NLP Applications

Muhammad Abdul-Mageed

muhammad.mageed@ubc.ca

Natural Language Processing Lab

The University of British Columbia

Table of Contents

NLP Areas

- Dialogue and Interactive Systems
- Discourse and Pragmatics
- Document Analysis
- Generation
- Information Extraction and Text Mining
- Linguistic Theories, Cognitive Modeling and Psycholinguistics
- Machine Learning
- Machine Translation
- Multidisciplinary
- Multilinguality
- Phonology, Morphology and Word Segmentation
- Question Answering
- Resources and Evaluation
- Sentence-level Semantics
- Sentiment Analysis and Argument Mining
- Social Media
- Summarization
- Tagging, Chunking, Syntax and Parsing
- Textual Inference and Other Areas of Semantics
- Vision, Robotics, Multimodal, Grounding and Speech
- Word-level Semantics

Figure: NLP/CL areas. From ACL 2018. More information about each area available at [link].

Morphology and Morphologically-Rich Languages (MRL)

MRLs

- **MRLs:** Languages like Arabic, Hebrew, and Turkish, where substantial grammatical information is expressed at word level (Tsarfaty et al., 2010).
- **e.g.**
  - In **English:** The concept of a **subject** is realized at the level of the clause (i.e., via word order): The boy is skillful.
  - In **Arabic:** The concept of a **subject** is realized at the word level via use of the *nominative case:* Buckwalter: *"AlaWald***u** *mahir"*.
- **Consequence:** Existence of too many surface word forms in MRLs.

Arabic Varieties

Arabic Varieties

- Arabic is a collection of varieties.
  1. **Modern Standard Arabic (MSA)** (e.g., Badawi, 1973; Abdul-Mageed, 2015)
  2. **Classical Arabic (CA)** (e.g., Bateson, 1967; Ryding, 2005)
  3. **Dialects** (e.g., Bassiouny, 2009; Diab et al., 2010; Holes, 2004; Palva, 1982; Elaraby & Abul-Mageed, 2018)
- Arabic is a **diglossic** language (Ferguson, 1959)

Arabic Morphology

Arabic Morphology

- **Derivational** morphology
- **Inflectional** morphology
- **Agglutinative** morphology

Arabic Derivational Morphology

Arabic Derivation

- **Derivational morphology:** Mostly templatic: word = root + pattern + vocalism, plus some idiosyncratic information
  - **e.g.** root ك ت ب (*k t b*) + vocalism "a" + pattern *123*, ([1,2,3] = root radicals) = *1a2a3*
    → **derivational form** كتب (*katab*, Eng. 'to write')

Arabic Inflectional Morphology

- **Inflectional morphology:** various types of functional features (e.g., voice, number, aspect, gender, tense).
  Resulting word: *lexeme*
  - **e.g.** lexeme كتبت (*katabat*, Eng. 'she wrote')
    Inflectional features: feminine [gender], singular [number], past [tense], perfective [aspect], 3rd [person]
- Verbs inflect for:
  - **Aspect:** *{perfective, imperfective, imperative}*.
  - **Mood:** *{indicative, subjunctive, jussive}*.
  - **Person:** *{1st, 2nd, 3rd}*.
  - **Voice:** *{passive, active}*.
- Nominals inflect for:
  - **Case:** *{nominative, accusative, genetive}*.
  - **State:** *{definite, indefinite, construct}*.
- Verbs & Nominals inflect for:
  - **Gender:** *{masculine, feminine}*.
  - **Number:** *{singular, dual, plural}*.

Arabic Agglutinative Morphology

Arabic Agglutination

- **Agglutinative morphology:** Arabic words often undergo clitic agglutination to form surface words.
  - **e.g.** lexeme كاتبت (*kAtabat*, Eng. 'she corresponded') + enclitic/suffixal pronoun هم → كاتبتهم (*kAtabathum*, Eng. 'she corresponded with them').
- Arabic has **proclitics** and **enclitics**

	Proclitic	**Proclitic**	**Stem**	**Affix**	**Enclitic**
BUCK.	w	b	Hsn	At	hm
GLOSS	and	by	virtue	s	their
TRANS.	And by their virtues				

Table: Clitics: Example Arabic word "wbHsnAthm" (Eng. "And by their virtues") (Diab et al., 2004)

Automatic Processing

Segmentation

- Morphological complexity causes **data sparsity**.
- **Segmentation** of surface words reduces sparsity.

Surface form	Segments	Clitics	Lemma	Lexeme
وبحسناتهم	و+ ب +حسن+ ات+ هم+	و+ ب+ ... +هم	حسنة	حسنات
wbHsnAtHm	*w+b+Hsn+At+Hm*	*w+ b+ ... +hm*	*Hsnp*	*HsnAt*

Table: Difference between segments, lemma, and Lexeme word forms of the word وبحسناتهم (Eng. "and by their virtues").

Morphosyntax

- **Morphosyntactic Disambguation: wbHsnAtHm:**

  w/CONJ+b/PREP+Hsn/NOUN+At/NOM_SUFF_PL+Hm/PRON_POSS

ASMA

ASMA

- New assumption: full morpho-syntactic disambiguation of Arabic can be successfully performed without a morphological analyzer
- AMIRA does not identify inflectional morpheme boundary
  ASMA performs both inflectional morpheme segmentation and agglutinative clitic segmentation
- ASMA then assigns each one of the resulting morphemes a POS tag

Method: Memory-Based Learning

MBL

- One MBL classifier for segmentation and one for POS tagging
- MBL: k-nn learner, stores training instances; for new instances, retrieves k most similar examples from training
- Suitable bias for NLP problems since it does not abstract over irregularities or subregularities.

Data Sets and Splits

ATB

Penn Arabic Treebank:

Data set	# wrds	# tkns	# segs	# texts	Source
ATB1V4	145,386	167,280	209,187	734	AFP
ATB2V3	144,199	169,319	221,001	501	UMMAH
ATB3V3.1	340,281	402,291	551,171	600	An Nahar
ATB3V3.2	339,710	402,291	512,932	599	An Nahar

Table: Data statistics and sources

Data Splits

ATB

- **AMIRA-SPLIT:** split each of first three parts into 10% development data, 80% training data, 10% test data; then concatenate respective splits
- **MADA-SPLIT:** use ATB1V4 + ATB2V3 + first 80% of ATB3V3.1 as TRAIN set, and 10% of ATB3V3.1 as DEV, 10% as TEST

Segmentation Setup

Segmentation

- Definition: IOB classification task, where each letter in word is tagged with a label indicating its place in a segment
- tagset: *{B-SEG, I-SEG, O}*
  B = beginning of segment; I = inside of segment; O = spaces between words
- MBL parameter setting: IB1 algorithm, *weighted overlap*, gain ratio, $k = 1$
- feature set: focus character, its ambiguity tag, six preceding characters, previous tag decisions of all six preceding characters - character immediately preceding focus character, seven following characters

Segmentation Results

System	Split	Acc.	Prec.	Rec.	*F*	Wrd Acc.
ASMA	AMIRA	99.53	97.97	98.04	98.01	98.34
	MADA	99.49	97.72	97.85	97.79	98.10
MADA 3.2						98.85

Table: Segmentation results

SEG Res

- Our segmentation results are not fully comparable to the tokenization performance of AMIRA since AMIRA does not split off inflectional morphology.

Segmentation Results

System	Split	Acc.	Prec.	Rec.	*F*	Wrd Acc.
ASMA	AMIRA	99.53	97.97	98.04	98.01	98.34
	MADA	99.49	97.72	97.85	97.79	98.10
MADA 3.2						98.85

Table: Segmentation results

SEG Res

- Our segmentation results are not fully comparable to the tokenization performance of AMIRA since AMIRA does not split off inflectional morphology.
- MADA, in contrast, does perform segmentation, but it is based on a morphological analyzer. ASMA, without the use of any external resources, achieved a word accuracy of 98.10% on the MADA-SPLIT, which is only slightly lower than MADA's 98.85% word accuracy.

POS Tagging

POS Tagging

- Definition: each segment(not full word) is assigned a POS tag
- Use ATB tagset, but remove case + mood tags
  $\rightarrow$ 139 segment-based tags
- AMIRA uses two tagsets: Reduced TagSet (24 tags) + enriched tagset (75 tags)
- MADA uses ATB tagset
- We test on gold segmented data (same as AMIRA)

POS Tagging Setup

POS Tagging

- MBL parameters: *modified value difference metric*, gain ratio, $k = 1$ for known words, $k = 30$ for unknown words
- Best feature set:
  - **Known words:** focus segment, its ambitag, two previous segments, predicted tag of three previous segments
  - **Unknown words:** five previous segments, their predicted tags, the focus segment itself, its ambitag, first five characters + last three characters of focus segment, six following segments + their ambitags
- Evaluation based on segments, (no full words)

POS Tagging Results

System	Split	Acc_knwn	Acc_unknwn	Acc_all
ASMA	AMIRA	96.61	74.46	96.26
	MADA	94.80	86.00	94.67
MADA 3.2				94.70
AMIRA 2.1 - ERTS				96.13

Table: POS tagging results

POS Res

- The MADA split is considerably more challenging than the AMIRA split.

ASMA in Comparison

ASMA Compared

- For MSA:
  - Inflectional morpheme segmentation and agglutinative clitic segmentation
  - fine grained POS tagging
- Comparison to AMIRA: ASMA has more fine grained morphological disambiguation (it identifies inflectional morpheme boundaries)
- Comparison to MADA: ASMA performs same tasks, BUT: without morphological analyzer; only minimally lower results than MADA's
- Advantage: ASMA's high speed (process 100 000 words in ca. 5 min.)

PyTorch LSTM POS Tagger I

```
def prepare_sequence(seq, to_ix):
    idxs = [to_ix[w] for w in seq]
    return torch.tensor(idxs, dtype=torch.long)


training_data = [
    ("The dog ate the apple".split(), ["DET", "NN", "V", "DET", "NN"]),
    ("Everybody read that book".split(), ["NN", "V", "DET", "NN"])
]
word_to_ix = {}
for sent, tags in training_data:
    for word in sent:
        if word not in word_to_ix:
            word_to_ix[word] = len(word_to_ix)
print(word_to_ix)
tag_to_ix = {"DET": 0, "NN": 1, "V": 2}

# These will usually be more like 32 or 64 dimensional.
# We will keep them small, so we can see how the weights change as we train.
EMBEDDING_DIM = 6
HIDDEN_DIM = 6
```

Out:

```
{'The': 0, 'dog': 1, 'ate': 2, 'the': 3, 'apple': 4, 'Everybody': 5, 'read': 6, 'that': 7,
'book': 8}
```

Figure: [From PyTorch Tutorial: Link]

PyTorch LSTM POS Tagger II

```
class LSTMTagger(nn.Module):

    def __init__(self, embedding_dim, hidden_dim, vocab_size, tagset_size):
        super(LSTMTagger, self).__init__()
        self.hidden_dim = hidden_dim

        self.word_embeddings = nn.Embedding(vocab_size, embedding_dim)

        # The LSTM takes word embeddings as inputs, and outputs hidden states
        # with dimensionality hidden_dim.
        self.lstm = nn.LSTM(embedding_dim, hidden_dim)

        # The linear layer that maps from hidden state space to tag space
        self.hidden2tag = nn.Linear(hidden_dim, tagset_size)
        self.hidden = self.init_hidden()

    def init_hidden(self):
        # Before we've done anything, we dont have any hidden state.
        # Refer to the Pytorch documentation to see exactly
        # why they have this dimensionality.
        # The axes semantics are (num_layers, minibatch_size, hidden_dim)
        return (torch.zeros(1, 1, self.hidden_dim),
                torch.zeros(1, 1, self.hidden_dim))

    def forward(self, sentence):
        embeds = self.word_embeddings(sentence)
        lstm_out, self.hidden = self.lstm(
            embeds.view(len(sentence), 1, -1), self.hidden)
        tag_space = self.hidden2tag(lstm_out.view(len(sentence), -1))
        tag_scores = F.log_softmax(tag_space, dim=1)
        return tag_scores
```

PyTorch LSTM POS Tagger IIIa

```
model = LSTMTagger(EMBEDDING_DIM, HIDDEN_DIM, len(word_to_ix),
len(tag_to_ix))
loss_function = nn.NLLLoss()
optimizer = optim.SGD(model.parameters(), lr=0.1)

# See what the scores are before training
# Note that element i,j of the output is the score for tag j for word i.
# Here we don't need to train, so the code is wrapped in torch.no_grad()
with torch.no_grad():
    inputs = prepare_sequence(training_data[0][0], word_to_ix)
    tag_scores = model(inputs)
    print(tag_scores)

for epoch in range(300):  # again, normally you would NOT do 300 epochs, it
is toy data
    for sentence, tags in training_data:
        # Step 1. Remember that Pytorch accumulates gradients.
        # We need to clear them out before each instance
        model.zero_grad()

        # Also, we need to clear out the hidden state of the LSTM,
        # detaching it from its history on the last instance.
        model.hidden = model.init_hidden()

        # Step 2. Get our inputs ready for the network, that is, turn them
into
        # Tensors of word indices.
        sentence_in = prepare_sequence(sentence, word_to_ix)
        targets = prepare_sequence(tags, tag_to_ix)

        # Step 3. Run our forward pass.
        tag_scores = model(sentence_in)

        # Step 4. Compute the loss, gradients, and update the parameters by
        #  calling optimizer.step()
        loss = loss_function(tag_scores, targets)
        loss.backward()
        optimizer.step()
```

PyTorch LSTM POS Tagger IIIb

```
# See what the scores are after training
with torch.no_grad():
    inputs = prepare_sequence(training_data[0][0], word_to_ix)
    tag_scores = model(inputs)

    # The sentence is "the dog ate the apple".  i,j corresponds to score for
tag j
    # for word i. The predicted tag is the maximum scoring tag.
    # Here, we can see the predicted sequence below is 0 1 2 0 1
    # since 0 is index of the maximum value of row 1,
    # 1 is the index of maximum value of row 2, etc.
    # Which is DET NOUN VERB DET NOUN, the correct sequence!
    print(tag_scores)
```

Out:

```
tensor([[-1.1389, -1.2024, -0.9693],
        [-1.1065, -1.2200, -0.9834],
        [-1.1286, -1.2093, -0.9726],
        [-1.1190, -1.1960, -0.9916],
        [-1.0137, -1.2642, -1.0366]])
tensor([[-0.0858, -2.9355, -3.5374],
        [-5.2313, -0.0234, -4.0314],
        [-3.9098, -4.1279, -0.0368],
        [-0.0187, -4.7809, -4.5960],
        [-5.8170, -0.0183, -4.1879]])
```

Representation Learning | Zellig Harris (1954)

> *“words that are used and occur in the same contexts tend to purport similar meanings.”*

> *“oculist and eye-doctor . . . occur in almost the same environments.”*

> *“If A and B have almost identical environments. . . we say that they are synonyms.”*

1: Statistical LM

$$\hat{P}(W_1^T) = \prod_{t=1}^{T} \hat{P}(w_t | w_1^{t-1})$$

Resurgence of Language Models

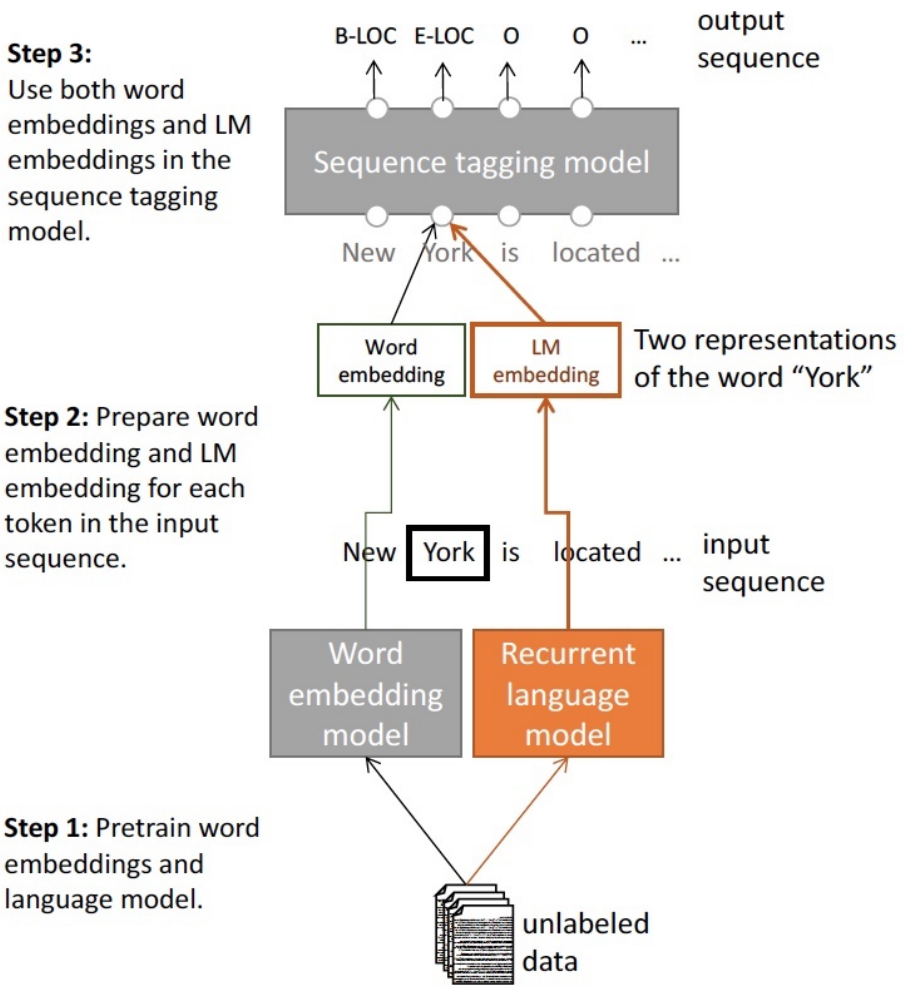


Figure: Peters et al. (2017): Semi-supervised sequence tagging with bidirectional language models.

Machine Translation

Facebook's AI Just Set A New Record In Translation And Why It Matters

Hieroglyphics are seen on the sarcophagus once belonging to the judge and prime minister Gemenefherbek of Sais at the Egyptian Museum in Turin, Italy, Tuesday, February 21, 2006. Photographer: Adam Berry/Bloomberg News

Figure: [From Forbes]. **Note:** An improvement of 1 BLEU point (a common metric for judging the accuracy of MT) is considered a remarkable achievement in this field; FB methods showed an improvement of more than 10 BLEU points.

Issues With MT

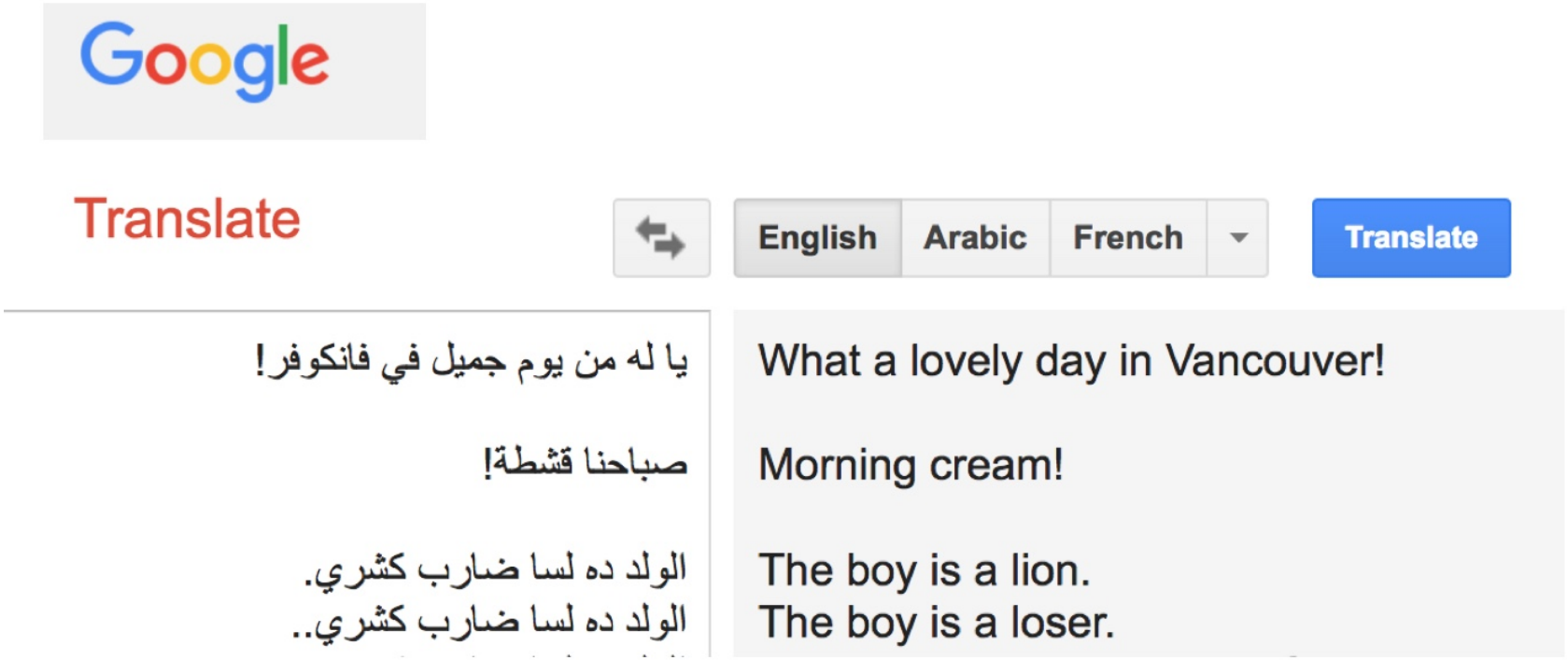


Figure: Google Arabic/Egyptian Arabic-English Translation samples, Oct. 24, 2018

Issues With MT

The Atlantic

The Shallowness of Google Translate

The program uses state-of-the-art AI techniques, but simple tests show that it's a long way from real understanding.

DOUGLAS HOFSTADTER | JAN 30, 2018 | TECHNOLOGY

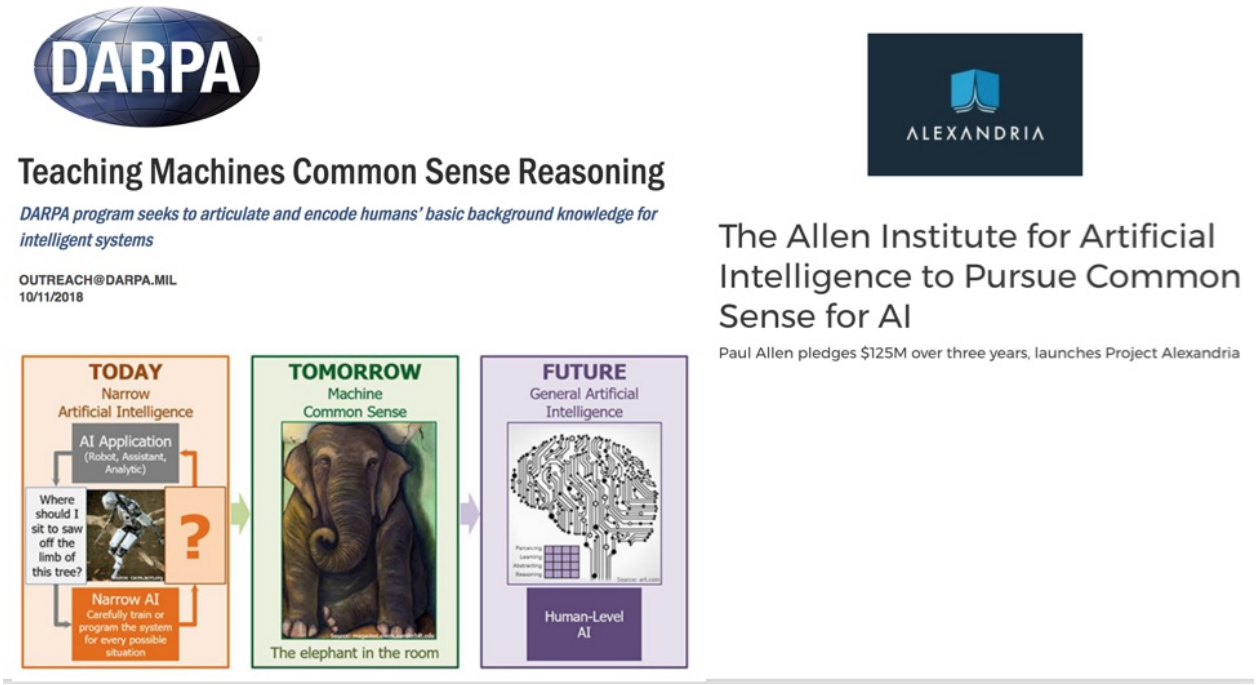
Style Transfer

Style Transfer: Language

Relaxed ↔ Annoyed	
Relaxed	Sitting by the Christmas tree and watching Star Wars after cooking dinner. What a nice night 💗🎄✨
Annoyed	Sitting by the computer and watching The Voice for the second time tonight. What a horrible way to start the weekend 😡😡😡
Annoyed	Getting a speeding ticket 50 feet in front of work is not how I wanted to start this month 😕
Relaxed	Getting a haircut followed by a cold foot massage in the morning is how I wanted to start this month 😊
Male ↔ Female	
Male	Gotta say that beard makes you look like a Viking...
Female	Gotta say that hair makes you look like a Mermaid...

Figure: [From Anonymous authors on Open Review, under review for ICLR 2019: MULTIPLE-ATTRIBUTE TEXT REWRITING]

Common Sense AI



Oren Etzioni, CEO of AI2

- 'If I put my socks in a drawer, will they still be in there tomorrow?'
- 'How can you tell if a milk carton is full?'

Natural Language Inference

Natural Language Inference

- **Task: determine whether a natural language hypothesis *h* can be inferred from a natural language premise *p*:**
  - Julian is a computational linguist.
  - **Julian is a scientist.**
  - Julian owns a cat.

Conversational Agents in the Wild

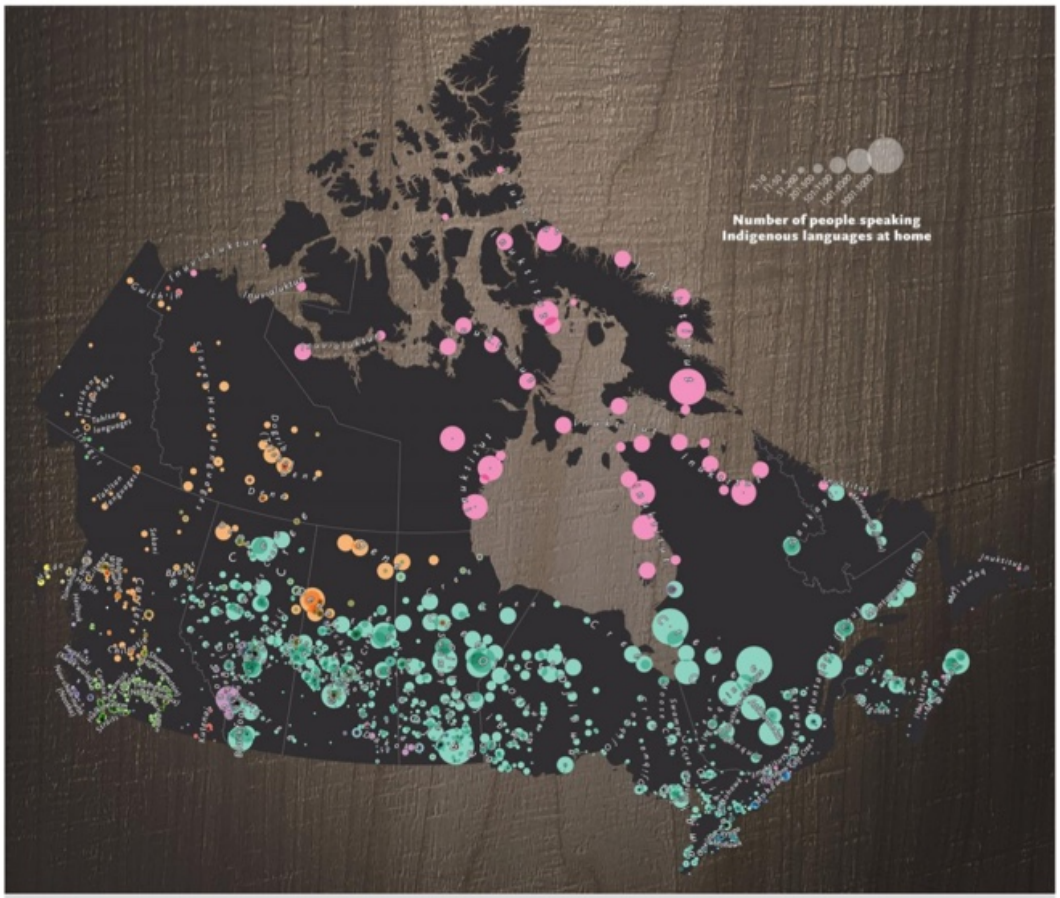
Figure: [From Google Image Search]

Low-Resource Languages

CANADIAN Geographic

Mapping Indigenous languages in Canada

See where 60 languages belonging to 12 language families are being used right now



Where Indigenous languages are being spoken now in Canada. Scroll down for a closer look at this map and the country's 12 Indigenous language families. (Map: Chris Brackley/Can Geo)

Lessons from Natural Language Inference in the Clinical Domain

Alexey Romanov
Department of Computer Science
University of Massachusetts Lowell*
Lowell, MA 01854
aromanov@cs.uml.edu

Chaitanya Shivade
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
cshivade@us.ibm.com

Abstract

State of the art models using deep neural networks have become very good in learning an accurate mapping from inputs to outputs. However, they still lack generalization capabilities in conditions that differ from the ones encountered during training. This is even more challenging in specialized, and knowledge intensive domains, where training data

MultiNLI corpus (Williams et al., 2018) which introduced NLI corpora from multiple genres (e.g. fiction, travel) was a welcome step towards addressing these limitations. MultiNLI offers diversity in linguistic phenomena, which makes it more challenging.

Patient data is guarded by careful access protection due to its sensitive content. Therefore, the common approach of using crowd sourcing plat-