use an existing script as supervision to train it, or even interpolate with a character model.

Conversational Agents As intelligent devices become more accessible, so will the need for better, longer and more accurate communication between humans and machines. Existing dialogue systems currently fall under two extremes: (a) they are either confined to very small domains, e.g., booking hotels or giving directions and recommendations, (b) they resemble *chat-bots* with limited or no goals to communicate. I am interested in building conversational agents that can address a wide variety of topics, and are able to model and maintain discussion beyond a single response. The ideal system should be able to use sophisticated language, which is in line with my current research on generating from meaning representations such as AMR. It should be able to generate the output incrementally, so as to be more responsive to human interruptions; ideas can be borrowed from my previous work on incrementality of language and semantics [9, 10]. Finally, it should use elements of common-sense knowledge, in order to generate text that does not immediately derive from facts, and ultimately be able to make inference on existing facts in order to produce new.

References

[1] Ioannis Konstas. *Joint Models for Concept-to-Text Generation*. PhD thesis, University of Edinburgh, 2014.

[2] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, Berlin, Germany, August 2016. Association for Computational Linguistics.

[3] Ioannis Konstas and Mirella Lapata. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–761, Montréal, Canada, June 2012. Association for Computational Linguistics.

[4] Ioannis Konstas and Mirella Lapata. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48(1):305–346, October 2013.

[5] Ioannis Konstas and Mirella Lapata. Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 369–378, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[6] Ioannis Konstas and Mirella Lapata. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1503–1514, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[7] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.

[8] Rik Koncel-Kedziorski, Ioannis Konstas, Luke Zettlemoyer, and Hannaneh Hajishirzi. A theme-rewriting approach for generating algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, page To Appear, Austin, Texas, USA, October 2016. Association for Computational Linguistics.

[9] Ioannis Konstas, Frank Keller, Vera Demberg, and Mirella Lapata. Incremental semantic role labeling with tree adjoining grammar. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 301–312, Doha, Qatar, October 2014. Association for Computational Linguistics.

[10] Ioannis Konstas and Frank Keller. Semantic role labeling improves incremental parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1191–1201, Beijing, China, July 2015. Association for Computational Linguistics.