Development of Chatbot Technologies and Challenges

Yik-Cheung (Wilson) Tam

wilsontam@tencent.com

Wechat AI

Tencent

Talk Outline

- Introduction to Wechat AI
- Chatbot development history
- Chatbot technologies
  - Task-oriented: Personal Digital Assistants (PDAs) (语音助手)
  - Chit-chat oriented: Xianer Monk
- Challenges and future

WeChat is the leading mobile social network in China. In 6 years, WeChat has gained...

846 million monthly active users

10 million Official Accounts

200 thousand developers

300 million WeChat Pay users

Data: Tencent Financial Reports

Service Accounts

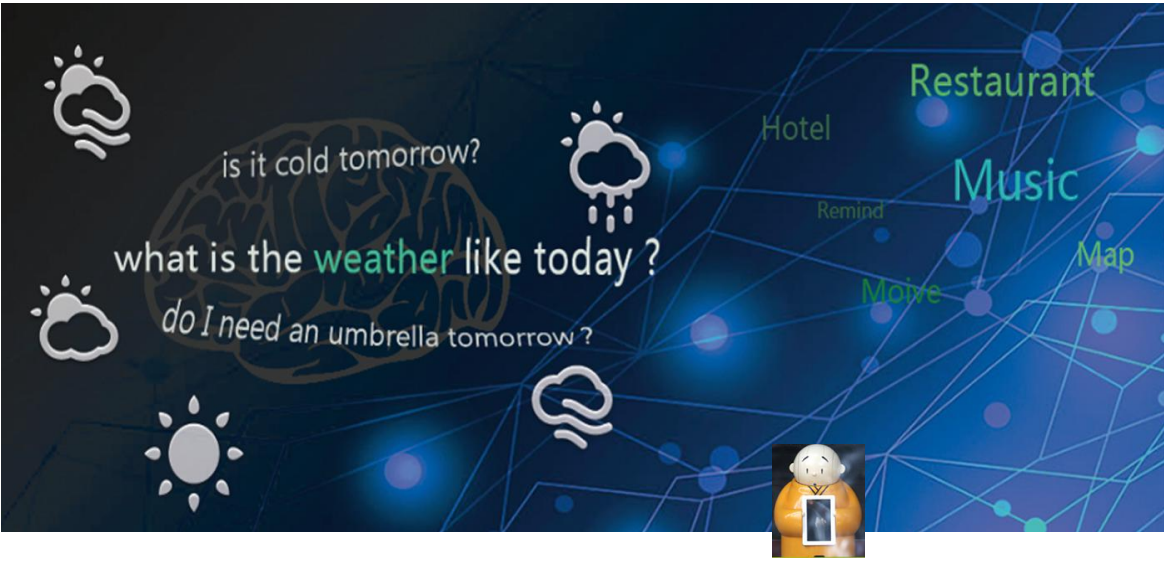
China Merchants Bank

- Over 10 million followers
- Open an account
- Pay bill/loan
- Receive payment notifications
- Receive CRM promotions

Messaging
(Can be automated)

Account management

Powered by WeChat



Chatbot

- Natural to server customers
- Powerful for users to acquire service, information, knowledge, etc.

Chatbot on WeChat

Examples

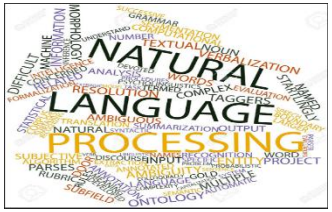
- WeBank
- WeChat official account
- Tencent games
- Xiao'er Monk

WeChat AI are hiring now!

AI@tencent.com wilsontam@tencent.com

Beijing, Guangzhou, Shenzhen, Palo Alto

Voice & Audio



Natural Language Processing

Machine Translation

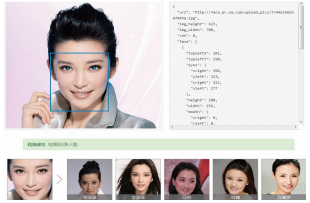


Image & Video

Machine Learning

Talk Outline

- Introduction to Wechat AI
- Chatbot development history
- Chatbot technologies
  - Task-oriented Personal Digital Assistant (PDA) (语音助手)
  - Chit-chat oriented Xianer monk (贤二机器僧)
- Challenges and future

Amazon sold nine times as many Amazon Echo devices this holiday

Posted Dec 27, 2016 by Brian Heater (@bheater)

BRIAN BARRETT GEAR 02.23.17 12:00 PM

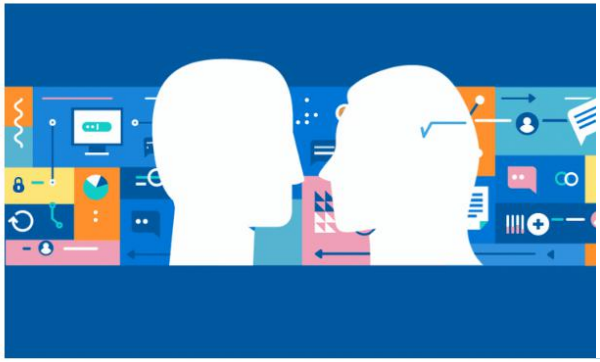
AMAZON ALEXA HITS 10,000 SKILLS. HERE COMES THE HARD PART

AMAZON

A YEAR AND a half ago, Amazon opened up its Alexa voice assistant to developers. With the Alexa Skills Kit, Alexa and

Microsoft and AI: Introducing social chatbot Zo, Cortana takes on new roles and more

Posted December 13, 2016 By Microsoft Corporate Blogs



技术沙龙邀请 | 解密 chatbot 人工智能聊天机器人 第二期

「2016 年最火的行业方向不是 VR，而是对话交互/对话机器人（bot/chatbot）。Google 发布了 Gmail 和 Allo 上的智能回复功能，并收购了美国最大的 bot 创建平台 api.ai。Microsoft 推出了自己的语言理解平台 LUIS 以及 Bot Framwork。Facebook 的 bot 创建平台 wit.ai 进行了大规模的改版以便提升机器人的智能性，其 Messenger 上对话机器人数量已超过 3 万。原 Siri 团队研发的 bot 平台 Viv 还未发布就已被三星收购。Amazon 推出了自己的语音交互系统 Lex，以丰富 Echo 的使用场景……」

这些，只是对话交互时代到来的序曲。

CHATBOTS AND COMMERCE

Can Apple's Siri Stand Up To The Chatbot Competition?

By PYMNTS

Posted on September 21, 2016

腾讯云小微激活硬件生态 携合作伙伴产品正式亮相

2017年06月22日 20:36:00

来源：36氪

0人参与 0评论

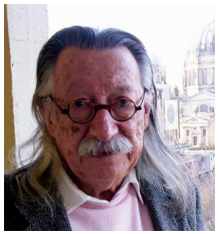
传阿里巴巴7月5日发布智能音箱 苹果压力倍增

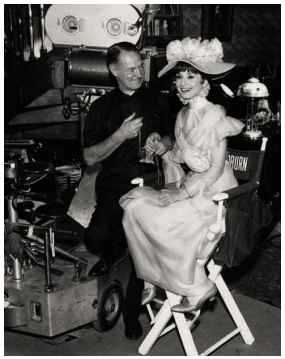
IT业界 腾讯科技 2017-06-30 07:11

110 评论

ELIZA

- 1964-1966, Joseph Weizenbaum, MIT
- Eliza Doolittle
- Mimick a psychologist talking to a patient
- Exploit medical record, keyword & rules, generate answers





http://www.masswerk.at/elizabot/eliza_test.html

This is the "typical conversation" as described by Joseph Weizenbaum:

Men are all alike.

IN WHAT WAY?

They're always bugging us about something or other.

CAN YOU THINK OF A SPECIFIC EXAMPLE?

Well, my boyfriend made me come here.

YOUR BOYFRIEND MADE YOU COME HERE?

He says I'm depressed much of the time.

I AM SORRY TO HEAR YOU ARE DEPRESSED.

It's true.
I am unhappy.

DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY.

I need some help, that much seems certain.

WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP?

Perhaps I could learn to get along with my mother.

TELL ME MORE ABOUT YOUR FAMILY.

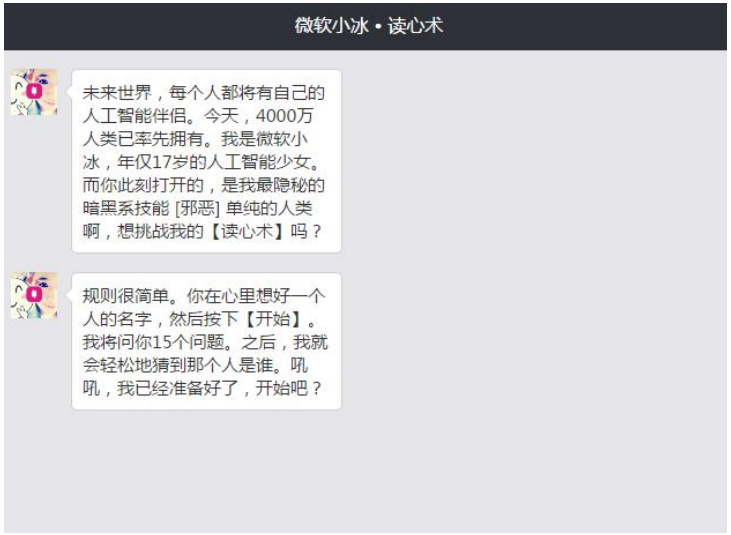
- "ELIZA — A Computer Program for the Study of Natural Language Communication between Man and Machine," *Communications of the Association for Computing Machinery 9* (1966): 36-45.[1]

Siri

- A spinoff company from SRI International at 2007
- Originated from a DARPA project
- 2010: acquired by Apple
- 2011: deployed on iphone-4

Microsoft Cortana & Xiaoice

- Sister chatbots
- Cortana： personal assistant， 2014年， 与Siri相似
- XiaoIce： 闲聊对象， 2014年微软亚洲研究院



Amazon Echo

- 智能音响 Amazon Echo (2014年底)
- 智能助手Alexa
- 功能：播放音乐、讲故事、控制家庭电器(开灯、关灯）
- 支持第三方功能扩充 （App）
  - Alexa Skills Kit （API）
  - 云端运算
- 硬体厂商制作支持Alexa的电器、汽车等等

亚马逊Echo

天猫精灵

Google Home

苹果HomePod

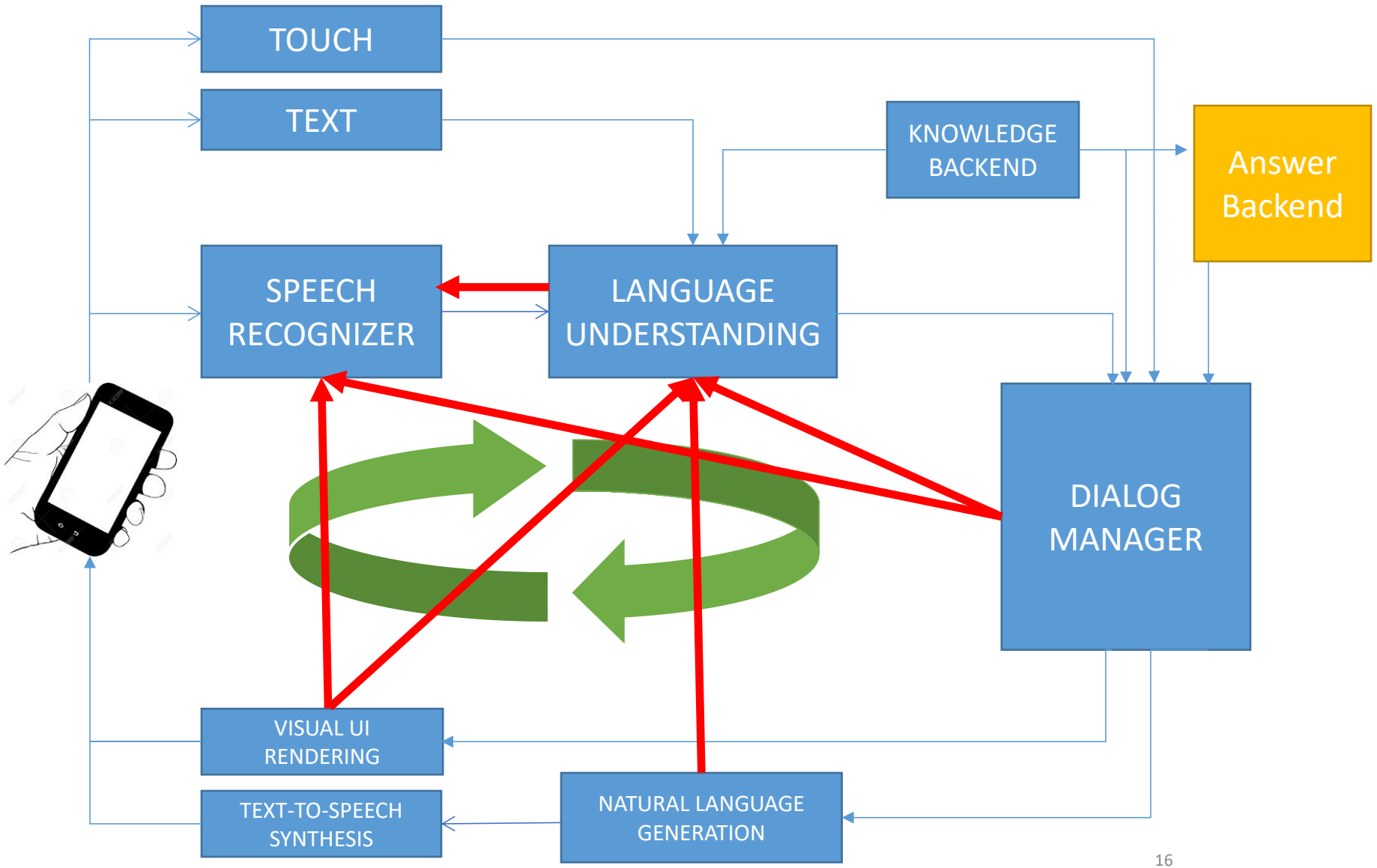
Talk Outline

- Introduction to Wechat AI
- Chatbot development history
- Chatbot technologies
  - Task-oriented Personal Digital Assistant (PDA) (语音助手)
  - Chit-chat oriented Xianer monk (贤二机器僧)
- Challenges and future

What is a Personal Digital Assistant?

- **Meta layer of intelligence**
  - Sits on top of other services and applications
  - Performs actions using services/apps to fulfill the user's intent
  - Serves structured content and data to the user
  - Natural language interface
  - Relies on
    - speech recognition,
    - natural language understanding,
    - dialog management,
    - answers,
    - ranking, inference, personalization, etc..

Architecture



Language understanding (LU) as form filling

- “我想订一张明天从上海到北京的机票”

intent classification

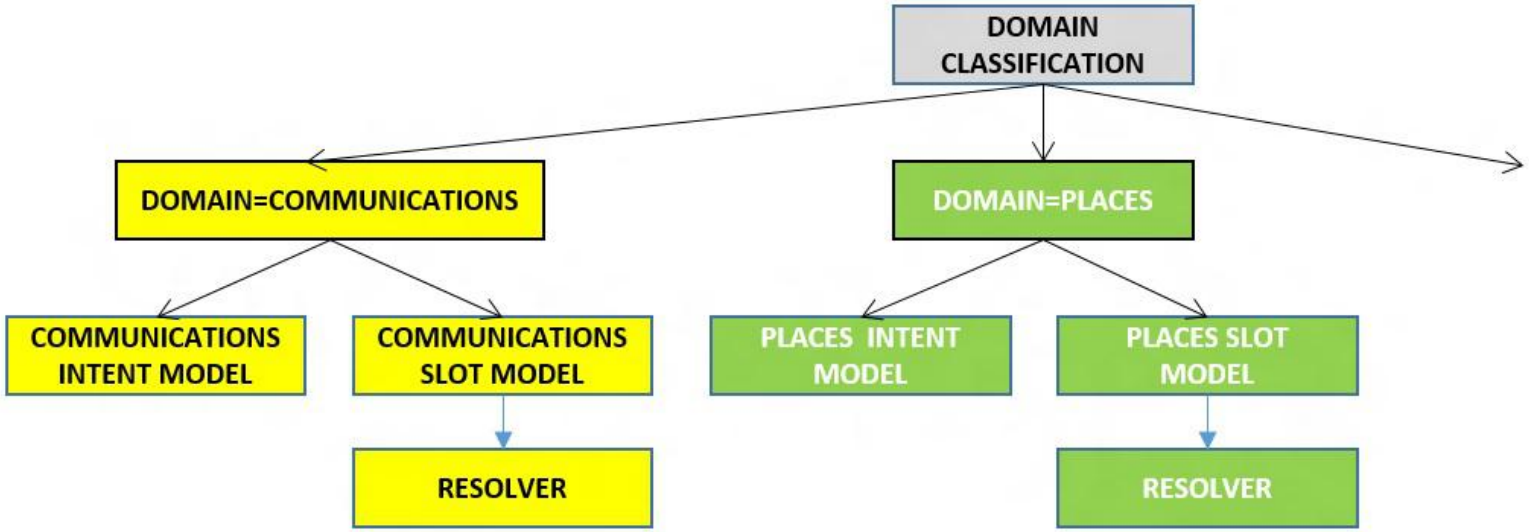
canonical form conversion

Tomorrow -> 2016-07-19

Action: Query database

LU tasks

- Understanding semantics = {domain, intent, slot}
  - A domain can have multiple intents
- *I want to watch a <genre> funny </genre> movie*
  - Domain = movie
  - Intent = find_movie
- Domains = {movie, sports, …}
  - Intents(movie)={find_movie, find_director, ..}

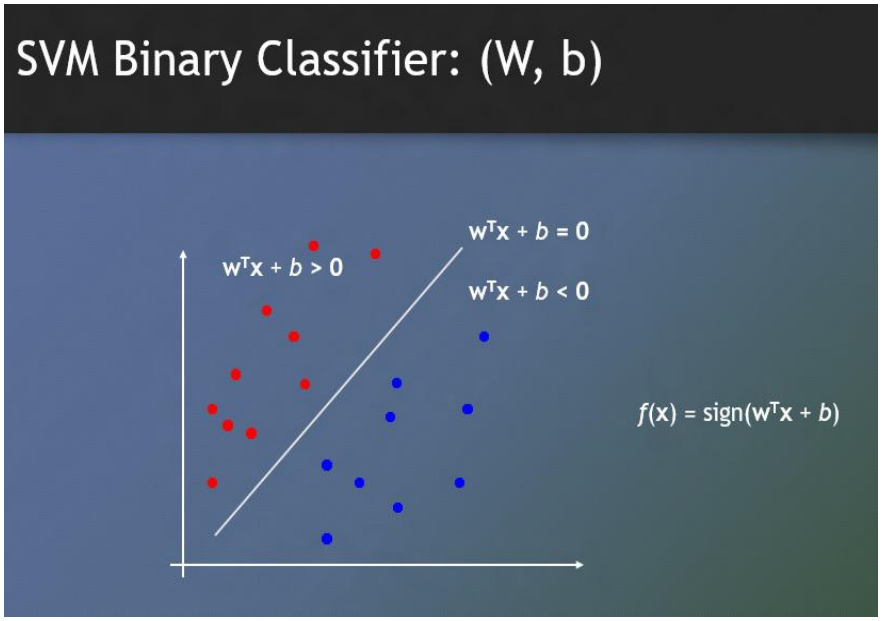


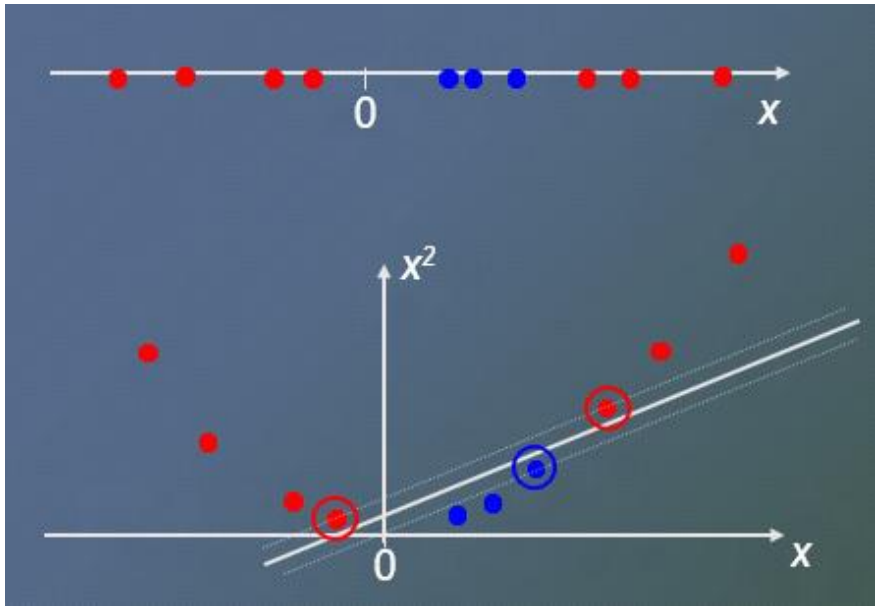
Domain/Intent classification

- Support vector machine (SVM)
- Convolutional neural network (embedding based)
- Fasttext (embedding based)

Binary SVM

- 1 class against other classes
  - Positive domain data = {I want to watch a funny movie, …}
  - Negative domain data = {How is the weather tomorrow?, …}
  - Ngram features, etc.





Word N-gram features for SVM

- Query = “i want to watch a funny movie”, Domain = “Movie”
- 1 gram: “i”, “want”, “to”, “watch”, “a”, “funny”, “movie”
- 2-gram: “I want”, “want to”, “to watch”, “watch a”, “a funny”, “funny movie”
- 3-gram: …
- {1,2,3} grams form a high-dimensional sparse feature space of a query
  - Feature dimensions: O(V * V * V)
  - Very sparse: Only some N-gram features are turned ON; Rest are OFF
  - Usually, a linear kernel is enough
- Public software library: liblinear
  - http://www.csie.ntu.edu.tw/~cjlin/liblinear/

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Word2vec [Mikolov 2013]

- Distributed representation of word
- Skip-gram

Interesting properties

Simple vector addition can produce meaningful results:

- Vec(“king”) – vec(“man”) + vec(“woman”) ~= vec(“queen”)
- Vec(“Russia”) + vec(“river”) ~= vec(“Volga River”)
- Vec(“Germany”) + vec(“capital”) ~= vec(“Berlin”)

Skip-gram model

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

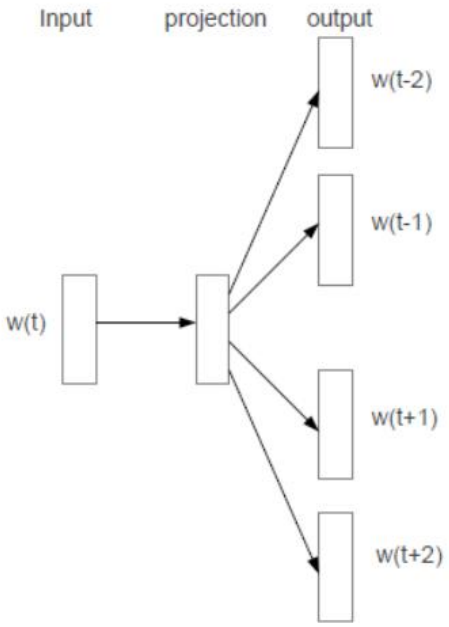


Figure 1: The Skip-gram model architecture. The training objective is to learn word vector representations that are good at predicting the nearby words.

Distributed Representations of Words and Phrases and their Compositionality, 2013

Negative sampling

- Softmax (expensive normalization)

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

- Avoid softmax

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I})\right]$$

Convolutional Neural Networks for Sentence Classification [Kim 2014]

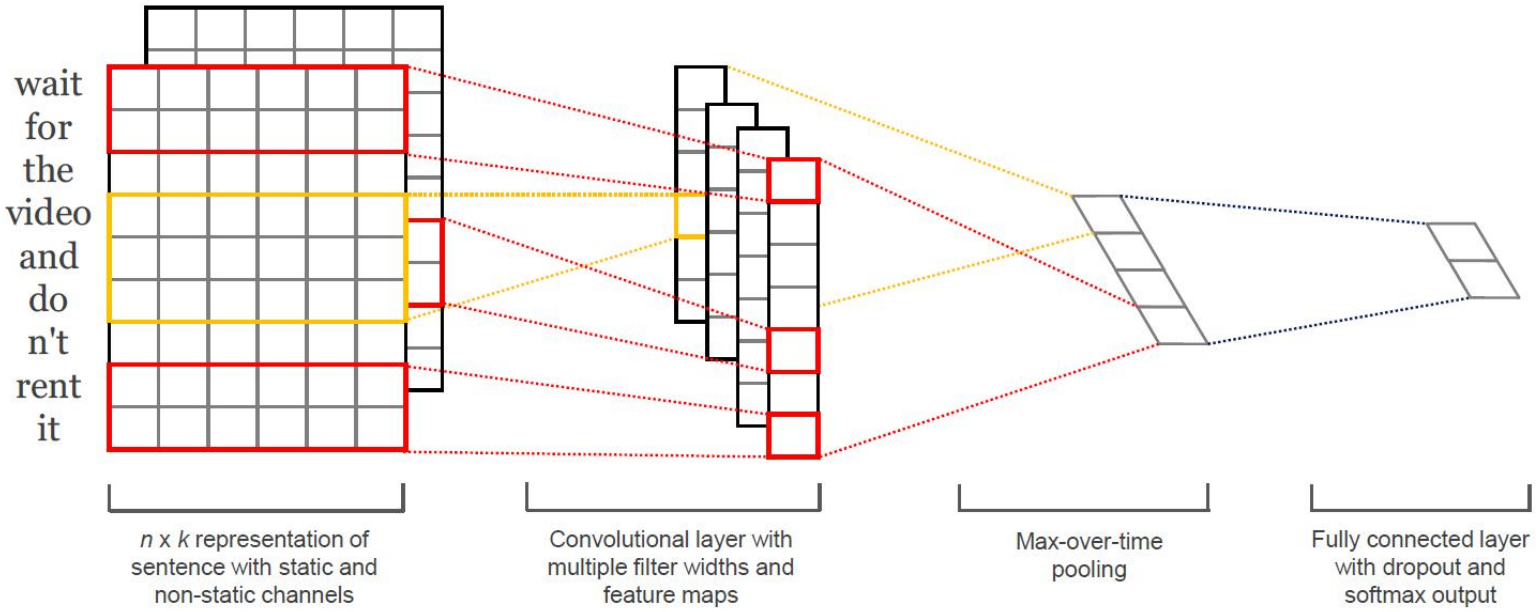


Figure 1: Model architecture with two channels for an example sentence.

CNN for sentence classification – Kim et. al. 2015

Note

- A variable-length sentence is mapped into a fixed dimensional vector
- Word embedding can be pretrained using large amount of text corpora
  - Google word2vec (100B Google news)
  - Benefit from unsupervised learning
- Updating word embedding during training usually yields better results

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Table 2: Results of our CNN models against other methods. **RAE**: Recursive Autoencoders with pre-trained word vectors from Wikipedia (Socher et al., 2011). **MV-RNN**: Matrix-Vector Recursive Neural Network with parse trees (Socher et al., 2012). **RNTN**: Recursive Neural Tensor Network with tensor-based feature function and parse trees (Socher et al., 2013). **DCNN**: Dynamic Convolutional Neural Network with k-max pooling (Kalchbrenner et al., 2014). **Paragraph-Vec**: Logistic regression on top of paragraph vectors (Le and Mikolov, 2014). **CCAE**: Combinatorial Category Autoencoders with combinatorial category grammar operators (Hermann and Blunsom, 2013). **Sent-Parser**: Sentiment analysis-specific parser (Dong et al., 2014). **NBSVM, MNB**: Naive Bayes SVM and Multinomial Naive Bayes with uni-bigrams from Wang and Manning (2012). **G-Dropout, F-Dropout**: Gaussian Dropout and Fast Dropout from Wang and Manning (2013). **Tree-CRF**: Dependency tree with Conditional Random Fields (Nakagawa et al., 2010). **CRF-PR**: Conditional Random Fields with Posterior Regularization (Yang and Cardie, 2014). **SVM_S**: SVM with uni-bi-trigrams, wh word, head word, POS, parser, hypernyms, and 60 hand-coded rules as features from Silva et al. (2011).

Fasttext [Joulin 2016]

- Bag of Tricks for Efficient Text Classification
  - https://arxiv.org/abs/1607.01759
- Ngram embedding features
- Hashing trick
- Highly efficient in computation (multi-core CPU)

Fasttext architecture

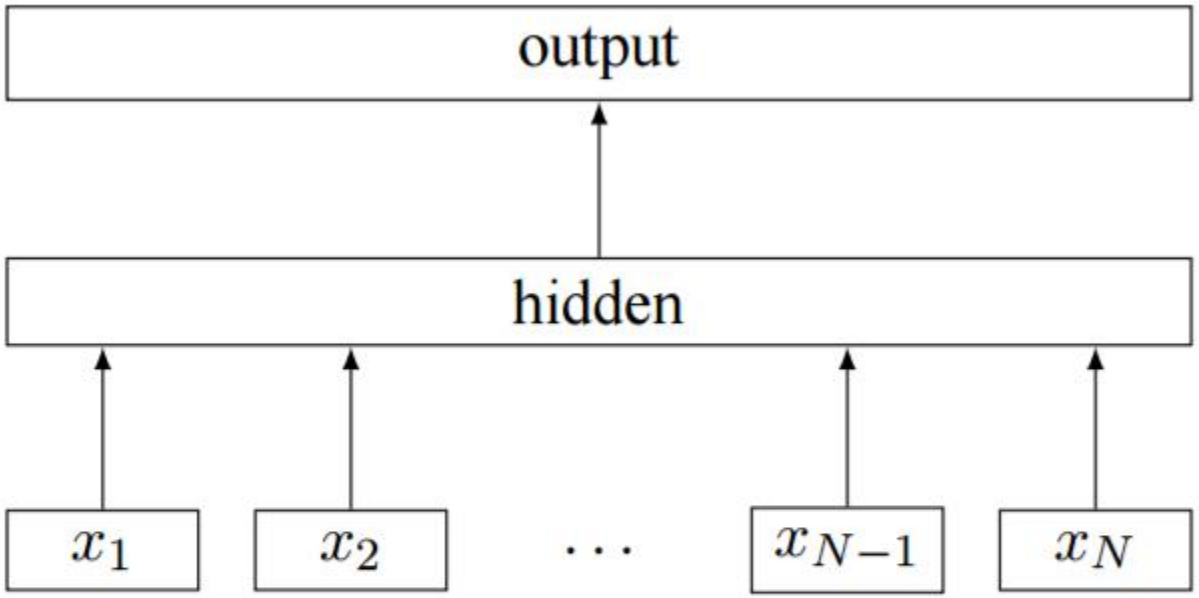


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

Bag of Tricks for Efficient Text Classification – Joulin 2016

Compute embedding average

- Query = “<s> i want to watch a funny movie </s>”
- 1 gram: vec(“i”), vec(“want”), vec(“to”), vec(“watch”), vec(“a”), vec(“funny”), vec(“movie”)
- 2-gram: vec(“<s> I”), vec(“I want”), vec(“want to”), vec(“to watch”), vec(“watch a”), vec(“a funny”), vec(“funny movie”), vec(“movie </s>”)
- 3-gram: …
- Hidden vector = Average(1-gram vec, 2-gram vec, 3-gram vec)

Hashing trick

- Use a hash function to map word Ngram into an index of an embedding table
  - Word trigram (i, j, k): index = (i * p^2 + j * p + k) mod table size

```
for (int32_t i = 0; i < hashes.size(); i++) {
  uint64_t h = hashes[i];
  for (int32_t j = i + 1; j < hashes.size() && j < i + n; j++) {
    h = h * 116049371 + hashes[j];
    int64_t id = h % args_->bucket;
    if (pruneidx_size_ > 0) {
      if (pruneidx_.count(id)) {
        id = pruneidx_.at(id);
      } else {continue;}
    }
    line.push_back(nwords_ + id);
  }
}
```

- Improve efficiency without storing Ngram lookup tables in memory
- Embedding table is trained via backpropagation

Model	AG	Sogou	DBP	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW (Zhang et al., 2015)	88.8	92.9	96.6	92.2	58.0	68.9	54.6	90.4
ngrams (Zhang et al., 2015)	92.0	97.1	98.6	95.6	56.3	68.5	54.3	92.0
ngrams TFIDF (Zhang et al., 2015)	92.4	97.2	98.7	95.4	54.8	68.5	52.4	91.5
char-CNN (Zhang and LeCun, 2015)	87.2	95.1	98.3	94.7	62.0	71.2	59.5	94.5
char-CRNN (Xiao and Cho, 2016)	91.4	95.2	98.6	94.5	61.8	71.7	59.2	94.1
VDCNN (Conneau et al., 2016)	91.3	96.8	98.7	95.7	64.7	73.4	63.0	95.7
`fastText`, $h = 10$	91.5	93.9	98.1	93.8	60.4	72.0	55.8	91.2
`fastText`, $h = 10$, bigram	92.5	96.8	98.6	95.7	63.9	72.3	60.2	94.6

Table 1: Test accuracy [%] on sentiment datasets. `FastText` has been run with the same parameters for all the datasets. It has 10 hidden units and we evaluate it with and without bigrams. For char-CNN, we show the best reported numbers without data augmentation.

	Zhang and LeCun (2015)		Conneau et al. (2016)			`fastText`
	small char-CNN	big char-CNN	depth=9	depth=17	depth=29	$h = 10$, bigram
AG	1h	3h	24m	37m	51m	1s
Sogou	-	-	25m	41m	56m	7s
DBpedia	2h	5h	27m	44m	1h	2s
Yelp P.	-	-	28m	43m	1h09	3s
Yelp F.	-	-	29m	45m	1h12	4s
Yah. A.	8h	1d	1h	1h33	2h	5s
Amz. F.	2d	5d	2h45	4h20	7h	9s
Amz. P.	2d	5d	2h45	4h25	7h	10s

Table 2: Training time for a single epoch on sentiment analysis datasets compared to char-CNN and VDCNN.

Lexicon features

- Motivation: Entity may not be covered in training data (e.g. song title)
- How to exploit lexicon to improve coverage?
- What we tried and worked:
  - As binary features
  - Lexical replacement
- Lexicon denoising

Slot Tagging

- linear-chain conditional random field [Lafferty 2001]
- A sequence tagger: tag each word with a label. I.e. |X|=|Y|

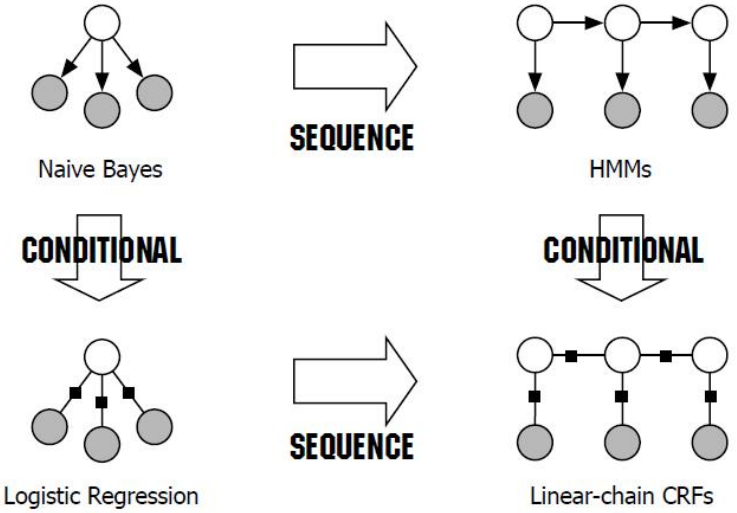
I	want	to	fly	from	San	Francisco	to	Seattle
O	O	O	O	O	B_SCity	I_SCity	O	B_TCity

- Each word is labeled by one of the n possible tags =

  {O, B_SCity, I_SCity, B_TCity, I_TCity, B_Date, I_Data, B_Time, I_Time}
- RNN variations
- Rules are used to fix embarrassing errors by statistical models
  - Statistical model: high recall
  - Rule-based approach: high precision

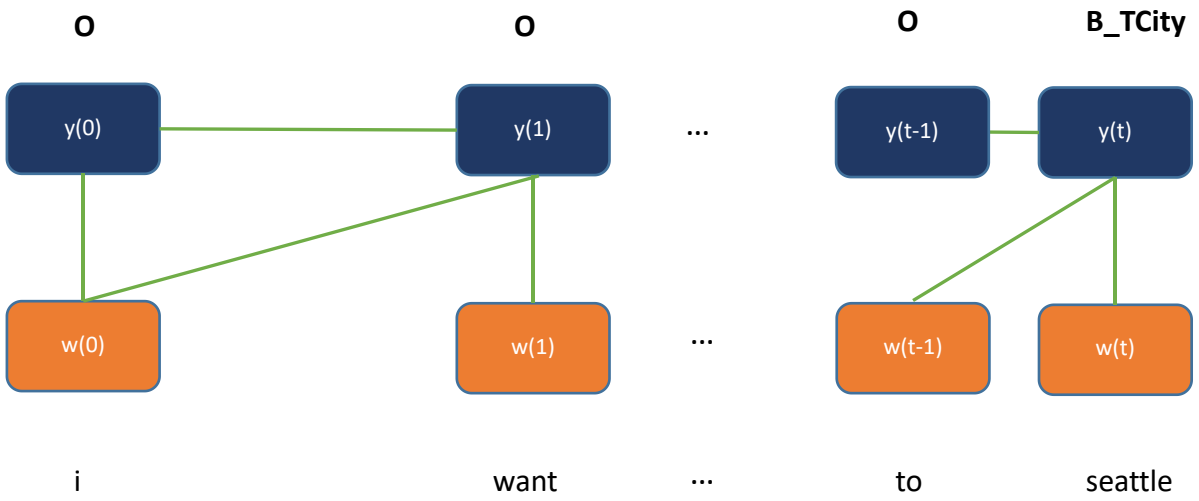
Linear-Chain CRF (LCCRF)

- A sequential discriminative model Pr(Y | Q) [Lafferty 2001]
  - Generative: Naïve Bayes: Pr(w, y); HMM: Pr(Q, Y)
  - Discriminative: LR: Pr(y | w)
- Q = “i want to fly from san francisco to seattle”
- Y = “O O O O O B_Scity I_Scity O B_Tcity”



http://homepages.inf.ed.ac.uk/csutton/publications/crftutv2.pdf

Feature engineering in LCCRF



Triggered feature functions at time t:

- Tag transition indicator features (0/1):
  - Is previous tag = “O” AND current tag = “B_TCity”?
- Word bigram indicator features (0/1):
  - Is previous word = “to” AND current word = “seattle” AND current tag = “B_TCity” ?
- Lexicon indicator features (0/1):
  - Is current word “seattle” in Absolute_Location lexicon?

Mathematical form of LCCRF

- Exponential model:

$$Pr(Y|X) = \frac{1}{Z(X)} \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, X)$$

$$\Psi_t(y_t, y_{t-1}, X) = e^{\sum_k \theta_k f_k(y_t, y_{t-1}, X)}$$

  - Local feature extractor at position t
  - Similar to unnormalized transition and emission probability in HMM

- Feature templates:
  - O(|Y| * |Y|) tag transition features
  - O(|V| * |V| * |Y|) joint word bigrams and tag features
  - Feature functions are collected from slot-labeled training data
  - Count cutoff can be applied to remove rare features
- Model training:
  - Forward-backward algorithm like HMM
  - L1, L2 norm for model regularization. L1 norm to compress model size
- Public software: e.g. CRF++, Mallet, Wapiti etc

  http://mallet.cs.umass.edu/sequences.php
  https://sourceforge.net/projects/crfpp/
  https://wapiti.limsi.fr/

From LCCRF to RNN

- Number of feature functions are based on training data
  - Feature functions may not be triggered on unseen word context
  - Feature functions are generally local N-gram context

- We employ a recurrent neural network approach

$\Rightarrow$Hidden features are learned with raw inputs

$\Rightarrow$Feature functions capture long-distance context

$\Rightarrow$Replace N-gram feature functions in CRF with hidden features:
  - Recurrent neural network [Yao et al. 2013]
  - Long-short term memory network [Yao et al. 2014]
  - Recurrent CRF [Yao et al. 2014]
  - Recurrent SVM [Shi et al. 2015]

- Analogy: N-gram LM versus RNNLM
  - Given word history, predict the next word
  - e.g. RNNLM has better word prediction power than N-gram LM on ATIS

Introduction to recurrent neural network

- Simple architecture (Has a more complex model like LSTM, GRU):

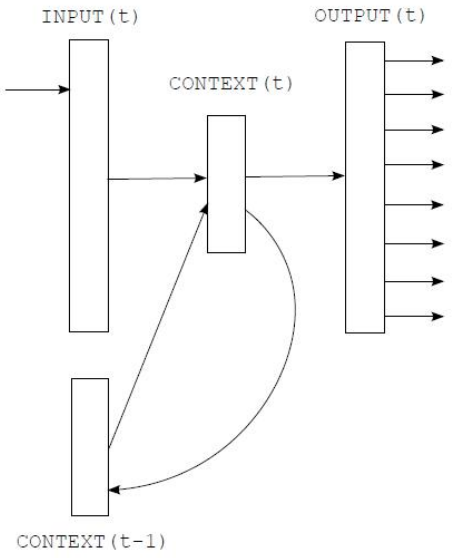
$$\Pr(y(t) \mid x(t), s(t-1)) = \Pr(h(t) \mid x(t), s(t-1)) * \Pr(y(t) \mid s(t))$$

Predict hidden features at time t

Predict output label given hidden feature

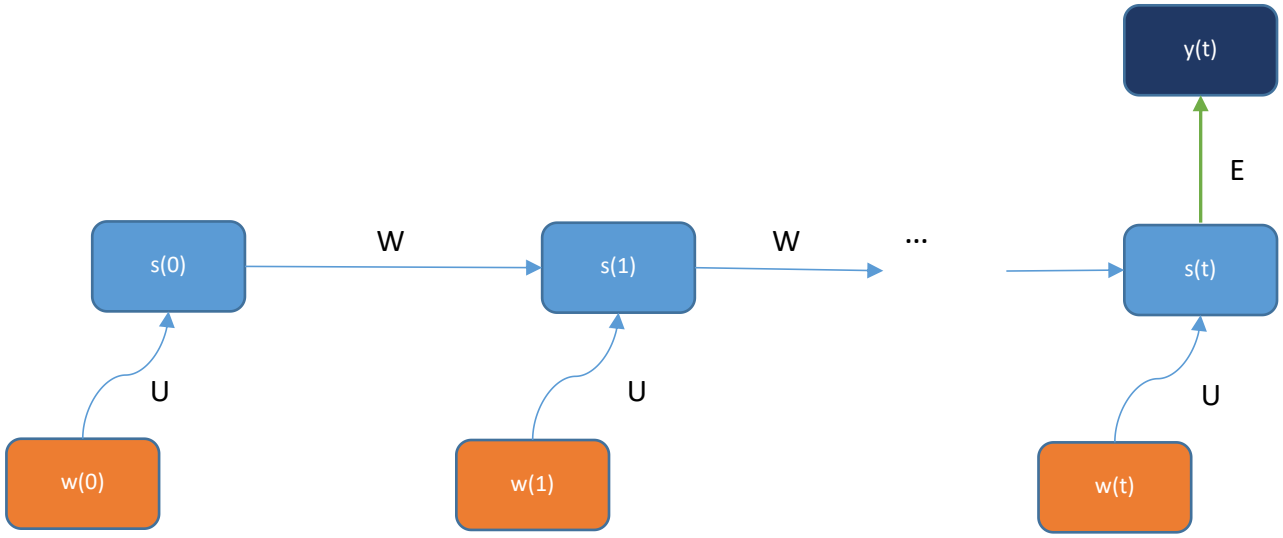
- s(t-1) to capture (long) historical context

- Applications:
  - language modeling [Mikolov et al 2010]
  - Slot tagging [Yao et al 2013,2014]
  - Handwriting recognition [Graves et al 2009]



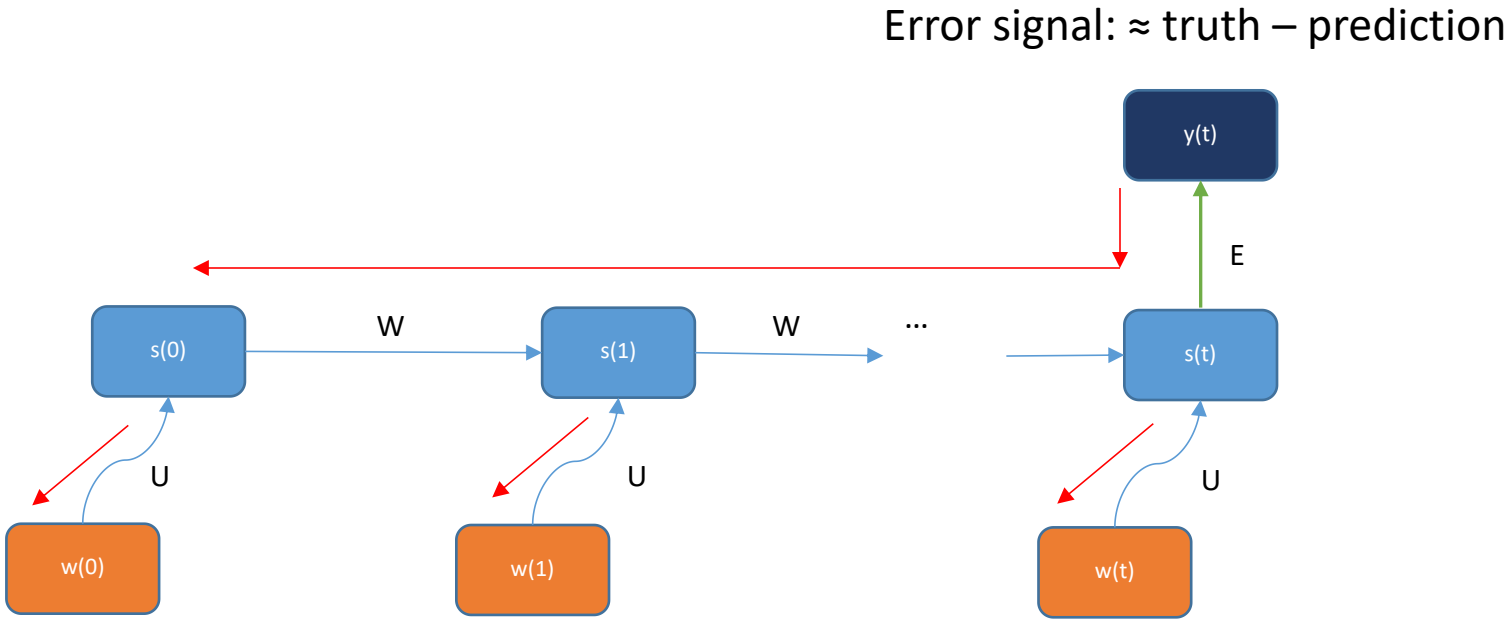
RNN as a deep neural network after unfolding

Prediction stage: Feed forward



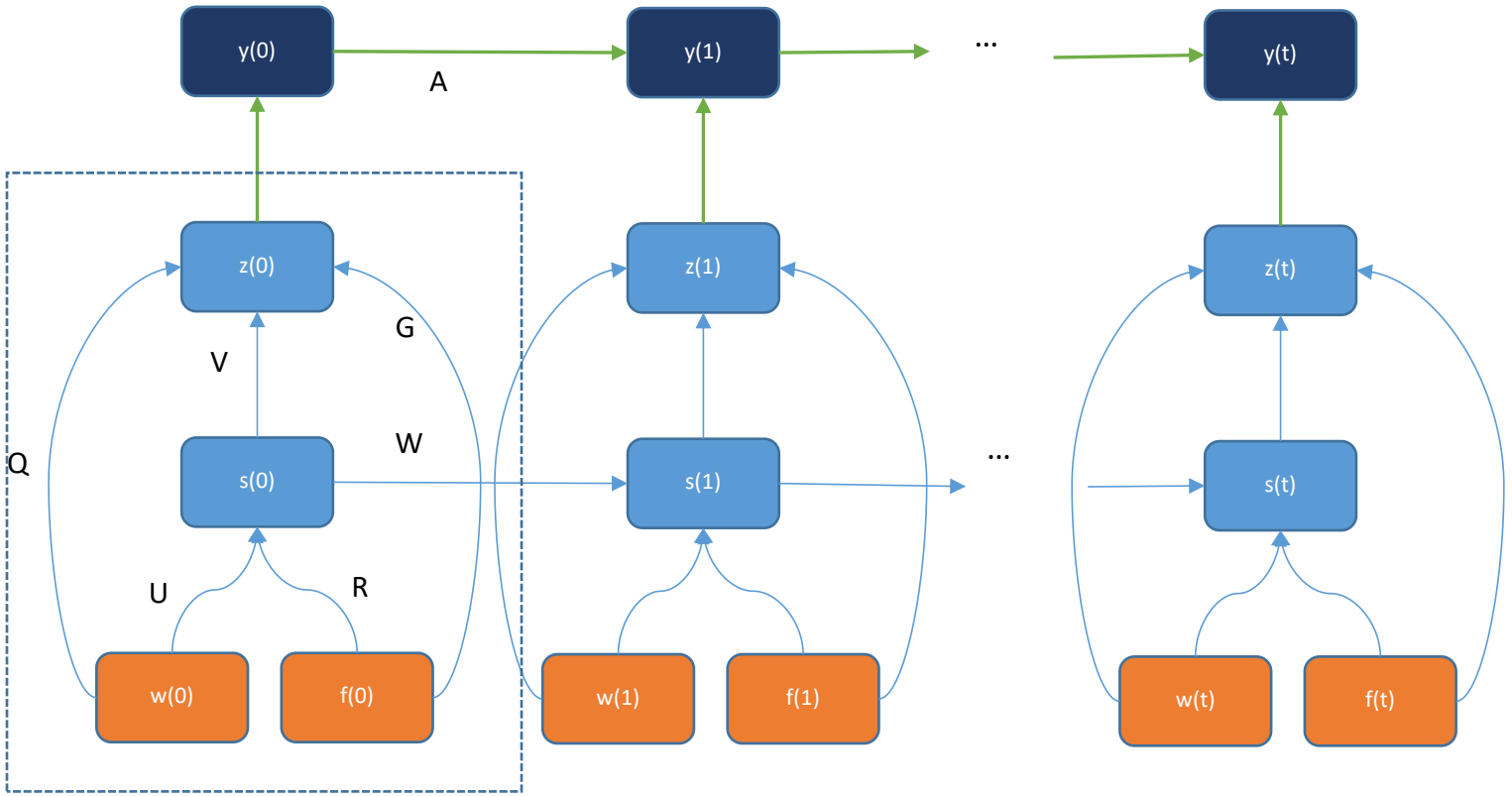
Model update: Backpropagation through time

Update stage: Backpropagation



RNN-CRF Models (RCRF) for Slot Tagging

z(t) is the hidden feature vectors before Softmax
Use sequential criteria for optimization like in CRF



Recurrent CRF [Yao et al. 2014]

From RCRF to RSVM

- Replace Softmax by maximum margin
- CRF Training

$$\max\{\frac{e^{\sum_{t=1}^{T}(\rho a_{y(t-1)^* y(t)^*} + z(t)_{y(t)^*})}}{\sum_{\forall y(1:T)} e^{\sum_{t=1}^{T}(\rho a_{y(t-1)y(t)} + z(t)_{y(t)})}}\}$$

- SVM Training

$$\min_{V,A}\{\frac{1}{2}\|V\|_2^2 + \frac{1}{2}\|A\|_2^2 + C\sum_{k=1}^{K}\zeta_k\}$$

Truth Predict loss

$$s.t.\ F(Y^{k*}) + \zeta_k \geq F(Y^k) + L(Y^k, Y^{k*}) \quad \forall Y^k$$

$$\zeta_k \geq 0$$

$$F(Y^k) = \sum_{t=1}^{T} a_{y^k(t-1)y^k(t)} + V_{y^k(t)}^T z(t)$$

"Recurrent Support Vector Machines For Slot Tagging In Spoken Language Understanding", *Yangyang Shi, NAACL 2016*

Evaluation metric

- F1 score: harmonic mean of precision and recall
- Commonly used in information retrieval, information extraction etc

Query:	check	weather	in	seattle	at	tomorrow	morning
Truth:	O	O	O	LOC	O	DATE	TIME
Predict:	O	O	O	LOC	O	TIME	O

Precision = 1 / 2 (hypothesized two slots, got 1 correct)

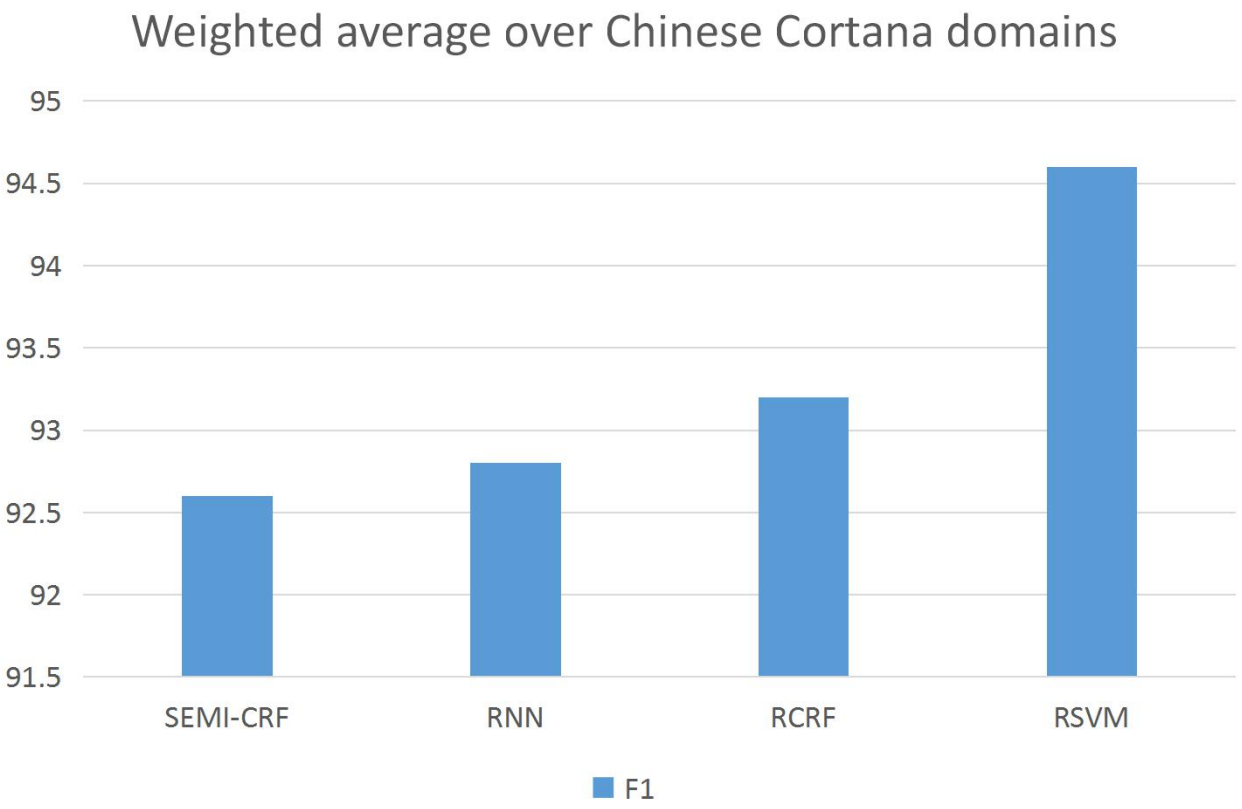
Recall = 1 / 3 (3 slots in reference, got 1 correct)

F1 = 2 * (precision * recall) / (precision + recall)

= 2 * (0.5 * 0.33) / (0.5 + 0.33)

Perfect F1 = 1

Slot tagging results



RNNLM data generation

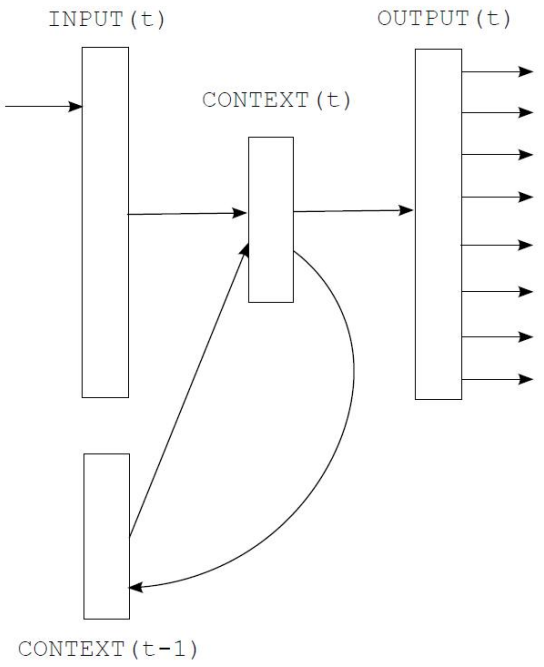
- In spoken language understanding, manually labeled data are usually required for model training
  - Domain & intent classification, slot tagging
- Manually labeled data are limited
- With manually labeled data, can we learn a probability distribution to generate synthetic labeled data?
  - Use recurrent neural network language model (RNNLM)
  - [Tam 2016]
- Can synthetic labeled data help?

RNNLM background

- Given word history, predict the next word:
  $P(w_i|w_1 w_2 \dots w_{i-1}) \approx P(w_i|h_{i-1}, w_{i-1})$

$= P(c_i|h_{i-1}, w_{i-1}) \times P(w_i|h_{i-1}, w_{i-1}, c_i)$

- "Generative" by nature
- Sample text from RNNLM to train an N-gram LM [Deoras 2011]
- Improves speech recognition accuracy [Mikolov 2010, 2011]
- In this work: Use RNNLM to generate synthetic labeled data



SLU task: Slot tagging

- Input: a word sequence $W = w_1 w_2 \ldots w_N$
- Output: a labeled sequence $Y = y_1 y_2 \ldots y_N$

W=	I	Want	To	Fly	From	San	Francisco	To	Seattle
Y=	O	O	O	O	O	B_FROM_CITY	E_FROM_CITY	O	B_TO_CITY

Different views:

- Slot tagging: P(Y | W)
- Language modeling: P(W)
- Labeled data generation: P(W, Y)

RNNLM labeled data generation for slot

- View $(w:y)_i$ pair as a token, W:Y as a sequence

W	I	Want	To	Fly	From	San	Francisco	To	Seattle
Y	O	O	O	O	O	B_FROM_CITY	E_FROM_CITY	O	B_TO_CITY
W:Y	I:O	Want:O	To:O	Fly:O	From:O	San:B_FROM_CITY	Francisco:E_FROM_CITY	To:O	Seattle:B_TO_CITY

- Use RNNLM to approximate P(W, Y) ≈ P(W:Y)

$$P((w:y)_i|(w:y)_{i-1}, h_{i-1})$$

- Proposal: Given some labeled data {W, Y}
  - Sample synthetic labeled data set: $\{(W:Y)_j\} \sim P(W:Y)$
  - Augment with labeled training set: Filter($\{W_j, Y_j\}$) + {W, Y}
  - Train a classifier using the augmented training set

Advantage of RNNLM data generation

- An easy way to inject “RNN” knowledge into any task and existing system
  - RNNLM model ⇔ RNNLM-generated data
  - No code change
- May provide combination effect at the data level
  - e.g. conditional random field enhanced by RNN for slot tagging
  - e.g. SVM enhanced by RNN for domain classification
- Extension: sentence template generation followed by human validation

Example artificial data in calendar

我 的 星期:B_start_date 五:E_start_date 有 什么 安排
建立 日程 明天:start_date 上午:B_start_time 11:I_start_time: 点:E_start_time 的 电话
把 今天:original_start_date 上午:B_original_start_time 十:I_original_start_time 点:E_original_start_time 的 会 改 到 下午:B_start_time 三:I_start_time 点:E_start_time
请 给 我 今天:start_date 的 安排

Slot tagging results

- RNNLM generated queries helped
- Semi-CRF tagger trained with RNNLM generated queries performed better than sequential RNN tagger

Table 7: Slot tagging F1 scores on the Chinese test set.

Data setup	Reminder
Train	98.5
Train + 1M	99.02
Train + 5M	**99.06**
Sequential RNN	98.84

Table 8: Slot tagging F1 scores on the Italian test sets.

Data setup	Comm	Reminder	Weather
Train	93.57	90.57	97.21
Train + 1M	**95.41**	**93.44**	**97.93**
Sequential RNN	94.75	92.25	97.72

Contextual Language Understanding

- Dialog session can be highly contextual:
  - Turn 1: what's the weather like today? Answer: It's sunny with …
  - Turn 2: and for the weekend?

  - Turn 1: postpone my meeting with John. Answer: Okay, how do you want to postpone your meeting?
  - Turn 2: from 2 pm to 4 pm.

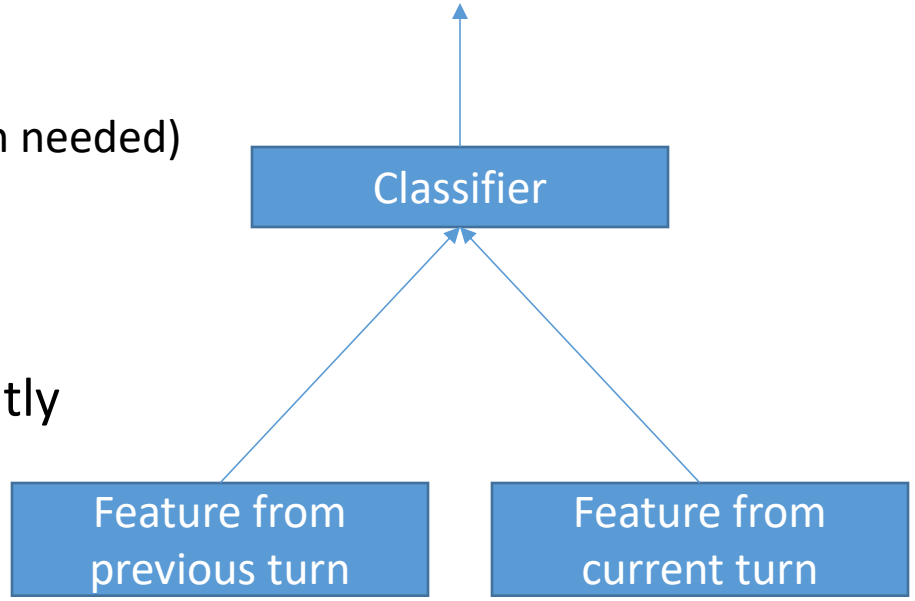
- How do we know if previous user intent should be carried over to the current turn?
  - Handle in LU
  - Handle in dialog manager

Build a contextual intent classifier

- Idea: Previous turn can be used as features in contextual intent classification

- Feature from previous turn:
  - Word N-gram features (dialog session needed)
  - Previous intent label
  - Previous system action label

- Feature representation are not jointly optimized



Jointly optimized language understanding

- Deep Contextual Language Understanding in Spoken Dialogue Systems [Liu 2015]
- Motivation:
  - Knowing the slot labels may help intent classification
    - “给老爷忘情水”
  - Intent classification & slot tagging are unified via multi-task learning
  - Previous turn features are treated as recurrent features (like RNN)
- Intent & slot models are joined together using neural network
  - Models can be optimized jointly using backpropagation

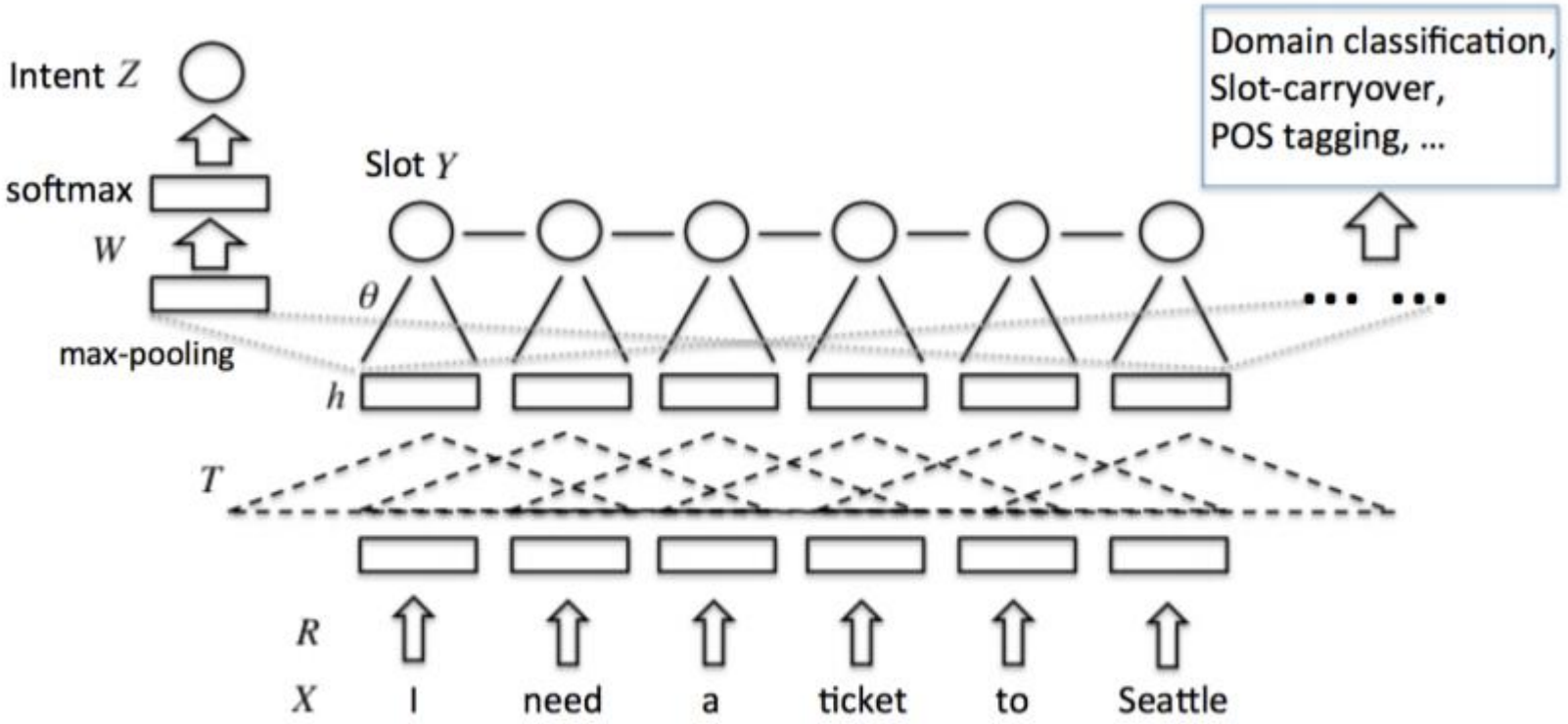


Figure 1: A CNN based multi-task learning framework for intent detection and slot filling. The same shared feature layer can be used for any classification or labeling tasks.

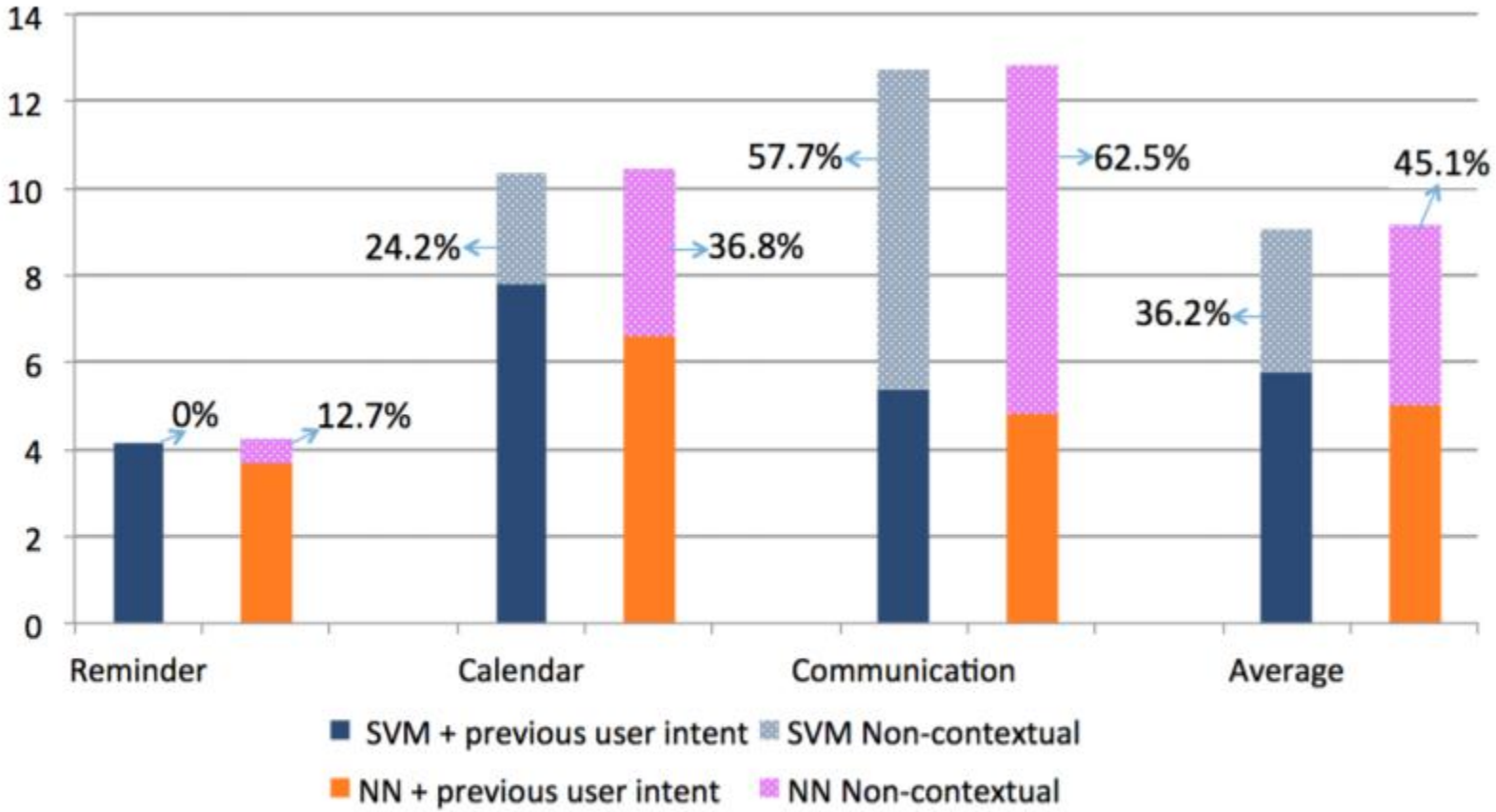


Figure 4: For both SVM and Joint CNN, intent error rates on follow-up turns with and without adding previous user intent features are shown for each domain, and the corresponding relative intent error rate reductions denoted in percentage.

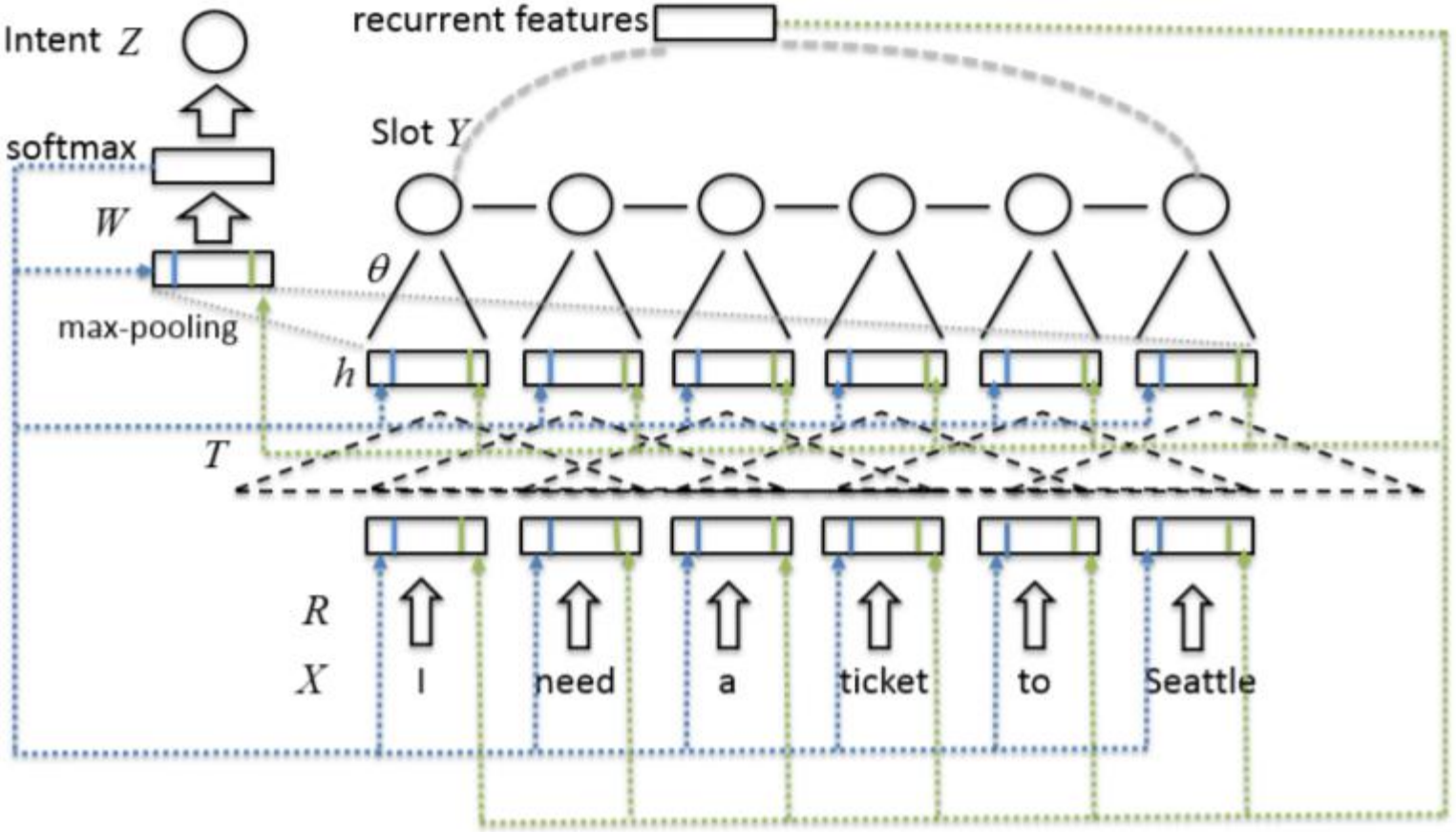


Figure 2: A RCNN based framework for contextual intent detection and slot filling. Feedback connections are added from intent and slot components back to the shared feature layers.

Table 1: *Intent error rate and slot filling F1 comparisons of adding recurrent connections from previous intent and slot outputs (evaluated on follow-up turns only).*

Domain	Task	NN non-contextual	NN + previous user intent	NN + previous user slot	NN + both
Reminder	Intent error	4.27	3.73	3.97	3.73
	Slot F1	84.58	84.65	85.83	85.84
Calendar	Intent error	10.49	6.64	9.90	6.58
	Slot F1	85.85	85.40	85.44	85.42
Communication	Intent error	12.90	4.90	11.18	4.89
	Slot F1	85.64	86.21	86.73	86.73
Average	Intent error	9.22	5.09	8.35	5.07
	Slot F1	85.36	85.42	86.00	86.00

Table 2: *Intent error rate and slot filling F1 score comparisons of adding external connections from dialog system response (evaluated on a subset of follow-up turns that have non-null previous system action features).*

Domain	Task	NN non-contextual	NN + previous system action	NN + previous user intent, slot + previous system action
Reminder	Intent error	4.04	3.36	3.36
	Slot F1	86.81	87.69	88.01
Calendar	Intent error	10.08	7.75	6.20
	Slot F1	86.55	87.39	87.30
Communication	Intent error	14.02	7.23	6.50
	Slot F1	84.66	87.00	88.49
Average	Intent error	9.38	6.11	5.35
	Slot F1	86.01	87.36	87.93

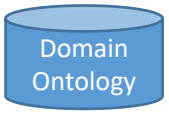
Dialog Manager (DM)

- Controls the conversation flow between human and bot
- Tracks required information (Semantic frame) to complete a task
  - Semantic frame: Intent + required slots
  - Slot carry-over, intent carry-over strategy
- Perform system action (DialogAct) at each turn
  - Request for a slot value
  - Ask for confirmation
  - Query an external database
- Rule-based DM
- Statistical-based DM (Markov Decision Process)

Slot Key	Value
Intent	Book Flight
Date	null
From	null
To	null

Slot Key	Value
Intent	Weather
Date	
Location	

Example of task completion chatbot



Slot Key	Value
Intent	Book Flight
Date	09-20-2016
From	Beijing
To	Shanghai

Slot Key	Value
Intent	Weather
Date	09-20-2016
Location	Shanghai

Key Technologies:

- Intent classification
- Slot filling
- Multi-initiative context management

Dialog State Tracking

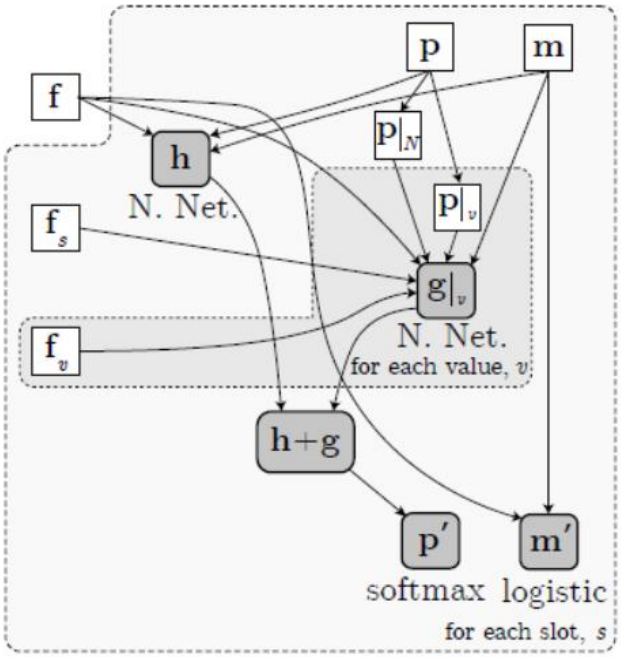
- Goal: To build machine learning model to track values of slots
- State space: {Slot=value} pairs
- Dialog State Tracking Challenges (DST, DST2, DST3, DST4)
- DST2: restaurant domain (search for restaurants given various criteria: e.g. price_range = cheap, location = midtown)

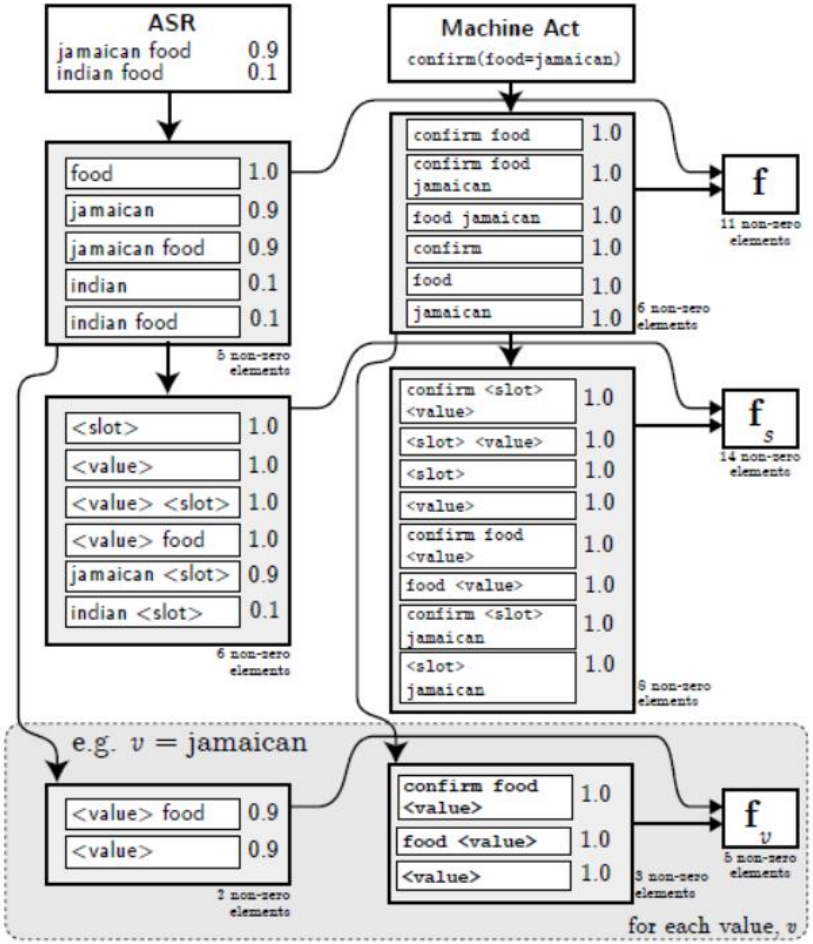
Dialog session annotation

User query	Machine Act	Food_type	Price_range	Location
	Welcome to Cambridge restaurant reservation system. How may I help you?	Null	Null	Null
I want to have some Italian food	What's your price range?	Italian	Null	Null
Cheap	Okay, what is your preferred location?	Italian	cheap	Null

RNN-based dialog state tracker [Henderson 2014]

- Extract features in user query and machine act at each turn
- Feed the extracted features into RNN, optimizing the slot-value tracking accuracy





$$\mathbf{h} = \mathrm{NNet}\left(\mathbf{f} \oplus \mathbf{p} \oplus \mathbf{m}\right) \in \mathbb{R}^N$$

$$\mathbf{g}|_v = \mathrm{NNet}\left(\begin{array}{c} \mathbf{f} \oplus \mathbf{f}_s \oplus \mathbf{f}_v \oplus \\ \{\mathbf{p}|_v,\ \mathbf{p}|_N\} \oplus \mathbf{m} \end{array} \right) \in \mathbb{R}$$

$$\mathbf{p}' = \mathrm{softmax}\left([\mathbf{h} + \mathbf{g}] \oplus \{B\}\right) \in \mathbb{R}^{N+1}$$

$$\mathbf{m}' = \sigma\left(W_{m0}\mathbf{f} + W_{m1}\mathbf{m}\right) \in \mathbb{R}^{N_{\mathrm{mem}}}$$

Task Completion Platform: Background

- Many big companies invest in bot development platform
  - e.g. api.ai, LUIS.ai, wit.ai etc

- Goal: To allow bot developer to quickly and easily build bots without any machine learning or NLP background
  - Developers define business logic
  - We provide LU and customizable dialog engine

Task completion platform

- Goal: To provide a standard way to define task requirements in chatbot
- [Crook 2016]
- Idea similar to an earlier paper from Frank Seide on ClippyScript

- A bot developer fills a set of task forms in Json format
  - Each task form contains one action to accomplish (e.g. order pizza)
  - Each task contains a set of Parameters to fill (e.g. deliveryAddress, pizzaType, pizzaSize)
  - Task parameters may have dependency on each other (i.e. a DAG graph is formed)
  - Task is triggered when a query matches the query triggering list, or a match domain/intent labels set by the developer
  - Support task chaining (i.e. if one task is completed, parameter variables can be “exported” to the next related task)

Architecture

- Task forms are loaded to enable various tasks in bot

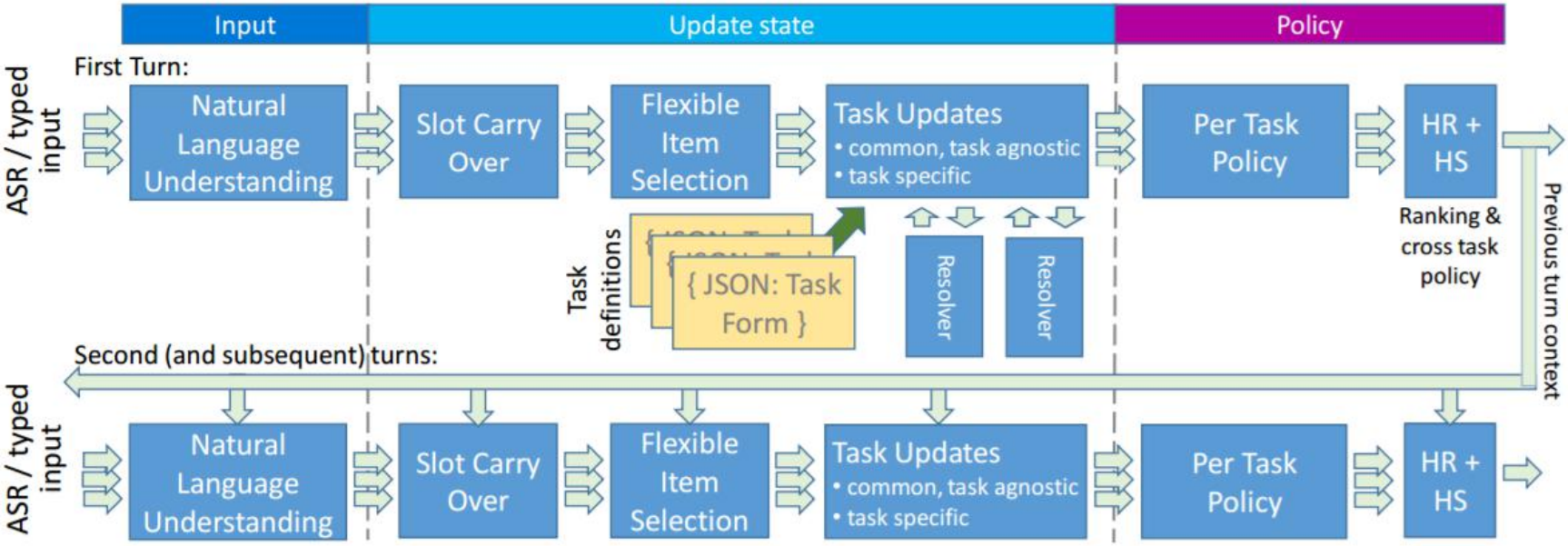


Figure 1: Functional modules of the TCP architecture

Task Completion Platform: A self-serve multi-domain goal oriented dialogue platform

Sample task definition (Order Pizza)

```
"TaskFormDefinition": {
  "Triggers": [
    "OrderPizzaIntent"
  ],
  "Parameters": [
    "SpecialtyPizza",
    "PizzaSize",
    "DeliveryAddress",
    "Contact",
    "PricedOrder"
  ],
  "FinalAction": "FinalAction",
  "TaskDialogActs": {
    "Welcome": "WelcomeDialogAct",
    "Cancellation": "CancellationDialogAct",
    "GlobalError": "GlobalErrorDialogAct"
  }
},
```

```
"TriggerDefinitions": [
  {
    "TriggerName": "OrderPizzaIntent",
    "Intent": "order_pizza",
    "TriggerQueries": [
      "i wanna order a pizza",
      "i want to order a pizza",
    ]
  },
],
...
```

```
"ParameterDefinitions": [
 {
   "ParameterName": "PizzaType",
   "ResolverInvocation": {
     "Resolver": "PizzaTypeResolver",
     "ResolverSlotTags": [ "cuisine", "product" ],
   },
   "DialogActs": {
     "MissingValue": "PizzaTypeMissingValue",
     "Disambiguation": "PizzaTypeDisambiguation"
   }
 },
 ...
```

Figure 1: A sample snippet of a task definition.

Task Completion Platform: A self-serve multi-domain goal oriented dialogue platform

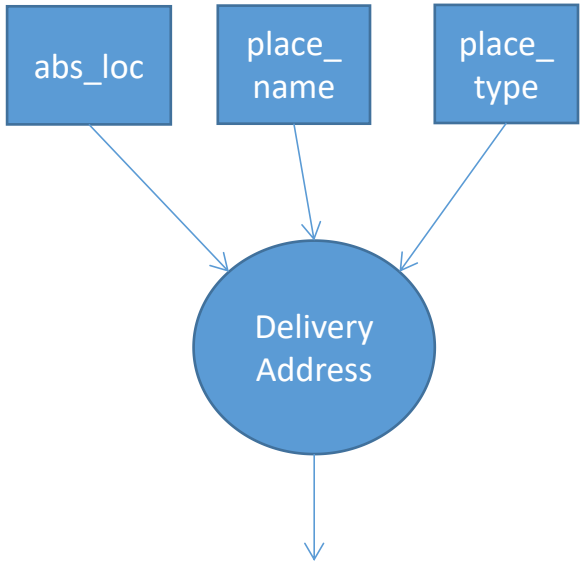
Task triggering

- Use results from LU domain, intent, slots (presence/absence), user-defined golden-path queries for task triggering
  - e.g If domain == places && intent==order_pizza || hit(golden query list, q), then trigger task

- Once a task is triggered, a dialogue hypothesis is either created or continued from the previous turn

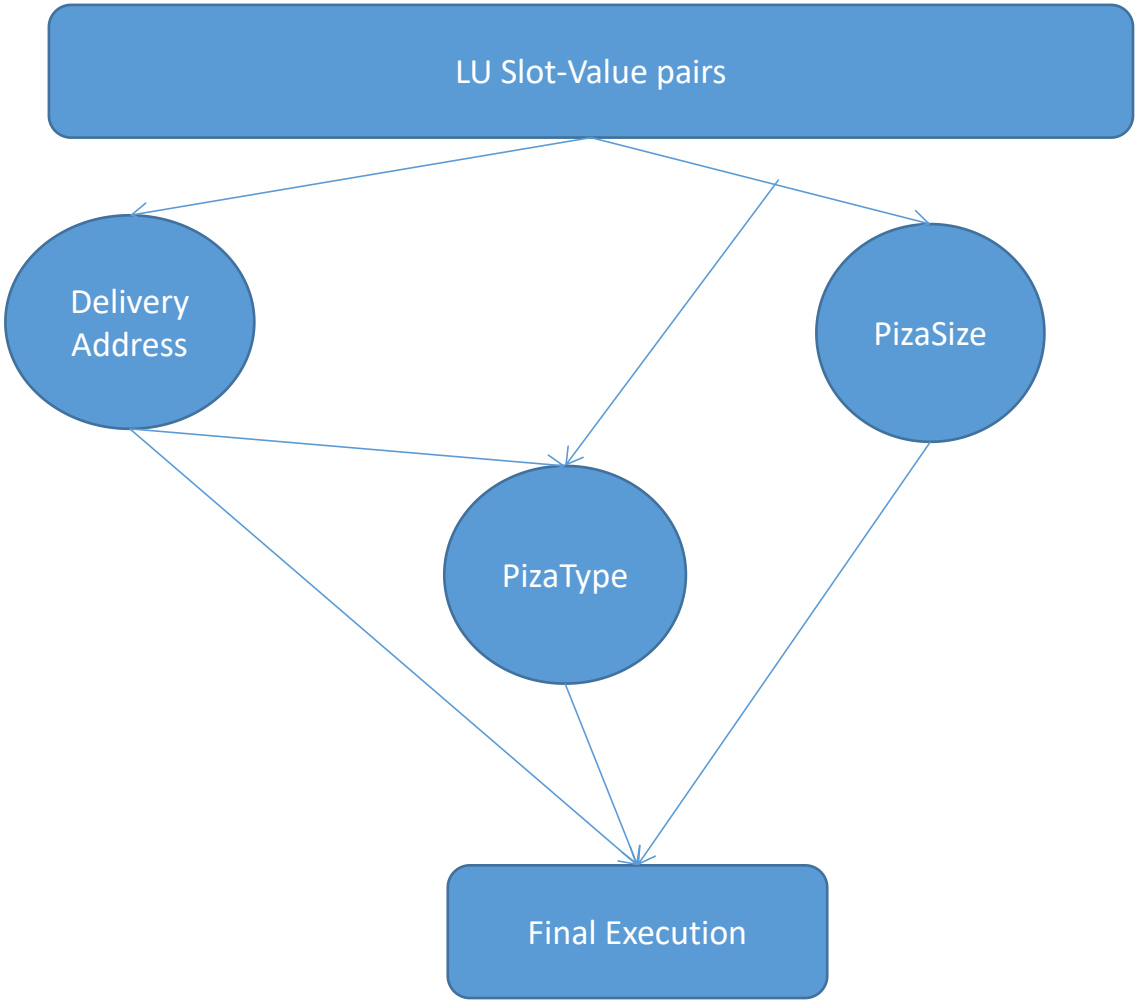
Task parameter

- A parameter depends on a set of LU slots, and other parameters
- Each parameter node has a resolver F({LU slots}, {Parameters})
  - resolver function call is implemented by bot developer using a SDK
  - maps LU text representation into object representation of a knowledge base (i.e. knowledge base access)
  - Each parameter node is associated with DialogActs (System actions)
    - MissingValue: Ask the user for input required to populate the parameter
    - NoResultsFollowUp: Prompt to change information as not results were found
    - Disambiguation: Ask the user to select the parameter value from a list
    - Confirmation: Ask for confirmation of the parameter value
    - ConfirmationFailure: If user says NO to the confirmation, ask user to redo



Parameter graph to describe a task

- Task parameters are topologically sorted
- Each parameter node is listening on the inputs
- Any parameter nodes can be executed in parallel once the input LU slots or parameter nodes are filled with values
- A dialog act is selected to present to user



Talk Outline

- Introduction to Wechat AI
- Chatbot development history
- Chatbot technologies
  - Task-oriented Personal Digital Assistant (PDA) (语音助手)
  - Chit-chat oriented Xianer monk (贤二机器僧)
- Challenges and future

Chit-chat chatbot

- Provide a general chit-chatting between user and bot
- Covers variety of topics
- Generally non-task driven (unlike personal digital assistant)
- Representative example: Microsoft XiaoIce
- Idea:
  - Maintain a large database of query-answer pairs
  - Given a test query, find/generate the best answer
- Popular approaches:
  - Information retrieval
  - Statistical machine translation (SMT)
  - Sequence-to-Sequence modeling

Related work

- An Information Retrieval Approach to Short Text Conversation [Ji 2014]
- Neural responding machine for short-text conversation [Shang 2015]
- DocChat: An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents [Yan 2016]
- And so many other papers developed on top of Seq2Seq

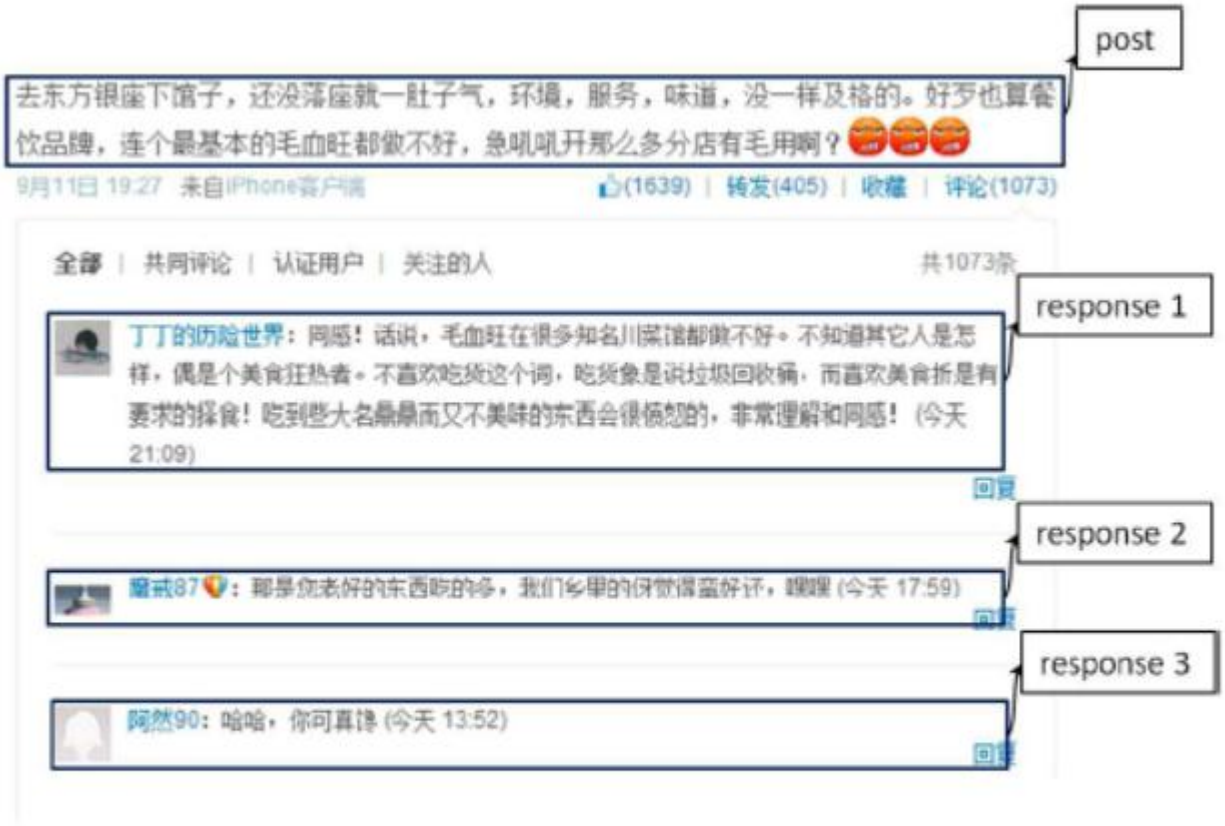


Figure 1: An example of post and associated comments at Weibo.

An Information Retrieval Approach to Short Text Conversation [Ji 2014]

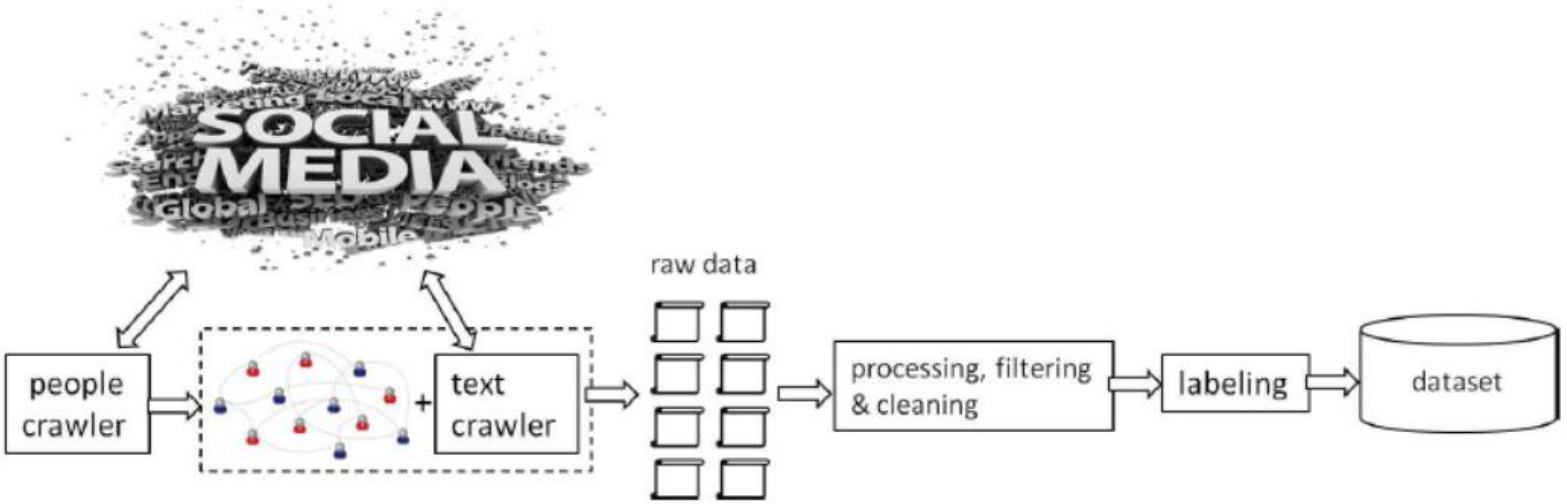


Figure 3: Diagram of the process for creating the original and the labeled post-comment pairs.

An Information Retrieval Approach to Short Text Conversation [Ji 2014]

Conversation as a search problem

- p: post
- r: response
- q: input query

$$r^* = \arg\max_{(p,r)} score(q, (p, r))$$

$$score(q, (p, r)) = \sum_{i \in \Omega} \omega_i \Phi_i(q, (p, r))$$

An Information Retrieval Approach to Short Text Conversation [Ji 2014]

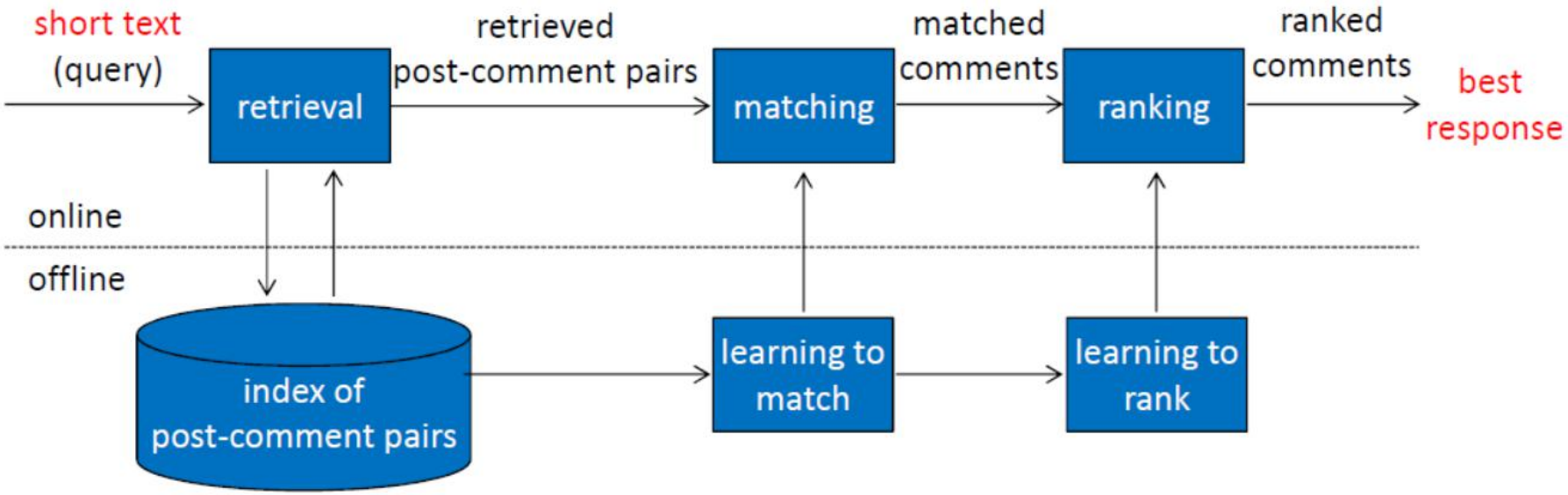


Figure 2: System architecture of retrieval-based short text conversation.

An Information Retrieval Approach to Short Text Conversation [Ji 2014]

Matching features

Query-Query similarity models:

- Sim(query, response)
- Sim(query, post)
- Deep matching model (neural network-based)
- And many more…

Sequence-to-Sequence approach

- End-to-end modeling: encoder and decoder
  - Encoder: encode a word sequence into a fixed-dimension vector
  - Decoder: decode a word sequence using the encoded vector
  - Encoder & decoder are jointly optimized using backpropagation
- Original motivation is for machine translation
  - Neural machine translation
- Attention mechanism is introduced to further improve performance
- Nowadays: the mainstream approach for machine translation
- Papers:
  - Sequence to Sequence Learning with Neural Networks [Sutskever 2014]
  - Neural machine translation by jointly learning to align and translate [Bahdanau 2015]

Sequence-to-Sequence modeling [Sutskever 2014]

- Each rectangle is a long-short term memory module (LSTM)

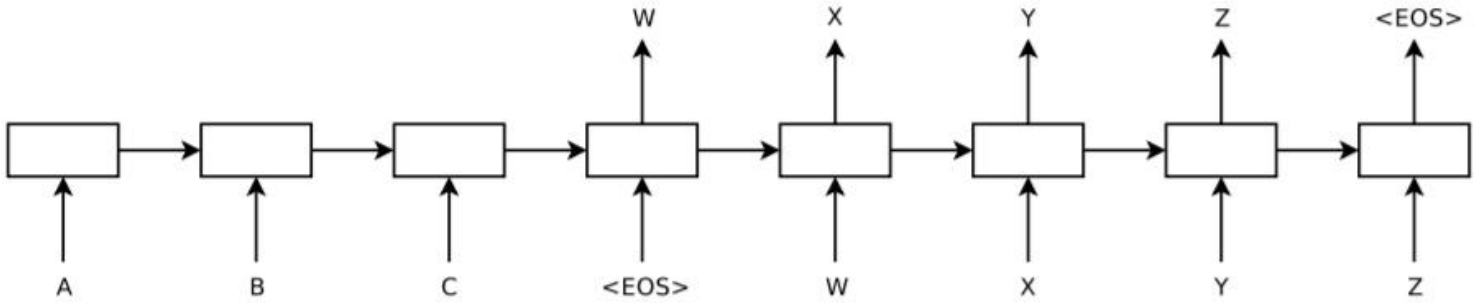


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

Attention modeling in NMT

- Encoder: Bidirectional LSTM
- Attention model: feed-forward neural network
  - At each decoding time step, it aligns to the input word position with a probability distribution
  - Proportional to similarity between decoder's hidden state and each encoded word embedding in an input sentence
- Decoder: LSTM
- End-to-end joint optimization

- Computation path for context vector c:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

$$a(s_{i-1}, h_j) = v_a^\top \tanh(W_a s_{i-1} + U_a h_j)$$

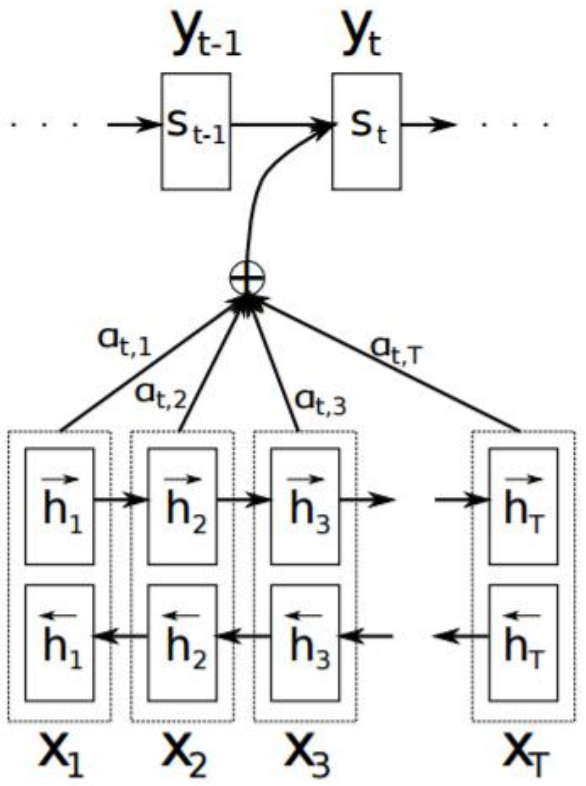
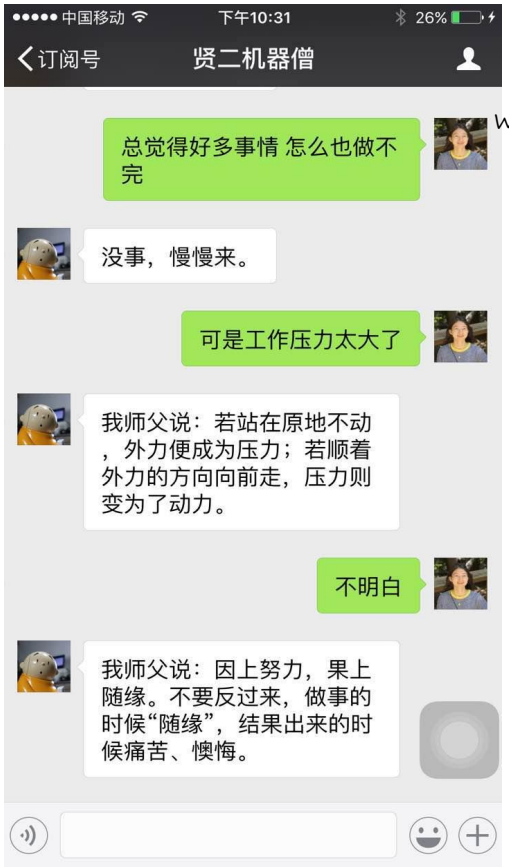


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

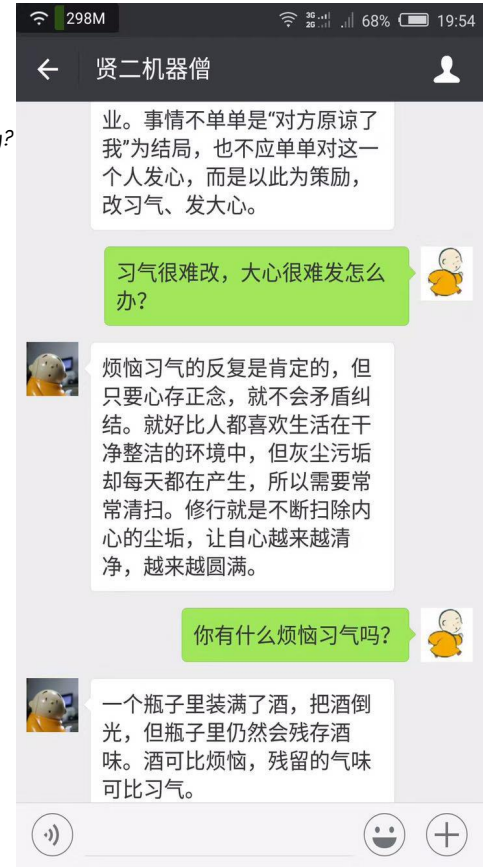
Xianer monk

- Mimick a spiritual teacher to help people relieve from life problems
- Collaborated with a Buddhist teacher who writes answers for us
- Currently exceeding 800k subscribers

How can be happy?



Why I'm so busy?



Technical focus

- Contextual modeling in conversation
- Query-Query similarity model

Contextual modeling in Xianer monk

Query rewriting to augment contextual information for better retrieval

→ **你去问问师父喜欢你吗**

→ **不会的，问你师父去**

→ **什么时候问必要**

Contextual query rewriting

Motivation: Use contextual queries to rewrite the current (potentially short) query into a “complete” query

Methods considered:

1. Replace pronouns with entity occurred in previous queries
2. Expand a short query into a long one that captures contextual information

- Leverage Seq2Seq and machine reading technique

Machine reading

- Given a question q, passage d, give an answer a

Original Version	Anonymised Version
Context	
The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the "Top Gear" host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon "to an unprovoked physical and verbal attack." ...	the *ent381* producer allegedly struck by *ent212* will not press charges against the " *ent153* " host , his lawyer said friday . *ent212* , who hosted one of the most - watched television shows in the world , was dropped by the *ent381* wednesday after an internal investigation by the *ent180* broadcaster found he had subjected producer *ent193* " to an unprovoked physical and verbal attack . " ...
Query	
Producer **X** will not press charges against Jeremy Clarkson, his lawyer says.	producer **X** will not press charges against *ent212* , his lawyer says .
Answer	
Oisin Tymon	*ent193*

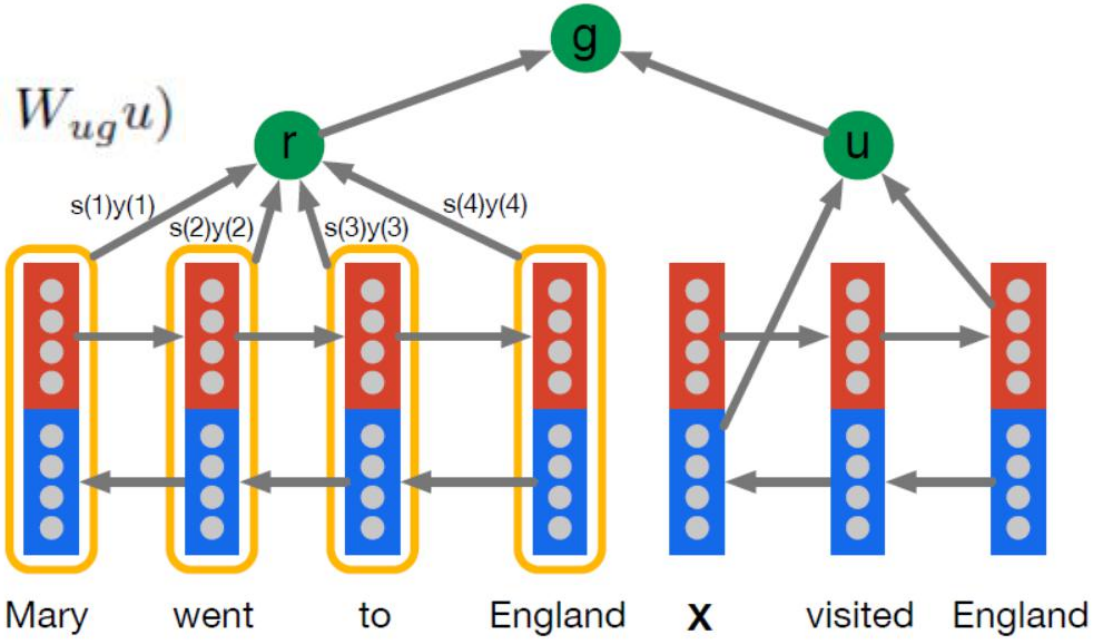
Table 3: Original and anonymised version of a data point from the Daily Mail validation set. The anonymised entity markers are constantly permuted during training and testing.

$$p(a|d,q) \propto \exp\left(W(a)g(d,q)\right), \quad \text{s.t. } a \in V,$$

Attentive reader

- Employ attention mechanism in Seq2Seq modeling

$$m(t) = \tanh\left(W_{ym} y_d(t) + W_{um} u\right),$$
$$s(t) \propto \exp\left(\mathrm{w}_{ms}^{\intercal} m(t)\right),$$
$$r = y_d s,$$
$$g^{\mathrm{AR}}(d, q) = \tanh\left(W_{rg} r + W_{ug} u\right)$$



Anaphora Resolution

$q' = H(q,c)$

Input:

q: current query

c: context

Output:

q': current query after anaphora resolution

H: replace pronouns in the current query with noun phrases in the context

About 5% of the total queries

Examples:

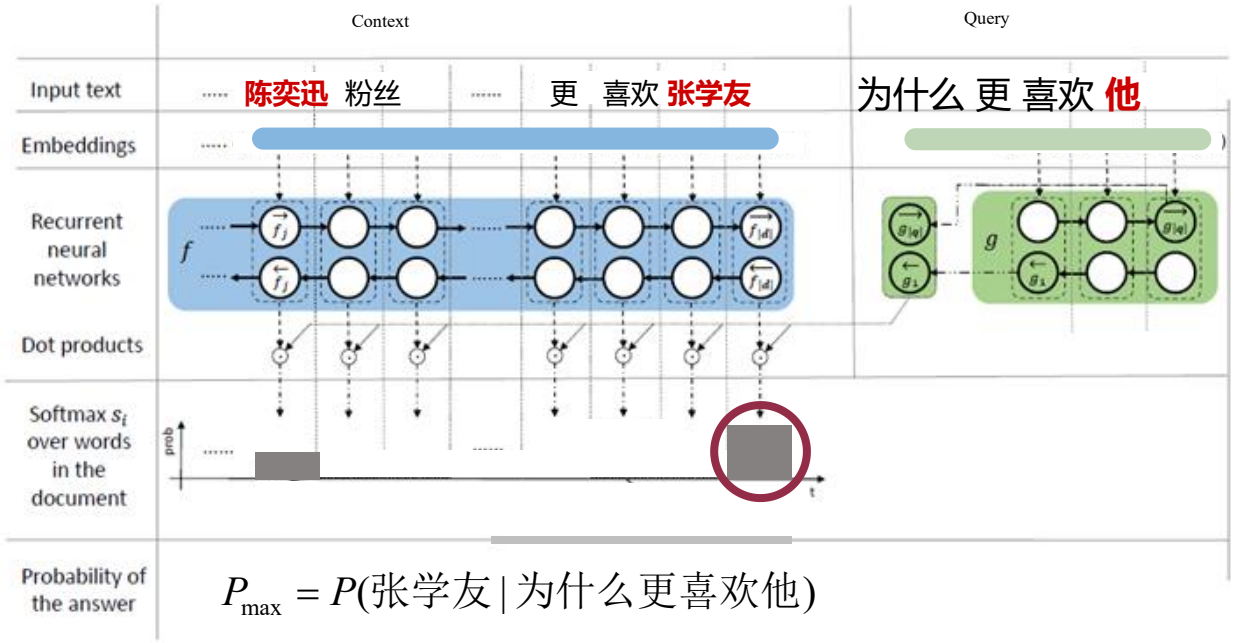
C1: 你是**陈奕迅**粉丝吗? (are you a fan of Eason Chan?)
C2: 更喜欢**张学友** (I like Jacky Cheung more)
q : 为什么更喜欢**他**？ (Why like him more?)
q‘: 为什么更喜欢**张学友** (Why like Jacky Cheung more?)

C1 : 你住哪儿? (where do you live?)
C2 : **不二寺**。 (Bu'er Temple)
q : **那**在哪儿？ (Where is it?)
q‘ : **不二寺**在哪儿？ (Where is Bu'er Temple ?)

Anaphora Resolution as machine reading

Example:

C1: 你是**陈奕迅**粉丝吗?
C2: 更喜欢**张学友**
q ：为什么更喜欢**他**？
q ‘: 为什么更喜欢**张学友**



$$P_{\max} = P(\text{张学友} \mid \text{为什么更喜欢他})$$

- 100K training data
- Accuracy: 90%
- Majority of the errors are caused by the mistakes of entity tagging

“他”(him) → “张学友”(Jacky Chueng)

q' = 为什么更喜欢张学友

A bad case:
C1: 你认识**贤三**吗?
C2: 当然认识。
q ：他是你什么人？
q ': **三**是你什么人？

Query Complement

$q' = H(q, C)$

Input:
q: current query
c: context

Output:
q': current query after query complement

H: complete the current query with information in the context

About 15% of the total queries

Examples:

C1: 那你会发表情包吗? (can you send emojis?)
C2: 一般**不发** (usually I don't send emojis)
q :为什么？ (Why?)
q': 为什么**不发表情包** (Why not send emojis?)

C1 :讲个故事给我听 (tell me a story)
C2 :等我学会了给你讲哦。 (I'll tell you a story once I learn how to)
q :我等着 (I'm waiting)
q': 我等着**听故事** (I'm waiting for the story)

Query Complement using Seq2Seq

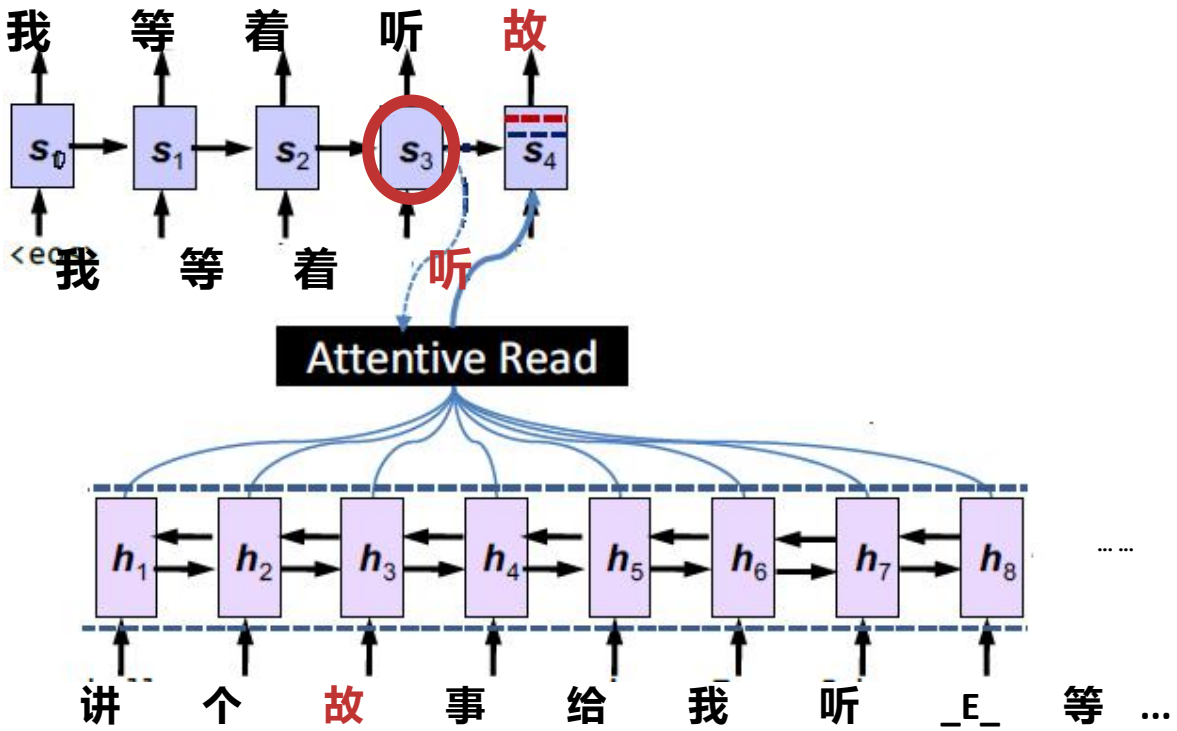
Training Sample:

C1:讲个故事给我听 C2:等我学会了给你讲哦。 x
q :我等着 q‘:我等着**听故事** y

$$\{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^N$$

$$\arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log P_{\mathbf{w}}(\mathbf{y}^i|\mathbf{x}^i)$$

- 100,000 training instances
- Accuracy: 70%
- Increased the engagement of Xian'er Mechanical Monk by 11%



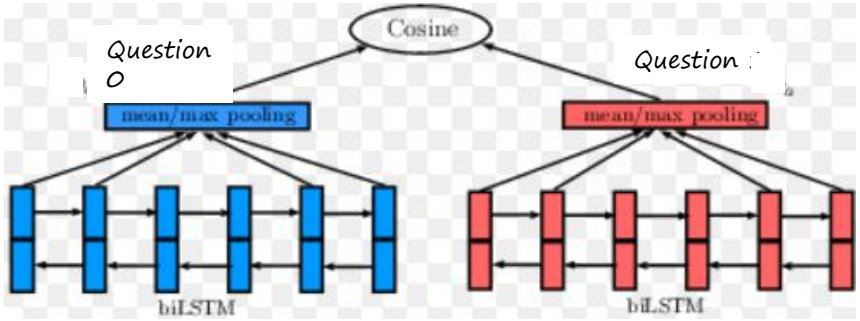
- Encode C1,C2,q
- Decode q'

Query-Query similarity issue

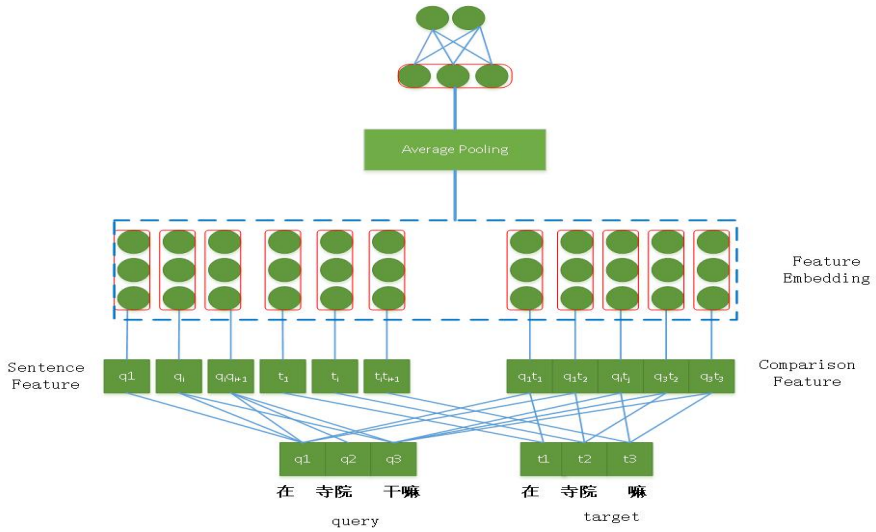
Unsupervised word embedding approach is not good enough

Sentence 0	Sentence 1	Similarity based on Word Embedding	Similar Enough?
你是谁 (who are you)	我是谁 (who am I)	0.93	No
我爱你 (I love you)	你爱我 (you love me)	0.89	No
吃饭了吗 (Do you have lunch?)	吃饭了 (just had lunch)	0.84	No
你干嘛的 (what is your job?)	你干嘛呢 (who are you doing?)	0.93	No
有轮回吗? (Is reincarnation true?)	轮回有结束吗 (will the cycle of life end?)	0.73	No
会不会轮回 (will reincarnation happen?)	会不会轮回结束 (Will reincarnation end?)	0.84	No
随喜您 (you did it well)	您做的很好 (you did it well)	0.20	Yes

Supervised Query-Query similarity modeli



RNN (CNN can be done in a similar manner)



Feature Embedding Model (Extension of Fasttext)

- Sentence features
  - unigrams
  - bi-grams
- Comparison Features
  - word pairs from two sentences each
    - edit operations
      - **什么 含义 vs. 什么 意思**
      - match-什么-什么
      - replace-含义-意思

Query Similarity Results

220,000 sentence pairs for training
20,000 for testing

Models	Accuracy
Unsupervised word embedding	0.63
RNN + cosine similarity	0.65
RNN + MLP	0.6878
CNN + MLP	0.6968
RNN + Tensor	0.728
Feature Embedding (bigram+cross word pairs)	0.75

Talk Outline

- Introduction to Wechat AI
- Chatbot development history
- Chatbot technologies
  - Task-oriented: Personal Digital Assistants (PDAs) (语音助手)
  - Chit-chat oriented: Xianer Monk
- Challenges and future

Challenges and future work

- Fundamental technical improvements
  - Deep contextual understanding
  - Flexible & customizable dialog management
  - knowledge representation & inference
- Personalization
  - Chatbot that knows your need (physical, emotional, spiritual)
- Domain expansion
  - Bank, food, music, medical etc
  - Accumulate more experience
- Different media
  - text, picture, video

Thank you!

Yik-Cheung (Wilson) Tam

wilsontam@tencent.com

Wechat AI

Tencent

Supplemental pages

Contextual Language Understanding Modeling

- Machine Learned Models

Improvements (Domain)

what's my schedule tomorrow **calendar**
what time is the first one **places → calendar**

what's the weather like today **weather**
what about tomorrow **web → weather**
and for the weekend **calendar → weather**

tacos places in seattle **places**
select rancho bravo tacos **web → places**

who went to the final four last year **web**
when is Michigan's next basketball game **calendar → web**

Improvements (Intent)

directions to home depot **get_route**
The one in Bellevue **find_place → get_route**

opening hours for home depot **get_hours**
The one in Bellevue **find_place → get_hours**

Flexible Selection: See-it-Say-it (SISI)

- Machine Learned Model

[Turn-1]: *"Hamburger places near me"*

((1))Five Guys Burgers and Fries

(2) Kidd Valley Burgers and Shakes

(3) Wibbley's Gourmet Hamburgers

(I) Explicit Referential (Turn 2)

"show five guys menu"

"directions to the five guys and fries"

(II) Implicit Referential

"the one on 24th street"

"the one in Redmond"

(III) Explicit and Implicit (Mixed) Referential

"five guys in redmond"

"the burger shop in redmond"

(IV) Explicit Locational

"look up hours for the top one"

"call the first (burger) place"

(V) Implicit Locational

"what time does the first one open"

"call the 1st one"

Multi-task bot

Initialization:

- All enabled tasks of an app are loaded

Test: Given a test query, LU is performed (intent classification, slot tagging)

- Context manager is accessed to restore semantic frame, task frame from previous turn
- Slot carry over is performed on previous semantic frame
- Parameter carry over is performed from previous task frame
- Execute triggered tasks
- Select the best task hypothesis according to system scores

Overall architecture