The Dissertation Committee for Nazneen Fatema Rajani
certifies that this is the approved version of the following dissertation:

Explainable Improved Ensembling for

Natural Language and Vision

Committee:

Raymond J. Mooney, Supervisor

Katrin Erk

Greg Durrett

Ken Barker

Explainable Improved Ensembling for Natural Language and Vision

by

Nazneen Fatema Rajani

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2018

To my mother, father and beloved husband, Zahed Reza.

“You presume you are a small entity, but within you is enfolded the entire universe.”

— Ali ibn Abu Talib

Acknowledgments

At the outset, I would like to thank my advisor Ray Mooney for being a supportive guide and coach to me during the last few years. I admire Rays empiricist attitude and his passion for the advancement of open science. I thoroughly enjoyed discussing intellectually exciting ideas during my weekly meetings with him. I genuinely appreciate Ray for working flexibly with me.

I am also grateful to the rest of my committee members, Katrin Erk, Greg Durrett and Ken Barker for their valuable feedback on my dissertation. I look up to Katrin for her ability to balance her work and family in a stellar way. It would be very remiss of me not also to mention Jason Baldridge, my Masters thesis advisor. Jason continues to be a mentor and inspiration for me.

My graduate school journey would never have been the same without my friends and lab-mates. I am thankful to Stephen Roller, Karl Pichotta, Iz Beltagy, Aishwarya Padmakumar, Jesse Thomason, Eric Holgate, Elisa Ferracane and Chris Brown for making my experience memorable.

I owe a debt of gratitude to the community at Islamic Ahlul Bayt Association (IABA), Austin for making me feel at home when I moved thousands of miles away from home to pursue graduate school at the University of Texas.

I owe what I am today to my father. His constant encouragement has always motivated me to strive to be a better version of myself. I am also thankful to my mother for being very helpful and supportive throughout my career. My younger brother and sister for all the good times we spent together.

There is no doubt that my Ph.D. would not have been possible without the support of my loving husband. He has been by my side through the good times and the bad times. My daughter, Zainab, who is five months old at the time of writing this thesis, has made my graduate school experience joyful.

The work done in this thesis was supported by the DARPA DEFT program under the AFRL grant FA8750-13-2-0026 and the DARPA XAI program.

Explainable Improved Ensembling for Natural Language and Vision

Nazneen Fatema Rajani, Ph.D.
The University of Texas at Austin, 2018

Supervisor: Raymond J. Mooney

Ensemble methods are well-known in machine learning for improving prediction accuracy. However, they do not adequately discriminate among underlying component models. The measure of how good a model is can sometimes be estimated from "why" it made a specific prediction. We propose a novel approach called Stacking With Auxiliary Features (SWAF) that effectively leverages component models by integrating such relevant information from context to improve ensembling. Using auxiliary features, our algorithm learns to rely on systems that not just agree on an output prediction but also the source or origin of that output.

We demonstrate our approach to challenging structured prediction problems in Natural Language Processing and Vision including Information Extraction, Object Detection, and Visual Question Answering. We also present a variant of SWAF for combining systems that do not have training data in an unsupervised ensemble with systems that do have training data. Our combined approach obtains a new state-of-the-art, beating our prior performance on Information Extraction.

The state-of-the-art systems on many AI applications are ensembles of deep-learning models. These models are hard to interpret and can sometimes make odd mistakes. Explanations make AI systems more transparent and also justify their predictions. We propose a scalable approach to generate visual explanations for ensemble methods using the localization maps of the component systems. Crowd-sourced human evaluation on two new metrics indicates that our ensemble's explanation significantly qualitatively outperforms individual systems' explanations.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

Ensembling multiple systems is a well known standard approach to improving accuracy in several machine learning applications (Dietterich, 2000). Ensembles have been applied to a wide variety of problems in all domains of Artificial Intelligence including Natural Language Processing (NLP) and Computer Vision. In NLP, these have been applied to Parsing (Henderson and Brill, 1999), Word Sense Disambiguation (WSD) (Pedersen, 2000), Sentiment Analysis (Whitehead and Yaeger, 2010) and Information Extraction (IE) (Florian et al., 2003; McClosky et al., 2012). In the domain of Computer Vision, ensembles have been used for Image Classification (Korytkowski et al., 2016), Object Tracking (Zhou et al., 2014), Object Detection (Malisiewicz et al., 2011) and Zero-shot Recognition (Jayaraman and Grauman, 2014). However, these techniques do not learn to adequately discriminate across the component systems and thus are unable to integrate them optimally. We seek to integrate knowledge from multiple sources to improve ensembling using a new approach we call Stacking With Auxiliary Features (SWAF) (Rajani and Mooney, 2017b). Stacking (Wolpert, 1992) uses supervised learning to train a meta-classifier to combine multiple system outputs. The auxiliary features enable the stacker to fuse additional relevant knowledge from multiple systems and thus leverage them to improve prediction.

We consider the general machine learning problem of combining structured outputs from multiple systems to improve accuracy by using auxiliary features. We propose two types of auxiliary features – those that enable the stacker to discriminate across instance types which we call, the *instance features* and those that enable the stacker to discriminate across component systems which we call, the *provenance features*. SWAF can be successfully deployed to any problem whose output instances have confidence scores and optionally *provenance* that justifies the output. Provenance indicates the origin of the generated output and can be used to measure the reliability of system output. The idea behind using auxiliary features is that an output is more reliable if not just multiple systems produce it but also agree on its provenance, and there are sufficient supporting instance features. The SWAF algorithm requires identifying the instance and provenance features for a given task.

We demonstrate SWAF on a variety of machine learning problems in the areas of natural language and vision. All the tasks are difficult and well-known challenge problems. We obtain new state-of-the-art results on two Knowledge Base Population (KBP) tasks – (i) relation extraction and (ii) entity linking (Rajani et al., 2015; Rajani and Mooney, 2017b). For Object detection and Visual Question Answering (VQA) problems in vision, our SWAF approach beats all the individual and ensemble systems considered in our experimental setup (Rajani and Mooney, 2017b, 2018). We propose auxiliary features for each of the aforementioned tasks that enable the stacking meta-classifier to exploit additional relevant knowledge of both the component systems and the problem.

Stacking uses supervised learning and requires training data to ensemble multiple systems. However, we would sometimes like to ensemble systems for which we have no historical performance data. Unsupervised ensembling methods such as voting and constrained optimization techniques (Wang et al., 2013) have been developed for this scenario. However, such methods fail to exploit supervision for those systems for which we *do* have training data. We designed a novel variant of SWAF that uses supervised *and* unsupervised ensembling to exploit the advantages of both (Rajani and Mooney, 2016). We first combine systems without training data into an unsupervised ensemble. Next, we use stacking to combine this unsupervised ensemble with other systems with available training data. SWAF can, therefore, handle both systems with and without training data for learning an ensemble.

Although there have been several innovative and ground-breaking ideas deployed to solve VQA problems, the current state-of-the-art on real-world images is still approximately 12 points behind human accuracy.[1] One way to reduce this gap in performance is to analyze how various neural architectures arrive at their

[1]Based on the performance reported on the CodaLab Leader-board and human performance reported on the task in (Antol et al., 2015).

predicted answers, and then design heuristics or loss functions that overcome the shortcomings of current networks. Also, systems that can explain their decisions make them more trustworthy, transparent, and user-friendly (Aha et al., 2017; Gunning, 2016). This has led to some work in generating explanations that help interpret the decisions made by CNNs (Goyal et al., 2016; Hendricks et al., 2016; Park et al., 2016; Ross et al., 2017). However, previous work focuses on generating explanations for individual models even though the top performing systems on various computer vision and language tasks are ensembles of multiple models. This motivated us to explore the problem of generating explanations for an ensemble using explanations from underlying individual models as input.

VQA systems have been shown to attend to relevant parts of the image when answering a question (Goyal et al., 2016). The regions of an image on which a model focuses can be thought of as a visual explanation for that image-question (IQ) pair. The Grad-CAM algorithm highlights the regions in an image that the model focuses on by generating a heat-map with intensity gradients (Selvaraju et al., 2017). We adapt their approach on three individual VQA models to obtain their explanations maps. We then propose two novel methods for generating visual explanation for our SWAF ensemble: (i) the weighted average and (ii) the penalized weighted average. These methods generate explanations by ensembling visual explanations of the component models.

Evaluating AI-system explanations is a challenging problem that has attracted attention in recent years (Samek et al., 2017; Ribeiro et al., 2016; Park et al., 2016; Das et al., 2017a). The work in this area uses crowd-sourcing to evaluate explanations. However, most of these evaluations measure the extent to which a machine-generated explanation overlaps with a human-generated explanation, considering human explanation as the ground truth. This has several disadvantages. Research shows that human and deep-learning models do not attend to the same input evidence even when they produce the same output (Das et al., 2017a). To aid interpretability and trust, machine explanations should accurately reflect the system's reasoning rather than try to produce "post-hoc rationalizations" that mimic human explanations and might convince users to *mistakenly* trust its results. Consequently, we propose two novel evaluation methods for judging the quality of explanations: (i) the comparison metric and (ii) the uncovering metric. The comparison metric evaluates explanations by asking human subjects to compare and score two machine generated explanations side-by-side. The uncovering metric measures how accurately a human subject can arrive at the same decision as the system using only the information from a system-generated explanation. Results on crowd-sourced evaluation using both these metrics indicate that our ensemble explanation significantly qualitatively outperforms the explanation of individual component models.

1.1 Thesis Outline

The remainder of this thesis is organized as follows:

In Chapter 2, we review the background information crucial to understanding this thesis, including a brief discussion of well-known ensembling techniques in machine learning. We also introduce each of relevant applications considered in this thesis, including Slot-filling, Entity linking, Object detection and Visual Question Answering. This chapter also discusses related work in each of these applications.

In Chapter 3, we introduce a novel ensembling approach, Stacking With Auxiliary Features (SWAF). We propose two types of auxiliary features: (i) the instance features and (ii) the provenance features. We also introduce a new variant of SWAF that combines systems that do *not* have training data with systems that *do* have training data.

In Chapter 4, we demonstrate our SWAF approach on two Knowledge Base Population (KBP) problems: (i) slot-filling and (ii) entity linking. We define the auxiliary features for each of the two tasks and discuss how we ensemble diverse systems with structured outputs. We then extensively compare our approach to several top-ranking individual and ensemble models and show that our approach outperforms each of the systems considered in our experimental setup. We also demonstrate a variant of SWAF for combining supervised and unsupervised systems on the two KBP tasks and show that it does better than using just the supervised systems.

In Chapter 5, we demonstrate SWAF on two well-known computer vision problems: (i) Object detection and (ii) Visual Question Answering (VQA). We identify auxiliary features for each of these problems and for VQA, we also introduce the idea of using visual explanations from models as auxiliary features. We present results on the 2015 version of the ImageNet object detection task and the 2016 version of the VQA task.

In Chapter 6, we propose a novel solution to generate explanations for an ensemble system by ensembling visual explanations of component models for VQA. We also propose two new evaluation metrics: (i) the comparison metric and (ii) the uncovering metric. These metrics measure how good a visual explanation is without requiring human-generated gold standard.

In Chapter 7, we propose some future directions for our research and finally, in Chapter 8, we summarize our contributions and findings.

1.2 List of Contributions

In this thesis, we make the following contributions:

Stacking With Auxiliary Features (SWAF) as an ensemble approach

We introduce a new ensembling algorithm, Stacking with Auxiliary Features (SWAF), that extends the idea of stacking proposed by Wolpert (1992) to also include other system and task-specific features along with confidence scores of the base systems. The meta-classifier learns to combine outputs of multiple systems using features of the component models and current problem as context.

Auxiliary Features

We propose two categories of auxiliary features: (i) the instance features and (ii) the provenance features. The instance features enable the stacker to discriminate between input instances, whereas the provenance features enable the stacker to discriminate between the component systems.

Variant of SWAF for combining supervised and unsupervised systems

We introduce a variant of SWAF that can ensemble not just models that have training data but also models without training data. We first use unsupervised ensembling to combine systems without training data, and then use stacking to combine this ensembled system with other systems for which training data is available.

Demonstrate SWAF on NLP problems

We demonstrate our SWAF approach on two Knowledge Base Population (KBP) tasks: (i) relation extraction and (ii) entity linking. We propose instance and provenance features for both these tasks and do a comprehensive analysis to show how each of these feature types affects the performance of SWAF. We also demonstrate the SWAF variant that combines supervised systems with an unsupervised ensemble on the KBP tasks.

Demonstrate SWAF on Vision problems

We extend the idea of SWAF to computer vision as well by demonstrating it on object detection and Visual Question Answering (VQA). We show how the notion of provenance can be extended to these two vision problems, in particular, how the idea of provenance ties in with the idea of visual explanation for VQA.

Visual explanation as auxiliary feature for SWAF

We propose the idea of using visual explanation as auxiliary features for VQA. The idea is that an output is reliable if not just multiple systems agree on the output prediction but they also agree with its explanation. Our work demonstrates that explanation is not limited to building human trust on machines and show how it can be used with other auxiliary features to improve the accuracy of VQA.

Generate visual explanation for ensemble models

We introduce a novel solution for generating visual explanations for ensemble systems by ensembling the explanation of the component models. Our approach produces explanation maps that are qualitatively better and less noisy than the individual model's explanation map. We generate explanations using two methods: (i) the weighted average and (ii) the penalized weighted average.

Two novel metrics for evaluating visual explanations

We evaluate visual explanations using two new metrics: (i) the comparison metric and (ii) the uncovering metric. Both our evaluation metrics use crowdsourcing but do not rely on the human-generated gold standard. The comparison metric compares two machine-generated visual explanations and ranks them based on the quality of the generated explanation. On the other hand, the uncovering metric judges whether the input evidence highlighted by a visual explanation is sufficient to allow a human judge to arrive at the same prediction as the model that produced it.

Chapter 2
Background and Related Work

2.1 Chapter Overview

In this chapter, we discuss the background and related work critical to this dissertation. We begin by reviewing the various ensembling methodologies in machine learning relevant to our thesis. We then discuss the two Knowledge Base Population (KBP) problems of Relation Extraction and Entity Linking. After that, we overview the Object Detection and Visual Question Answering (VQA) tasks in computer vision. Finally, we discuss the problem and challenges of producing Explainable AI (XAI) systems as well as relevant work in this area.

2.2 Ensemble Algorithms

Ensemble methods are models composed of multiple component models that are independently trained and whose predictions are combined in a certain pre-defined way to make the overall prediction. These methods vary in the ways the component models are combined and what types of models are combined. Ensemble algorithms are very powerful and therefore well known. Some of these popular types of ensembling algorithms are:

- **Boosting** (Freund and Schapire, 1995) refers to training a model on a set of data and then creating a second model that attempts to correct the errors of the first model. In this way, models are added until the training set is predicted perfectly or a maximum number of models are added. Boosting focuses on reducing the bias.
- **Bagging** (Breiman, 1996) or Bootstrap Aggregation refers to the prediction by generating additional data for training using repetitions to produce multi-sets of the same cardinality as the original data. Bagging focuses on reducing

the variance.

- **Stacking** (Wolpert, 1992) refers to building a meta-classifier on top of the component models to estimate the decision based on the outputs produced by the base models. In Stacking, the meta-classifier is trained on a held out dataset, such as the validation set, that is disjoint from the training data of the component models. The input to the meta-classifier is the confidence score of the component models and the output is a binary decision on each instance of the data.
- **Random Forest** (Liaw and Wiener, 2002) refers to the weighted combination of decision trees trained on random subsets of the data.

In this thesis, we use stacking as our ensembling algorithm and extend it to make it even more powerful by using task-specific features which we call *auxiliary features*.

In the past, many new state-of-the-art ensembling algorithms have been proposed. Bipartite Graph-Based Consensus Maximization (BGCM) is one such algorithm (Gao et al., 2009). The authors introduce BGCM as a way of combining supervised and unsupervised models for a given task. The idea behind BGCM is to cast the ensembling task as an optimization problem on a bipartite graph, where the objective function favors the smoothness of the prediction over the graph, as well as penalizing deviations from the initial labeling provided by supervised models. The algorithm entails consolidating a classification solution by maximizing the consensus among both supervised predictions and unsupervised constraints. BGCM outperforms all its component models on ten out of eleven classification tasks across three different datasets evaluated by Gao et al. (2009).

The Mixtures of Experts (ME) is another ensembling algorithm (Jacobs et al., 1991). The ME algorithm has a close proximity to our stacking with auxiliary features algorithm in terms of the underlying intuition of leveraging systems that are good at certain types of instances of a task. In this method, the problem space is partitioned stochastically into a number of sub-spaces and the idea is that the experts or learners are specialized on each subspace. ME uses divide-and-conquer principle to soft switch between learners covering different sub-spaces of the input. This method uses a supervised gating network which can be learned using Expectation-Maximization (EM). More recently, Eigen et al. (2013) extended ME to use a different gating network at each layer in a multilayer network, forming a Deep Mixture of Experts. By associating each input with a combination of experts at each layer, their model used different subsets of its units for different inputs, making it large as well as efficient.

2.3 Knowledge Base Population

Knowledge Base Population (KBP) is an NLP problem of discovering facts about entities from text and adding them to a Knowledge Base (KB) (Ji and Grishman, 2011). Information Extraction (IE) or Relation Extraction, in particular, is a sub-task for KBP. Relation extraction using a fixed ontology is called Slot-Filling (SF). Entity Discovery and Linking (EDL) is another KBP subtask that involves identifying all entity mentions in a text corpus and linking those mentions to a KB. NIST annually conducts the slot-filling (Surdeanu, 2013; Surdeanu and Ji, 2014) and entity discovery and linking (Ji et al., 2015, 2016) tasks in the KBP track of the Text Analysis Conference (TAC).

2.3.1 Slot-Filling (SF)

The goal of slot-filling is to collect information (fills) about specific attributes (slots) for a set of entities (queries) from a given corpus. The query entities in TAC KBP can be a person (PER), organization (ORG) or geo-political entity (GPE). In 2015, NIST replaced the slot-filling task with the cold start slot filling (CSSF) task. The task became more challenging because the queries used were entities that did not have a Wikipedia entry. CSSF also included the inverse of each slot, for example, the slot `org:subsidiaries` and its inverse `org:parents`. The evaluation included a total of forty-one slots and their inverses. Some slots (like *per:age*) are *single-valued* while others (like *per:children*) are *list-valued* i.e., they can take multiple slot fillers. Out of the forty-one slots, twenty-six have fills that are themselves entities and are therefore categorized as entity-valued slots, as shown in Table 2.1. Each entity-valued slot has an inverse. The remaining fifteen slots have string-valued fills, as shown in Table 2.2.

Relation	**Inverse(s)**
per:children	per:parents
per:other_family	per:other_family
per:parents	per:children
per:siblings	per:siblings
per:spouse	per:spouse
per:employee_or_member_of	{org,gpe}:employees_or_members
per:schools_attended	org:students
per:city_of_birth	gpe:births_in_city
per:stateorprovince_of_birth	gpe:births_in_stateorprovince
per:country_of_birth	gpe:births_in_country
per:cities_of_residence	gpe:residents_of_city
per:statesorprovinces_of_residence	gpe:residents_of_stateorprovince
per:countries_of_residence	gpe:residents_of_country
per:city_of_death	gpe:deaths_in_city
per:stateorprovince_of_death	gpe:deaths_in_stateorprovince
per:country_of_death	gpe:deaths_in_country
org:shareholders	{per,org,gpe}:holds_shares_in
org:founded_by	{per,org,gpe}:organizations_founded
org:top_members_employees	per:top_member_employee_of
{org,gpe}:member_of	org:members
org:members	{org,gpe}:member_of
org:parents	{org,gpe}:subsidiaries
org:subsidiaries	org:parents
org:city_of_headquarters	gpe:headquarters_in_city
org:stateorprovince_of_headquarters	gpe:headquarters_in_stateorprovince
org:country_of_headquarters	gpe:headquarters_in_country

Table 2.1: Entity-valued slots and their inverses.

PER	ORG
per:alternate_names	org:alternate_names
per:date_of_birth	org:political_religious_affiliation
per:age	org:number_of_employees_members
per:origin	org:date_founded
per:date_of_death	org:date_dissolved
per:cause_of_death	org:website
per:title	
per:religion	
per:charges	

Table 2.2: String-valued slot types for PER and ORG entities.

The input for slot-filling is a set of queries and a text corpus in which to look for information. The queries are provided in an XML format that includes an ID for the query, the name of the entity, and the type of entity (PER, ORG or GPE). The corpus consists of documents from discussion forums, newswire and the Internet, each identified by a unique ID. The output is a set of slot fills for each query. Along with each slot fill, systems must also provide *provenance* in the corpus in the form `docid:startoffset-endoffset`, where `docid` specifies a source document and the offsets demarcate the text in this document containing the extracted filler. Systems also provide a confidence score to indicate their certainty in the extracted information. Figure 2.1 illustrates an example entity, some slot fills, and the expected output format for the task.

The CSSF also included some two-hop queries. Two-hop queries have two slot types and the expected output is a fill for `slot0` in relation to the query entity and a fill for `slot1` in relation to the output of `slot0`. Below is a sample of a two-hop evaluation query from the 2015 CSSF task:

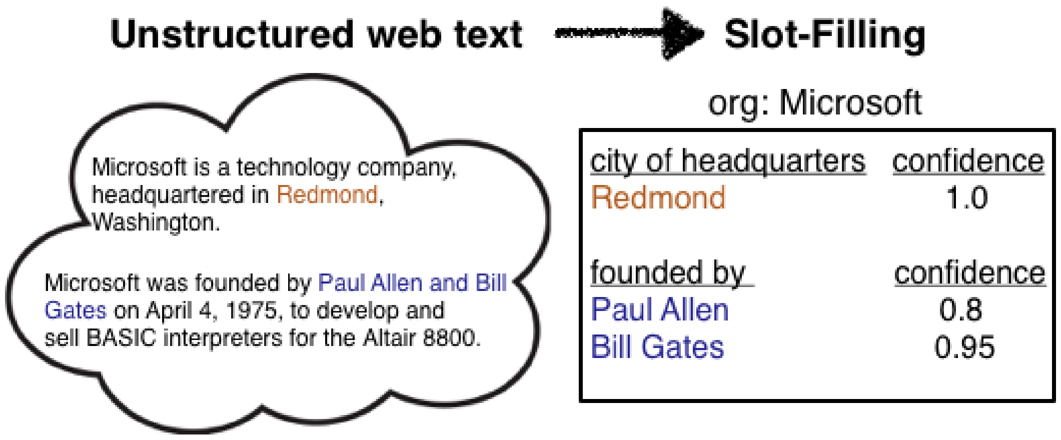

Unstructured web text
Slot-Filling
Microsoft is a technology company, headquartered in Redmond, Washington.
Microsoft was founded by Paul Allen and Bill Gates on April 4, 1975, to develop and sell BASIC interpreters for the Altair 8800.
org: Microsoft
city of headquarters
confidence
Redmond
1.0
founded by
confidence
Paul Allen
0.8
Bill Gates
0.95

Figure 2.1: Example of the slot-filling task. Systems extract fills from unstructured text for a fixed ontology and provide provenance in the form of text-substring along with a confidence score.

```
<query id="CSSF15_ENG_210">
  <name>June McCarthy</name>
  <docid>42</docid>
  <beg>16931</beg>
  <end>16943</end>
  <enttype>PER</enttype>
  <slot>per:children</slot>
  <slot0>per:children</slot0>
  <slot1>per:age</slot1>
</query>
```

The natural-language form of the query is the question “Who are June McCarthy’s children and what are their ages?”. From the above example, it is clear that the two-hop queries are much more challenging as they attempt to answer questions that are more compositional. The evaluation for such queries is also very tricky. NIST evaluated such multiple-hop queries as follows. If a system generated an incorrect `hop-0` response then all its `hop-1` responses will be treated as incorrect even if they were actually correct. So, only if the `hop-0` output of a system is correct, the `hop-1` output would be evaluated. Moreover, while calculating precision, the metric included `hop-1` outputs in the total number even though it was not even evaluated for corresponding incorrect `hop-0` outputs.

Participating teams in the slot-filling evaluation employ a variety of techniques such as distant supervision, universal schema, relevant document extraction, relation-modeling, OpenIE and inference (Finin et al., 2015; Soderland et al., 2015; Kisiel et al., 2015). The top-performing 2015 CSSF system (Angeli et al., 2015) leverages both distant supervision (Mintz et al., 2009) and pattern-based relation extraction. Another system, *UMass_IESL* (Roth et al., 2015), uses distant supervision, rule-based extractors, and semi-supervised matrix embedding methods. There has also been some work on increasing the performance of relation extraction through ensemble methods. The FAUST system for biomolecular event extraction used model combination strategies such as voting and stacking and was placed first in three of the four BioNLP tasks in 2011 (Riedel et al., 2011).

Google's Knowledge Vault system (Dong et al., 2014) combines the output of four diverse extraction methods by building a boosted decision stump classifier (Reyzin and Schapire, 2006). For each proposed fact, the classifier considers both confidence value of each extractor and number of responsive documents found by the extractor. A separate classifier is trained for each predicate, and Platt Scaling (Platt, 1999) is used to calibrate confidence scores. The use of *stacked generalization* for information extraction has been demonstrated to outperform both majority voting and weighted voting methods (Sigletos et al., 2005). In relation extraction, a stacked classifier effectively combines a supervised, closed-domain Conditional Random Field-based relation extractor with an open-domain CRF Open IE system, yielding a 10% increase in precision without harming recall (Banko et al., 2008). We are the first to apply stacking to KBP and the first to use provenance as a feature in a stacking approach. Figure 2.2 shows an overview of a generic slot-filling system's architecture.

The KBP evaluations also included the Slot Filler Validation (SFV) task[1] where the goal is to ensemble/filter outputs from multiple slot filling systems. Many KBP SFV systems apply a variety of techniques for validating output across documents such as rule-based consistency checks (Angeli et al., 2013), and techniques

[1]`http://www.nist.gov/tac/2015/KBP/SFValidation/index.html`

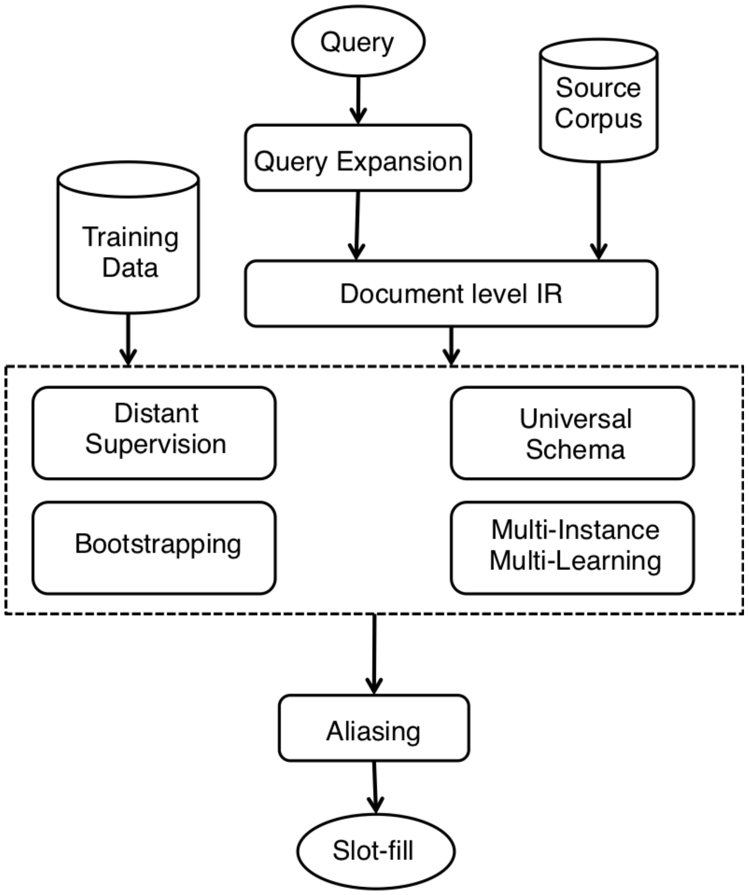


Figure 2.2: Overview of a generic slot-filling system's architecture.

from the well-known Recognizing Textual Entailment (RTE) task (Cheng et al., 2013; Sammons et al., 2014). In contrast, the 2013 *JHUAPL* system (Wang et al., 2013) aggregates the results of many different extractors using a constrained optimization framework, exploiting confidence values reported by each input system. A second approach in the *UI_CCG* system (Sammons et al., 2014) aggregates results of multiple systems by using majority voting. In the database, web-search, and data-mining communities, a line of research into "truth-finding" or "truth-discovery" methods, addresses the related problem of combining evidence for facts from multiple sources, each with a latent credibility (Yin et al., 2008). The *RPI_BLENDER* KBP system (Yu et al., 2014) casts SFV in this framework, using a graph propagation method that modeled the credibility of systems, sources, and response values.

NIST evaluated the slot-filling task using the Precision, Recall and F1 metrics. The output generated by systems is compared to the gold standard output created by the Linguistic Data Consortium (LDC) experts.[2] Precision is the fraction of correct slot-fills among the total slot-fills extracted whereas Recall is the fraction of correct slot-fills that have been extracted among the total number of slot-fills in the gold standard. F1 is the harmonic mean of precision and recall.

2.3.2 Entity Discovery and Linking (EDL)

Entity Discovery and Linking (EDL) is another sub-task for KBP. It involves two sub-problems widely known in NLP – (1) Named Entity Recognition (NER); and (2) Disambiguation. NIST annually conducts the EDL task (Ji et al., 2015) in the KBP track of the Text Analysis Conference (TAC). The source corpus provided for the task contains documents in Chinese and Spanish along with English, and thus it is called Tri-lingual Entity Discovery and Linking (TEDL).

The goal of EDL is to discover all entity mentions in a corpus of English, Spanish and Chinese documents and link those mentions to a KB. The entities can be a person (PER), organization (ORG), geo-political entity (GPE), facility (FAC), or location (LOC). The FAC and LOC entity types were newly introduced in 2015.

[2] `https://www.ldc.upenn.edu/`

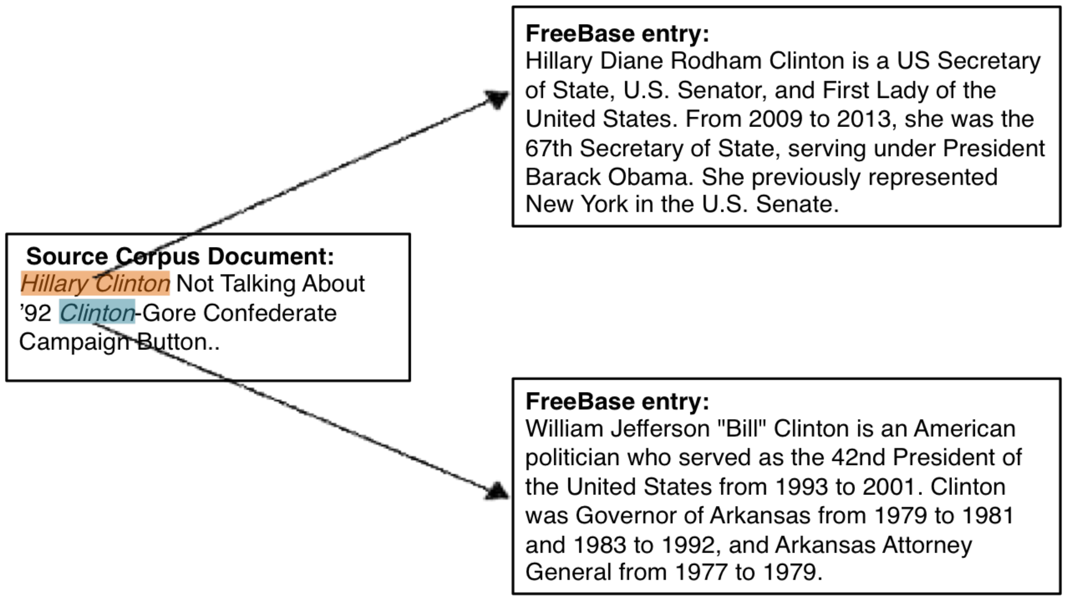

FreeBase entry:
Hillary Diane Rodham Clinton is a US Secretary of State, U.S. Senator, and First Lady of the United States. From 2009 to 2013, she was the 67th Secretary of State, serving under President Barack Obama. She previously represented New York in the U.S. Senate.
Source Corpus Document:
Hillary Clinton Not Talking About '92 Clinton-Gore Confederate Campaign Button..
FreeBase entry:
William Jefferson "Bill" Clinton is an American politician who served as the 42nd President of the United States from 1993 to 2001. Clinton was Governor of Arkansas from 1979 to 1981 and 1983 to 1992, and Arkansas Attorney General from 1977 to 1979.

Figure 2.3: The above example illustrates identifying and disambiguating entity mentions in a source document as part of the EDL task. The disambiguated mentions are then linked to their respective KB entries.

The extracted mentions are then linked to an existing English KB entity using its ID. If there is no KB entry for an entity, systems are expected to cluster all the mentions for that entity using a NIL ID. The input is a corpus of documents in the three languages and an English KB (FreeBase) of entities, each with a name, ID, type, and several relation tuples that allow systems to disambiguate entities. The output is a set of extracted mentions, each with a string, its provenance in the corpus, and a corresponding KB ID if the system could successfully link the mention, or else a mention cluster with a NIL ID. Systems can also provide a confidence score for each mention. Figure 2.3 illustrates an example of entity mention disambiguation in a document and its link to a corresponding KB entry.

Participating teams employ a variety of techniques for candidate generation and ranking. The *CMUML* team used a unified graph-based approach to do concept disambiguation and entity linking by leveraging the FreeBase ontology (Fauceglia et al., 2015). The *HITS* team combined local, unsupervised sieves with a global, supervised, joint disambiguation and NIL clustering to build a hybrid sys-

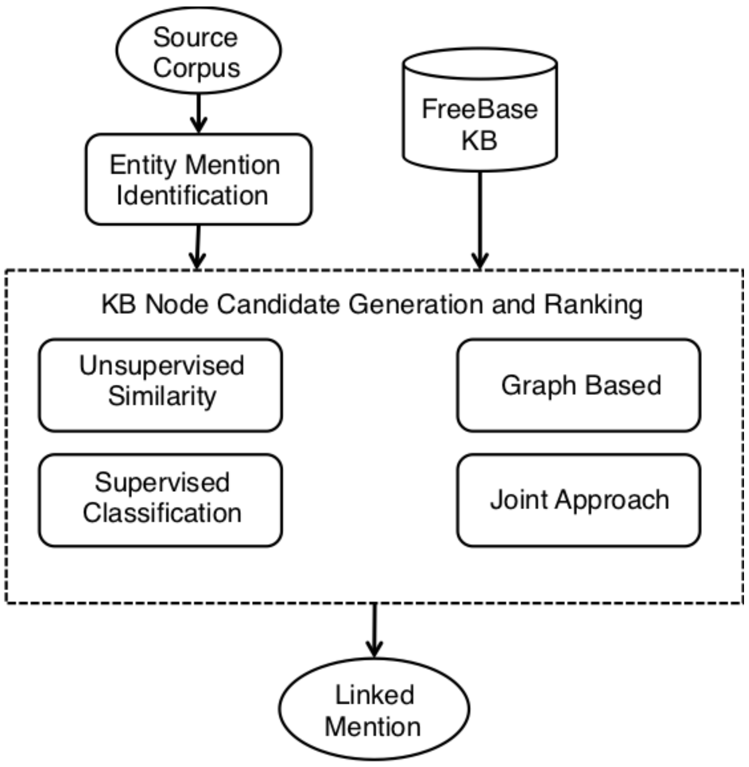

Source Corpus
FreeBase KB
Entity Mention Identification
KB Node Candidate Generation and Ranking
Unsupervised Similarity
Graph Based
Supervised Classification
Joint Approach
Linked Mention

Figure 2.4: Overview of a generic entity-linking system's architecture.

tem (Heinzerling et al., 2015). The *UI_CCG* focused on the Spanish language subtask of TEDL, by using Google Translation to translate Spanish documents into English and then using the Illinois Wikifier to identify entity mentions and disambiguate them to FreeBase entities (Sammons et al., 2015). The top-performing 2015 TEDL system used a combination of deep neural networks and Conditional Random Fields (CRFs) for mention detection and a language-independent probabilistic disambiguation model for entity linking (Sil et al., 2015). He et al. (2013) proposed a fast and scalable collective entity linking method that relies on stacking. They stack a global predictor on top of a local predictor to collect coherence information from neighboring decisions. Biomedical entity extraction using a stacked ensemble of a Support Vector Machine (SVM) and CRF was shown to outperform individual components as well as voting baselines (Ekbal and Saha, 2013). Rajani et al. (2017) use stacking for obtaining state-of-the-art results on entity linking in the medical domain on several datasets using multiple evaluation metrics. However, there has been no prior work on ensembling for the TEDL task, and our Stacking With Auxiliary Features (SWAF) approach beats the current state-of-the-art system. Figure 2.4 gives an overview of a very general entity-linking system.

NIST evaluated the EDL task using the CEAF (Luo, 2005) metric for measuring precision, recall, and F1. The reason is that entity linking is alternatively understood as a cross-document co-reference resolution task, in which the set of tuples is partitioned by the assigned entity ID. This metric finds the optimal alignment between system and gold standard clusters and then evaluates precision and recall micro-averaged over mentions.

2.4 ImageNet Object Detection

Object Detection is a well known Computer Vision problem that involves detecting instances of semantic objects of a certain category in images and videos. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a widely known annual competition in computer vision and has become the standard benchmark for large-scale object recognition (Russakovsky et al., 2015). The ImageNet dataset is organized according to the WordNet hierarchy, and thus the object categories are WordNet *synsets*. The goal of the ImageNet object detection task is to detect all instances of object categories (out of the 200 predefined categories) present in the image and localize them by providing coordinates of the axis-aligned bounding boxes for each instance. The object detection problem differs from the image classification problem that involves classifying the entire image into one of the 1000 predefined object categories. Figure 2.5 illustrates the difference between image classification and object detection.

The object detection corpus is divided into training, validation and test sets. The training set consists of approximately $450K$ images including both positive and negative instances, annotated with bounding boxes; the validation set consists of around $20K$ images also annotated for all object categories and the test set has $50K$ images. The output for the task is the image ID, the object category (1-200), a confidence score, and the coordinates of the bounding box. In case of multiple

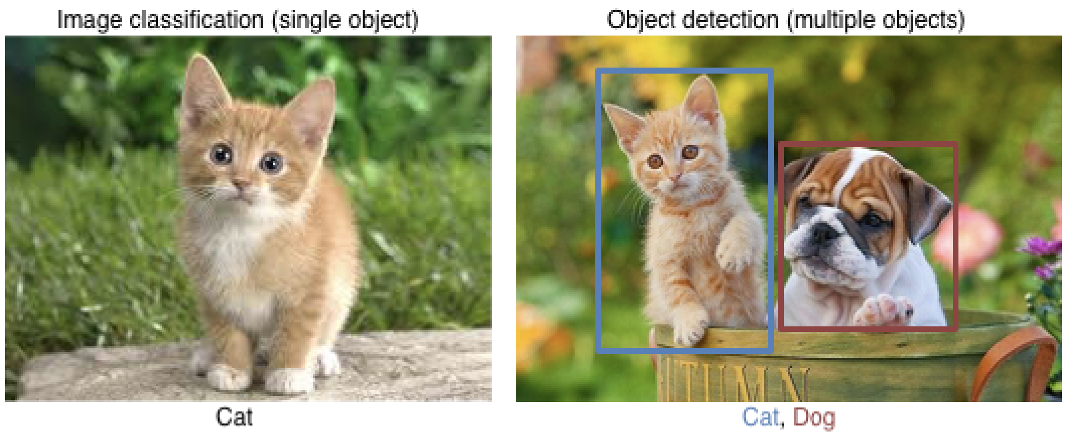


Image classification (single object)
Object detection (multiple objects)
Cat
Cat, Dog

Figure 2.5: Example comparing image classification and object detection.

instances in the same image, each instance is mentioned separately.

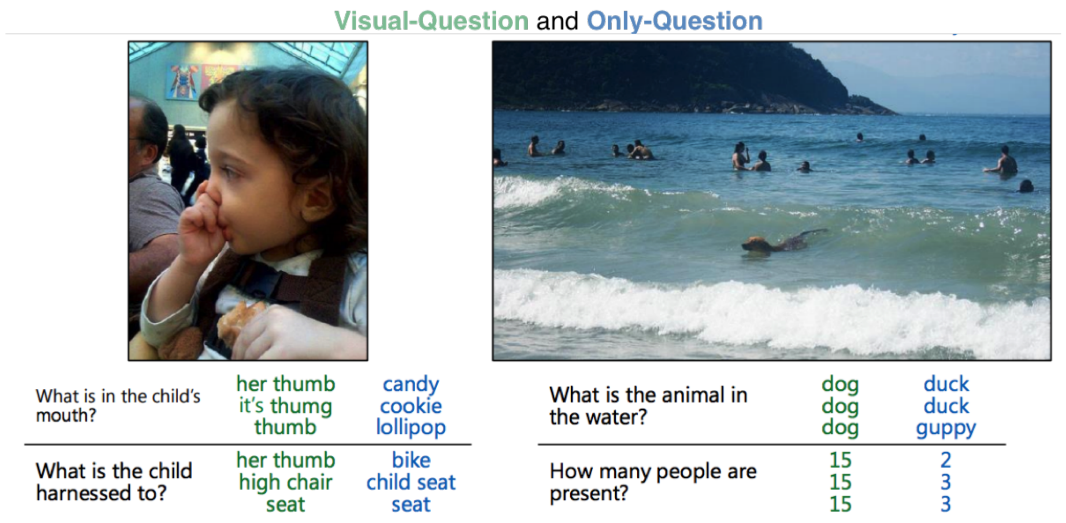
Because of the massive scale of the ImageNet dataset, almost all participating teams employ deep learning for image classification. For the object detection challenge in 2015, the top performing team used a deep residual net (He et al., 2016) and several other teams deployed a version of faster R-CNN (Region based Convolutional Neural Networks) with selective search (Ren et al., 2015). Faster R-CNN is a more efficient variant of fast R-CNN (Girshick, 2015) that first uses Region Proposal Networks (RPN) to train an end-to-end network for generating region proposals. There has been some work on stacking for multi-layer object recognition (Peppoloni et al., 2014) but our work is the first to use stacking for ensembling object *detectors*, and we obtain significant improvements over the component systems.

For the ImageNet challenge, the object detection task is evaluated using the average precision (AP) on a precision-recall curve. The predicted bounding box for a detection is considered correct if its intersection over union with the ground truth exceeds a threshold of 0.5 (Russakovsky et al., 2015). The official scorer gives the average precision for each of the 200 classes along with the overall median and mean AP.

2.5 Visual Question Answering (VQA)

Visual Question Answering (VQA) is the task of addressing open-ended questions about images. It has attracted significant attention in recent years (Andreas et al., 2016a; Goyal et al., 2016; Agrawal et al., 2016; Teney et al., 2017). The DAtaset for QUestion Answering on Real-world images (DAQUAR) was the first dataset and benchmark for this task (Malinowski and Fritz, 2014). Recently, many synthetic datasets such as the CLEVR (Johnson et al., 2017) and NLVR (Suhr et al., 2017) have also been proposed for the visual question answering problem. The CLEVR dataset consists of questions that require elementary reasoning on synthetic images while the NLVR dataset consists of natural language sentences describing the image and systems must output whether the sentence correctly describes the image or not. The Visual Question Answering (VQA) dataset (Antol et al., 2015) is the most well-known and widely used dataset for the VQA task. The dataset consists of real images as well as abstract scenes. Given an image and a natural language question about that image, the task is to provide an accurate natural language answer. VQA requires visual and linguistic comprehension, language grounding capabilities as well as common-sense knowledge. Figure 2.6 shows examples of answers given by humans when they were shown just the question versus both the image and the question. It is clear that the VQA task is multi-modal and therefore challenging.

A variety of methods to address the challenges of VQA have been developed in recent years (Fukui et al., 2016; Xu and Saenko, 2016; Lu et al., 2016; Chen et al., 2015). The vision component of a typical VQA system extracts visual features using a deep convolutional neural network (CNN), and the linguistic component encodes the question into a semantic vector using a recurrent neural network (RNN). An answer is then generated conditioned on the visual features and the question vector. Figure 2.7 demonstrates a generic neural network on a VQA example. Several deep learning models have been developed that combine a computer vision component with a linguistic component in order to solve the VQA challenge. Some of these models also use data-augmentation for pre-training. The iBowIMG (Zhou et

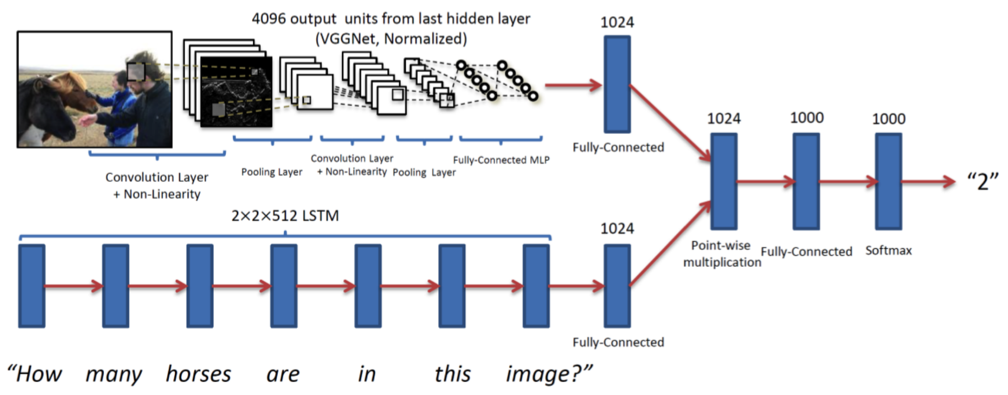


What is in the child's mouth?	her thumb it's thumg thumb	candy cookie lollipop
What is the child harnessed to?	her thumb high chair seat	bike child seat seat

What is the animal in the water?	dog dog dog	duck duck guppy
How many people are present?	15 15 15	2 3 3

Figure 2.6: Examples of VQA questions (black) and answers given by humans when looking at the image (green) and when not looking at the image (blue).

al., 2015b), the DPPNet (Noh et al., 2016), the Neural Module Networks (NMNs) (Andreas et al., 2016b), the LSTM (Antol et al., 2015), the HieCoAtt (Lu et al., 2016) and the MCB (Fukui et al., 2016) are some of the deep learning models that attempt to solve VQA. The iBowIMG uses image features with the bag-of-word question embedding in a softmax classifier resulting in performance comparable to other models that use deep or recursive neural networks. The DPPNet, on the other hand, learns a CNN with some parameters predicted from a separate parameter prediction network that uses a Gated Recurrent Unit (GRU) to generate a question representation and maps the predicted weights to a CNN via hashing. The DPPNet uses external data in addition to the VQA dataset to pre-train the GRU. The GRU is initialized with the skip-thought (Kiros et al., 2015) vector model trained on a book-collection corpus containing more than $74M$ sentences. Another well-known VQA model is the Neural Module Network (NMN) (Andreas et al., 2016b) that generates a neural network on the fly for every individual image and question. This is done using various sub-modules based on the question and composing these to generate the neural network, *e.g.*, the `find[x]` module outputs an attention map for detecting `x`. The question is first parsed into a symbolic expression and using



4096 output units from last hidden layer
(VGGNet, Normalized)
1024
Convolution Layer + Non-Linearity
Pooling Layer
Convolution Layer + Non-Linearity
Pooling Layer
Fully-Connected MLP
Fully-Connected
1024
1000
1000
"2"
2×2×512 LSTM
1024
Point-wise multiplication
Fully-Connected
Softmax
Fully-Connected
"How many horses are in this image?"

Figure 2.7: A CNN-RNN combination on a VQA example (Lu et al., 2016).

these expressions, modules are composed into a sequence to answer the query. The whole system is trained end-to-end through backpropagation.

The LSTM model uses the VGGNet as a CNN and LSTM as an RNN. The HieCoAtt system is similar to the LSTM except it uses co-attention at the level of both image and question. The MCB model uses a 152-layer ResNet as a CNN and LSTM as an RNN. It combines the image and question vector representations using an outer product. A non-deep learning approach to VQA uses a Bayesian framework to predict the form of the answer from the question (Kafle and Kanan, 2016).

The VQA challenge has two modalities for answering the questions: (i) open-ended and (ii) multiple choice. For the open-ended task, the ground-truth answer is collected from 10 different human subjects. The system generated answers are evaluated using the following accuracy metric:

$$\text{Accuracy} = \min\left(\frac{\text{\# humans that provided that answer}}{3}, 1\right) \tag{2.1}$$

That is, an answer is considered 100% accurate if at least 3 human subjects provided that same answer. The multiple-choice questions have 18 candidate answers for each question. These questions are also evaluated using the evaluation metric in Equation 2.1.

2.6 Explainable AI (XAI)

Recently, with the development of large-scale datasets and powerful computers, there has been a dramatic success in several machine learning applications especially in the domains of computer vision and natural language processing. These advancements have resulted in the production of systems that perceive, learn, decide and act autonomously. However, these systems are limited in their effectiveness because they do not produce any explanation for their decisions and actions. The state-of-the-art systems in many AI applications use ensembles of deep neural networks that are even more difficult to interpret. Explanation provides a means for AI systems to convey their strengths and weakness and enables them to provide a rationale for their decision, especially, when they fail in a spectacular manner. Traditionally, transparency has always been at odds with performance. Systems that are simple are more transparent but have poor performance compared to complex systems with a large number of parameters whose decisions are much more difficult to explain.

Beyond academia, the European Union's General Data Protection Regulation (GDPR) now includes the "right to explanation" which states that a user can ask for an explanation of an algorithmic decision that was made about them (Goodman and Flaxman, 2017). Figure 2.8 shows the concept underlying XAI as described in Gunning (2016).

In this thesis, our goal for generating explanations is not just to build trust between humans and AI systems, but to use explanations in turn for improving the performance of AI applications. Generating explanations for AI systems is, without a doubt, a challenging problem but research on explainable AI systems is incomplete without a good evaluation metric to measure their effectiveness and usefulness. Research on XAI is therefore divided into two main categories – (i) generating explanations and (ii) evaluating explanations. We review the background and related work in each of these categories below.

Generating Explanations: Ensembles of deep learning models have been used widely on several real-world vision and language problems. Despite their success,

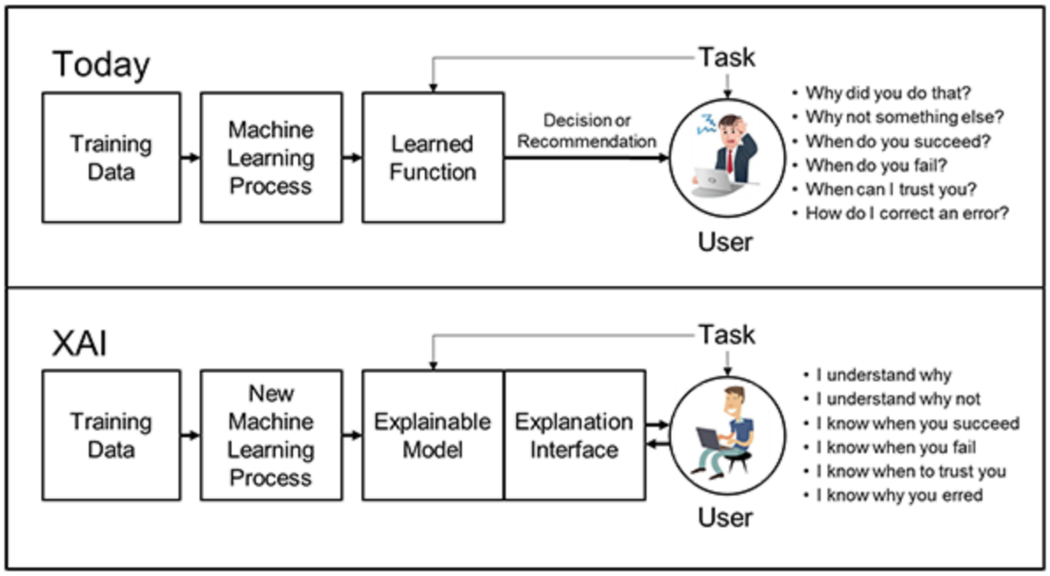


Figure 2.8: XAI concept as described in Gunning (2016).

ensembles lack transparency and are unable to explain their decisions. On the other hand, humans can justify their decisions in natural language as well as point to the visual evidence that supports their decision. AI systems that can generate explanations supporting their predictions have several advantages (Johns et al., 2015; Agrawal et al., 2016). When an AI system is weaker than humans, the goal of explanation is to identify failure modes (Agrawal et al., 2016). For AI systems that are on par with human performance, explanation serves to establish users' trust. Finally, when an AI system is significantly better than humans, explanation can be used to teach humans to make better decisions (Johns et al., 2015). This has motivated recent work on explainable AI systems, particularly in computer vision (Antol et al., 2015; Goyal et al., 2016; Park et al., 2016). Early work on explainable models used a template-based approach (Lane et al., 2005). Recent work uses more sophisticated methods for generating explanations. Hendricks et al. (2016) developed a deep network to generate natural language justifications for a fine-grained object classifier. Their model focuses on the discriminating properties of the object in the image, jointly predicts a class label and also explains why the predicted label

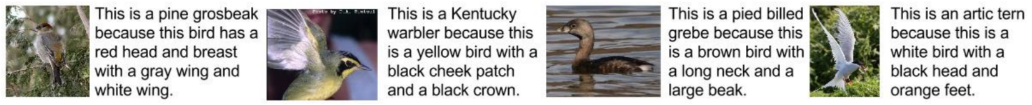

This is a pine grosbeak because this bird has a red head and breast with a gray wing and white wing.
This is a Kentucky warbler because this is a yellow bird with a black cheek patch and a black crown.
This is a pied billed grebe because this is a brown bird with a long neck and a large beak.
This is an artic tern because this is a white bird with a black head and orange feet.

Figure 2.9: Sample explanations produced by Hendricks et al. (2016) on the fine-grained bird species classification dataset.

is appropriate for the image. Figure 2.9 shows a sample of explanations produced by Hendricks et al. (2016) on a fine-grained bird species classification dataset (Wah et al., 2011).

Explanations can also be model-agnostic as shown by LIME (Local Interpretable Model-agnostic Explanations) which trains and presents local sparse models of how predictions change when inputs are perturbed (Ribeiro et al., 2016). A method for explaining differentiable models by selectively penalizing their input gradients has been demonstrated on toy and real-world datasets (Ross et al., 2017). Ramanishka et al. (2017) created a model for inferring top-down attention from captions. A body of work has proposed methods to visually explain decisions. Berg and Belhumeur (2013) use discriminative visual patches, whereas Zhou et al. (2015a) aim to understand intermediate features which are important for end decisions by naming hidden neurons that detect specific concepts. Selvaraju et al. (2017) used a gradient-based localization approach called Grad-CAM for generating visual explanation. Figure 2.10 shows a sample of visual explanations generated on some image-question pairs using the Hierarchical Co-attention (Lu et al., 2016) VQA model. However, there has been no prior work on generating explanations for *ensembles* of multiple AI systems. In this thesis, we propose algorithms for ensembling visual explanations of deep learning models that can be used to explain the decision of the ensemble. We demonstrate the success of our approach on the challenging task of Visual Question Answering (VQA).

Evaluating Explanations: Evaluating explanations generated by AI systems is a challenging problem and has attracted some attention in recent years. Although crowd-sourced human evaluation has been typically used to evaluate explanations, the actual metrics and approaches have differed widely across tasks and domains.

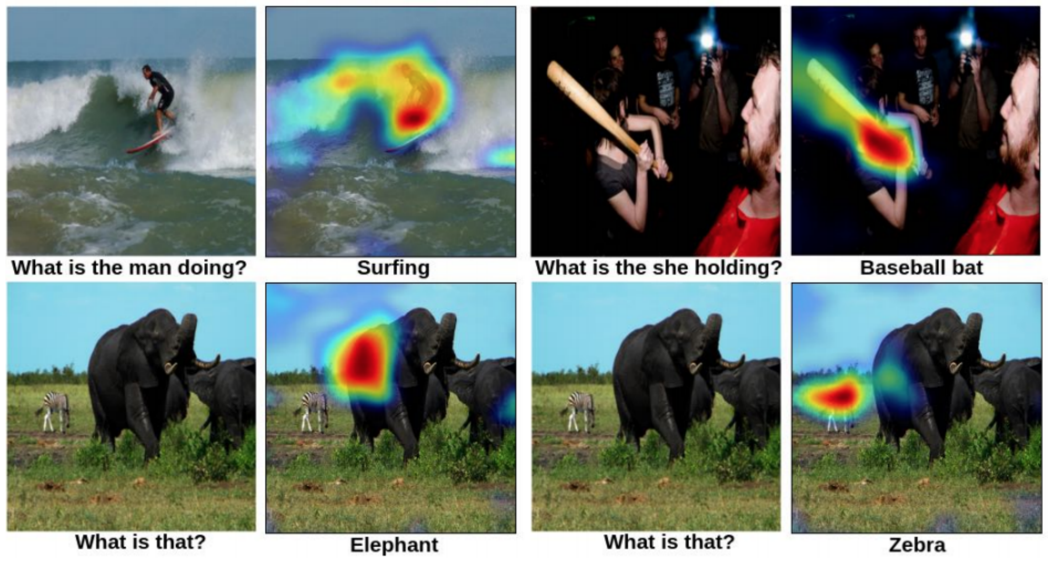


Figure 2.10: Visual explanations on sample VQA image-question pairs from Selvaraju et al. (2017)

Hendricks et al. (2016) used human experts on bird watching to evaluate explanations for fine-grained bird classification and asked them to rank the image-explanation pairs. On the other hand, Das et al. (2016) collect human attention maps for VQA by instructing human subjects on Mechanical Turk (MTurk) to sharpen parts of a blurred image that are important for answering the questions accurately. Typical explanation evaluation metrics rely on annotated ground truth explanations (Park et al., 2016; Goyal et al., 2016; Das et al., 2017a). Selvaraju et al. (2017) evaluated explanations for image captioning by instructing human subjects on MTurk to select if a machine generated explanation is reasonable or not based on the predicted output. In this thesis, we propose two new evaluation approaches that are not dependent on ground truth explanations. Our work evaluates explanations for VQA that does not rely on human-generated explanation. This is important because research shows that machines and humans do not have the same "view" of visual explanations (Das et al., 2017a).

2.7 Chapter Summary

In this chapter, we reviewed the background and related work relevant to this thesis. We discussed a variety of ensembling techniques in machine learning. We also discussed in detail four different AI applications. Two were in the domain of natural language processing – Slot-filling and Entity Linking. The Object detection task is in the domain of computer vision, while the Visual Question Answering task requires both language and vision understanding.

The input to all four tasks is a massive corpus of text or images, and all the tasks require structured outputs. We also demonstrated through examples the various challenges posed by these tasks. As discussed in this chapter, all the four problems are active areas of research and have plenty of exciting new work.

Chapter 3
Stacking with Auxiliary Features

This chapter introduces a new ensembling algorithm called Stacking With Auxiliary Features (SWAF) that can be applied to classification problems and problems with structured outputs. It also discusses two types of auxiliary features – the *provenance features* and the *instance features*. Parts of the work in this chapter have been published in (Rajani and Mooney, 2017b). All work in this chapter constitutes original contributions.

3.1 Chapter Overview

Ensembling multiple systems is a well known standard approach to improving accuracy in machine learning (Dietterich, 2000) and ensembles have been applied to a wide variety of problems in all domains of artificial intelligence including natural language understanding. However, these techniques do not learn to adequately discriminate across the component systems and thus are unable to integrate them optimally. Therefore, combining systems intelligently is crucial for improving overall performance.

In this chapter, we consider the general machine learning problem of combining structured outputs from multiple systems to improve accuracy using an algorithm we proposed called stacking with auxiliary features (SWAF). We used two types of auxiliary features - those that enable the stacker to discriminate across component systems, which we called the *provenance features*, and those that enable the stacker to discriminate across instances, which we called the *instance features*. We also propose a variation of SWAF that can ensemble systems that do not have training data, in an unsupervised manner.

3.2 Prior Work

Using ensembles of multiple systems is a standard approach to improving accuracy in machine learning (Dietterich, 2000). Ensembles have been applied to a wide variety of problems in natural language processing, including parsing (Henderson and Brill, 1999), word sense disambiguation (Pedersen, 2000), and sentiment analysis (Whitehead and Yaeger, 2010).

Meta-learning addresses the question of how can we improve the performance of learning algorithms by using metadata about learning (Vilalta and Drissi, 2002). Stacking is a type of meta-learning in which a meta-classifier is learned to combine the outputs of multiple underlying systems (Wolpert, 1992). The stacker learns a classification boundary based on the confidence scores provided by individual systems for each possible output. However, many times the scores produced by systems are not probabilities or not well calibrated and cannot be meaningfully compared. In such circumstances, it is beneficial to also have other reliable auxiliary features, as in our SWAF approach.

3.3 Stacking With Auxiliary Features (SWAF)

Stacking is a popular ensembling methodology in machine learning and has been very successful in many applications including the top performing systems in the Netflix competition (Sill et al., 2009). The idea is to employ multiple learners and combine their predictions by training a "meta-classifier" to weight and combine multiple models using their confidence scores as features. By training on a set of supervised data that is disjoint from that used to train the individual models, it learns how to combine their results into an improved ensemble model. As for the meta-classifier, we employ a wide range of approaches including a $L1$-regularized SVM with a linear kernel (Fan et al., 2008), SVM with an RBF kernel as well as a neural network. Our choice of the meta-classifier depends on the number and type of auxiliary features which in turn depend on the application at hand.

In a final *post-processing* step, the classifications considered "correct" by the meta-classifier are kept while the others are discarded. We use stacking as our ensembling algorithm and extend it to make it even more powerful by using task-specific features which we call as *auxiliary features*.

Stacking with Auxiliary Features (SWAF) (Rajani and Mooney, 2017b) is a type of ensembling algorithm which learns to combine outputs of multiple systems using features of the component models and current problem as context. The intuition behind SWAF is that certain systems are better at certain input types and if this can be learned then the classifier can make better decisions on the input instances. We use two types of auxiliary features:

(i) *Provenance features* for discriminating between component models. The idea behind using the provenance features is that an output is reliable if not just multiple systems agree on it but they also agree on the provenance of the output. The exact form of provenance is specific to the task (e.g., a region in a text or an image) and captures "where" the system got its answer. Provenance indicates the origin of the generated output for each system and can be used to measure the reliability of system outputs.

(ii) *Instance features* for discriminating between instances. The idea behind using the instance features is that certain systems are better at certain instance types.

Stacking with auxiliary features can be successfully applied to any problem whose output instances have confidence scores and optionally provenance that justifies the output. Figure 3.1 gives a generalized overview of our approach to combining multiple system outputs. The SWAF algorithm requires identifying appropriate instance and provenance features for a given task. We discuss these auxiliary features in more detail in Chapter 4 and Chapter 5 where we demonstrate SWAF on well known natural language and vision problems respectively.

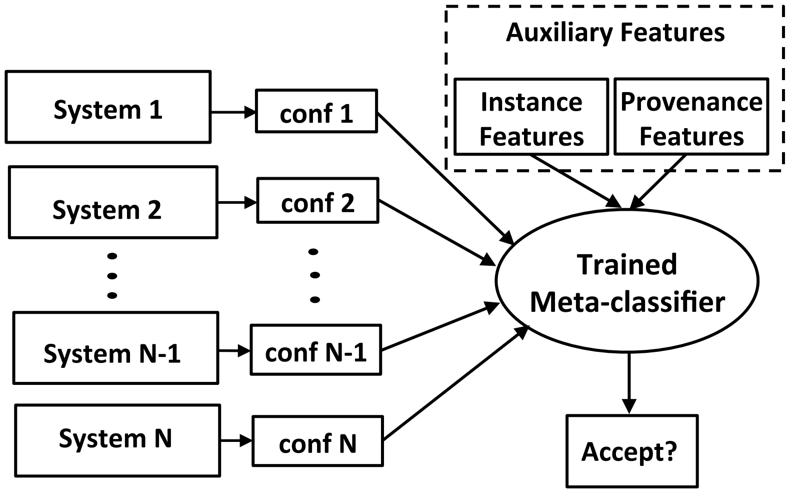


Figure 3.1: Our SWAF approach to combining system outputs using confidence scores and two types of auxiliary features for improving prediction.

3.4 Combining Supervised and Unsupervised Ensembles using SWAF

In our approach to using SWAF, in order to train the stacker, we are only able to use systems that are either open source or whose training data is available to us. However, in some situations, we would like to ensemble systems for which we have no performance data, i.e., the output of a system applied to a particular dataset so that we can do experiments without having to run the system. For example, due to privacy, some companies or agencies may be willing to share their final model's output or meta-level model output but not the raw data itself due to privacy protection. Also, it is not always possible to have access to the actual models themselves.

Simple methods such as voting permit "unsupervised" ensembling but fail to exploit supervision for those systems for which we *do* have training data. Therefore, we presented an approach that utilizes supervised *and* unsupervised ensembling to exploit the advantages of both. We first use unsupervised ensembling such as con-

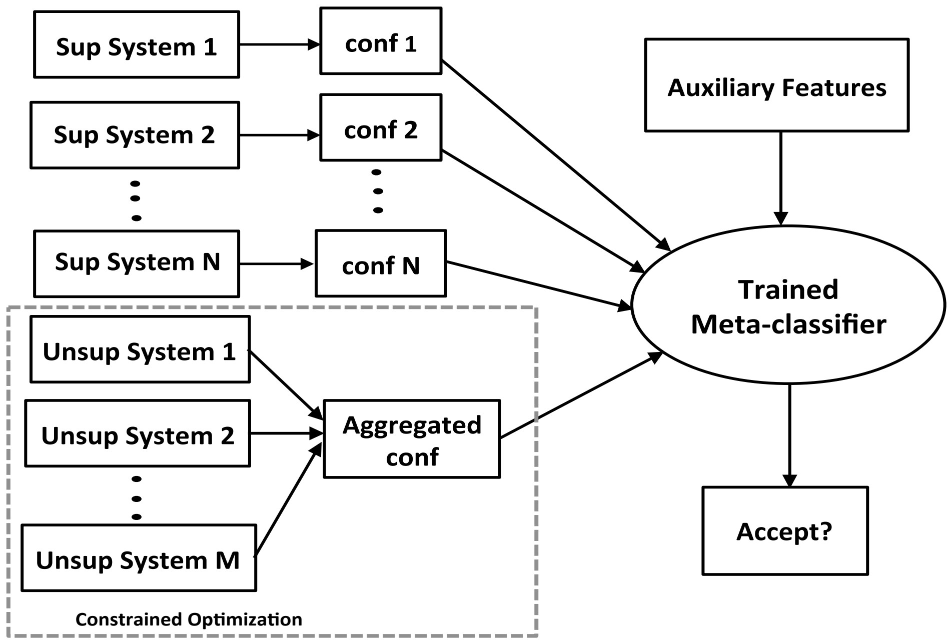


Figure 3.2: Given N systems with training data and M systems without training data, our variant of SWAF approach combines the unsupervised ensemble of M systems with N supervised systems.

strained optimization (Wang et al., 2013) to combine systems without training data, and then used stacking to combine this ensembled system with other systems for which training data is available. Figure 3.2 illustrates our system which trains a final meta-classifier for combining multiple systems using confidence scores and other auxiliary features depending on the task. The top half of the figure illustrates ensembling multiple systems with historical training data using the supervised SWAF approach while the bottom half of the figure shows the ensembling of the systems without historical training data. We demonstrate this variant of SWAF on two natural language understanding problems in Chapter 4.

3.5 Chapter Summary

In this chapter, we introduced stacking with auxiliary features (SWAF), a novel approach to ensemble multiple diverse system outputs. The auxiliary features enable the system to learn to appropriately use provenance and instance information to aid the optimal integration of multiple systems.

Chapter 4
Stacking with Auxiliary Features for Natural Language Processing

This chapter demonstrates the application of stacking with auxiliary features to two well-known natural language processing tasks – relation extraction and entity linking. Parts of the work in this chapter have been published in (Rajani et al., 2015), (Rajani and Mooney, 2016) and (Rajani and Mooney, 2017b). All work in this chapter constitutes original contributions.

4.1 Chapter Overview

As discussed in Chapter 3, Stacking with Auxiliary Features (SWAF) is a type of ensembling algorithm which learns to combine outputs of multiple systems using features of the component models and current problem as context. The auxiliary features aid the stacker to effectively integrate multiple individual models by appropriately using instance and provenance information.

In this chapter, we demonstrate our SWAF approach to multilingual information extraction as part of the knowledge base population (KBP) problem. In particular, we used the SWAF approach in the (i) Slot Filling (SF) (also known as relation extraction) and (ii) Entity Discovery and Linking (EDL) tasks conducted by NIST annually as part of the Text Analysis Conference (TAC).[1]

We also demonstrate a variant of SWAF, that was introduced in Chapter 3, that can ensemble systems which do not have training data in an unsupervised manner. We obtain state-of-the-art results on both the KBP tasks using SWAF and beat our own previous performance on both these tasks by also adding the unsupervised ensemble.

[1] https://tac.nist.gov/

4.2 Prior Work

As discussed in Section 2.2, stacking is a well-known ensembling approach that combines outputs of multiple diverse systems using their confidence scores (Wolpert, 1992). The use of stacked generalization for information extraction has been demonstrated to outperform both majority voting and weighted voting methods (Sigletos et al., 2005). In relation extraction, a stacked classifier effectively combines a supervised, closed-domain conditional random field (CRF) based relation extractor with an open-domain CRF Open IE system (Banko et al., 2008). We were the first to introduce novel auxiliary features to the stacking approach and demonstrate stacking with auxiliary features on KBP.

We would sometimes like to ensemble systems for which we have no historical performance data. Simple methods such as voting permit "unsupervised" ensembling, and several more sophisticated methods have also been developed for this scenario (Wang et al., 2013). Our variant of SWAF is able to exploit supervision for systems that *do* have training data. It utilizes both supervised *and* unsupervised ensembling to exploit the advantages of both. We first use unsupervised ensembling to combine systems without training data, and then use stacking with auxiliary features to combine this ensembled system with other systems with available training data.

4.3 Stacking with Auxiliary Features for Relation Extraction

Many end-to-end relation extraction systems are built using several underlying models that target specific sub-problems and are combined using ensembling. Given a set of query entities and a fixed set of slots, the goal of ensembling is to effectively combine the output of different slot-filling systems. As input, an ensemble takes the output of individual systems containing slot fillers and additional information such as provenance and confidence scores. The output of the ensembling system is similar to the output of an individual system, but it productively aggregates the slot fillers from different systems. For example, the bootstrapped self-training model by Angeli et al. (2015) uses pattern-based methods for improving precision as well as distant supervision for improving recall in relation extraction. These models are then combined using voting to obtain the final output. However, voting is not very effective as it ignores the subtleties of the underlying models. We, therefore, proposed a more intelligent approach to ensembling component models that uses stacking as a way of ensembling information extractors.

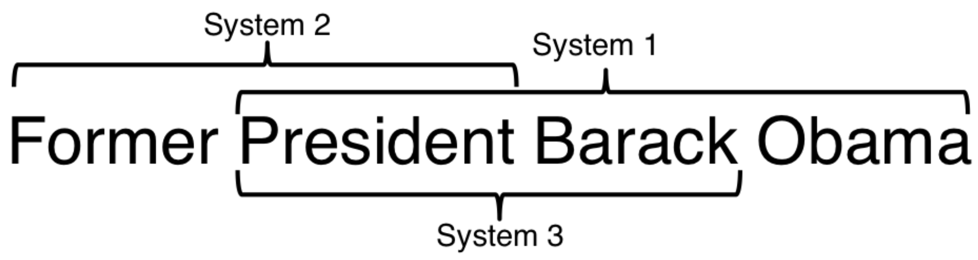
As discussed in Chapter 3, we use two types of auxiliary features – (i) provenance features and (ii) instance features. These features are task-dependent and we now discuss each of these in detail in the context of relation extraction.

4.3.1 Provenance Features

Each system that extracts a slot-fill for a query also provides provenance information about the fill. The filler provenance localizes the extracted slot-fill in a corpus of documents. Every provenance has a *docid* and *startoffset-endoffset* that gives information about the document and offset in the document from where the slot fill has been extracted. Where a slot-fill was extracted from can tell a lot about whether it is potentially correct or incorrect. This motivated us to use the provenance information as a feature to the stacker.

We computed two types of provenance scores, first using the *docid* information, and second, using the *offset* information. The document-based provenance score is defined as follows. For a given query and slot, if N systems provide answers and a maximum of n of those systems give the same *docid* in their filler provenance, then the document provenance score for those n slot fills is n/N. Similarly, other slot fills are given lower scores based on the fraction of systems whose provenance document agrees with theirs. Since this provenance score is weighted by the number of systems that refer to the same provenance, it measures the reliability of a slot fill based on the document from where it was extracted.

Our second provenance measure uses *offsets*. The degree of overlap among the various systems' offsets can also be a good indicator of the reliability of the slot fill. The Jaccard similarity coefficient is a statistical measure of similarity between sets. We extend the idea of Jaccard similarity to measure the degree of overlap



Offsets	System 1	System 2	System 3
Start offset	7	0	7
End Offset	28	15	22

$$OP_1 = \frac{1}{2}\left(\frac{9}{29} + \frac{16}{22}\right) \quad OP_2 = \frac{1}{2}\left(\frac{9}{29} + \frac{9}{23}\right) \quad OP_3 = \frac{1}{2}\left(\frac{16}{22} + \frac{9}{23}\right)$$

Figure 4.1: Suppose three systems extract fills from the string "Former President Barack Obama" at offsets shown above, starting at offset zero. The offset provenance feature for each system is calculated using Jaccard similarity measure applied on substrings as demonstrated in this image.

between substrings extracted by systems in a document. Slot fills have variable lengths and thus the provenance offset ranges are variable too. A metric such as the Jaccard coefficient captures the overlapping offsets along with normalizing based on the union and thus resolving the problem with variable offset ranges. For a given query and slot, if N is the size of the set of systems that answer with the same *docid* for their document provenance and *substring(i)* is a fill in the form of a substring extracted by a system i, then the offset provenance (OP) score for a slot fill by a system x is calculated as follows:

$$\text{OP}_n = \frac{1}{|\text{N}| - 1} \sum_{i \in \text{N}, i \neq \text{n}} \frac{|\text{substring(i)} \cap \text{substring(n)}|}{|\text{substring(i)} \cup \text{substring(n)}|} \tag{4.1}$$

Figure 4.1 shows a cartoon example of calculating offset provenance score for three systems. As per our definition, systems that extract slot fills from *different* documents for the same query slot have zero overlap among offsets. We note that the offset provenance is always used along with the document provenance and is thus useful in discriminating between slot fills extracted from the same document for the same query slot. Like the document provenance score, the offset provenance score is also a weighted feature and is a measure of the reliability of a slot fill based on the offsets in the document from where it is extracted. Our approach does not need access to the large corpus of documents from where the slot fills are extracted and is thus very computationally inexpensive.

4.3.2 Instance Features

The idea behind using the instance auxiliary features is that some systems are better at some sub-tasks. One could imagine that some systems are good at extracting relations for certain slot types (e.g.: slot types that expect a location as an output). Such information if available at the time of classification could be a deal breaker. For example, the stacker could learn not to trust an output for a slot type 'per:date_of_birth' from a system that is not good at extracting dates or numbers in general. The instance features enable the stacker to learn such patterns using task-specific information. For the slot filling task, we use the one-hot encoding of the slot type (e.g., per:age) as instance features.

4.3.3 Post-processing

We perform final post-processing on the classified outputs obtained from the stacker. This step ensures that the final output appears as if it was generated by an individual system and there is no conflicting information. Each list-valued slot fill that is classified as correct is included in the final output. For single-valued slots, if multiple fills are classified as correct for the same query and slot type, we only include the fill with the highest meta-classifier confidence.

4.3.4 Eliminating Slot-Filler Aliases

When combining the output of different SF systems, it is possible that some slot-filler entities might overlap with each other. A system *A* could extract a filler F_1 for a slot *S* while another system *B* extracts another filler F_2 for the same slot *S*. If the extracted fillers F_1 and F_2 are *aliases* (i.e. different names for the same entity) and not combined into one, the precision of the final system is penalized for producing duplicate outputs.

In order to eliminate aliases from the output of the ensembled system, we employed a technique derived by inverting the scheme used by Roth et al. (2013) for query expansion. The authors used a Wikipedia anchor-text model (Roth and Klakow, 2010) to generate aliases for given query entities. By including aliases for query names, the systems increase the number of candidate sentences fetched for the query. To eliminate filler aliases, we applied the same technique used to generate aliases for all slot fillers of a given query and slot type. Given a slot filler, we obtained the Wikipedia page that is most likely linked to the filler text based on query string matching. Then, we obtained the anchor texts and their respective counts from all other Wikipedia pages that link to this page. Using these counts, we choose top 10 anchor texts as aliases for the given slot filler. Using the generated aliases, we then verified if any of the slot fillers are redundant with respect to these aliases. This scheme is not applicable to slot types whose fillers are not entities (like date or age). Therefore, simpler string-based matching schemes are used to eliminate redundancies for these slot types.

4.3.5 Experimental Evaluation on 2014 KBP Slot-Filling

This section describes a comprehensive set of experiments evaluating the supervised ensembling approach on the 2014 KBP English Slot Filling (ESF) task. Our experiments are divided into two subsets based on the datasets they employed. Since our stacking approach relies on historical training data, we used the 2013 Slot Filler Validation (SFV) data for training and build a dataset of one run for every team that participated in *both* the 2013 and 2014 competitions and called it

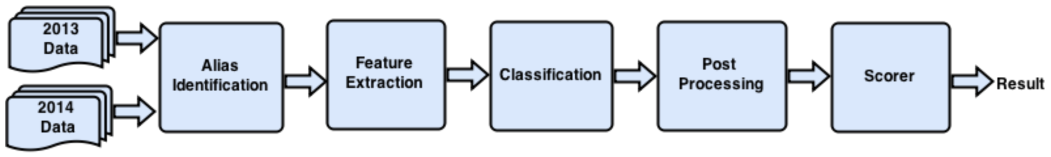
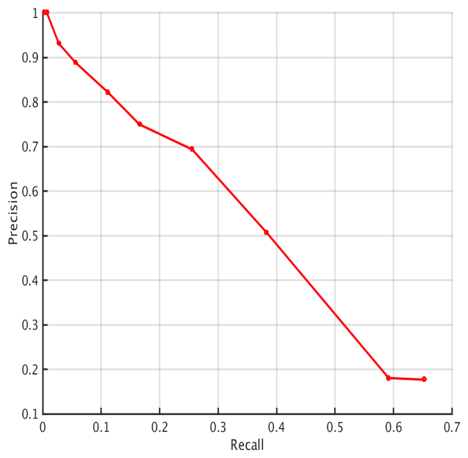


Figure 4.2: Our system pipeline for end-to-end development of the SWAF ensemble for the KBP SF task.

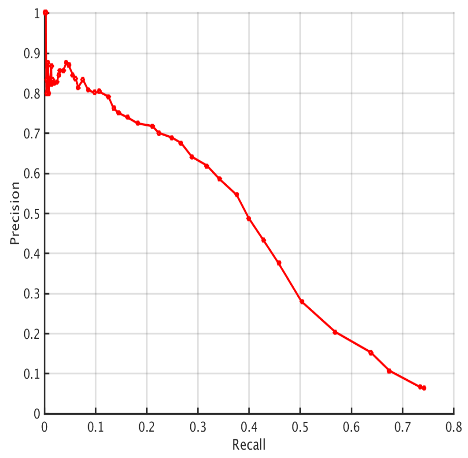
the *common systems dataset*. As described in Section 2.3.1, the SFV task involved ensembling/filtering outputs from multiple slot filling systems. There were 10 common teams of the 17 teams that participated in ESF 2014. The other dataset was composed of all 2014 SFV systems (including all runs of all 17 teams that participated in 2014). There were 10 systems in the common systems dataset, while there were 65 systems in the all 2014 SFV dataset. Figure 4.2 shows the pipeline for our SWAF approach on the 2014 KBP SF task.

We compared our stacking approach to voting and union ensembling baselines. Since both voting and union are unsupervised ensembling baselines, we evaluated on both the common systems dataset as well as the entire 2014 SFV dataset. The *Union* takes the combination of values for all systems to maximize recall. If the slot type is list-valued, it classifies all slot fillers as correct and always includes them. If the slot type is single-valued, if only one system attempts to answer it, then it includes that system's slot fill else if multiple systems attempt, then it only includes the slot-fill with the highest confidence value as correct and discards the rest.

For the *Voting* approach, we vary the threshold on the number of systems that must agree on a slot-fill from one to all. This gradually changes the system from the union to intersection of the slot fills, and we identify the threshold that results in the highest F1 score. We learned a threshold on the 2013 SFV dataset (containing 52 systems) that results in the best F1 score, thereafter, we used this threshold for the voting baseline on 2014 SFV dataset. The third baseline we compared to is an "oracle threshold" version of *Voting*. We did the same thing we did for 2013 dataset for the common systems dataset. However, since the best threshold for 2013 may not necessarily be the best threshold for 2014, we identified the best threshold for



Common systems dataset



All 2014 SFV systems dataset

Figure 4.3: Precision-Recall curves for identifying the best voting performance on the two datasets

2014 by plotting a Precision-Recall curve and finding the best F1 score for the voting baseline. Figure 4.3 shows the Precision-Recall curve for the two datasets for finding the best possible F1 score using the voting baseline. We found that for the common systems dataset, a threshold of 3 (of 10) systems gave the best F1 score, while for the all 2014 SFV dataset, a threshold of 10 (of 65) systems gave the highest F1. We note that this gave an upper-bound on the best results that can be achieved with voting, assuming an optimal threshold is chosen. Since the upper-bound could not be predicted without using the 2014 dataset, this baseline has an unfair advantage. Table 4.1 shows the performance of all 3 baselines on the all 2014 SFV systems dataset. The reason we evaluated the ensemble baselines on the all 2014 SFV systems dataset is to highlight that our SWAF approach beats all these ensemble baselines even when it uses only a fraction of the data, that is the common systems dataset, as discussed below.

We performed various ablations of the stacking algorithm by training on the 2013 data and testing on the 2014 data for the common systems. The first approach only used the confidence scores of the underlying systems as input for classifying an

Baseline	Precision	Recall	F1
Union	0.067	**0.762**	0.122
Voting (threshold learned on 2013 data)	**0.641**	0.288	0.397
Oracle Voting (optimal threshold on 2014 data)	0.547	0.376	**0.445**

Table 4.1: Performance of baselines on all 2014 SFV dataset (65 systems)

Approach	Precision	Recall	F1
Union	0.176	**0.647**	0.277
Best ESF system in 2014 (Stanford)	0.585	0.298	0.395
Voting (threshold learned on 2013 data)	**0.694**	0.256	0.374
Oracle Voting (optimal threshold on 2014 data)	0.507	0.383	0.436
Stacking	0.606	0.402	0.483
Stacking + Slot-type	0.607	0.406	0.486
Stacking + Provenance (document) + Slot-type	0.653	0.400	0.496
Stacking + Provenance (document and offset) + Slot-type	0.541	0.466	**0.501**

Table 4.2: Performance on the common systems dataset (10 systems) for various configurations. All approaches except the Stanford system are our implementations.

instance. The second approach used the slot type along with the confidence scores. The KBP slot-filling task had approximately 40 slot types for each query. This allowed the system to learn different evidence-combining functions for different slots if the classifier found that useful. Our third approach included document provenance feature and our fourth approach used the offset provenance in addition to the other features. We used for the L1-regularized SVM with a linear kernel (other classifiers gave similar results) for training. Table 4.2 gives the performance of all our supervised approaches as well as our unsupervised baselines for the common systems dataset.

Systems change from one year to another and training on previous year's data might not be very intuitive. In order to have a better understanding of this, we plot a learning curve by training on different sizes of the 2013 data and then evaluating on the 2014 data for the common systems. Figure 4.4 shows the learning curve thus obtained. Although there are certain proportions of the dataset when the

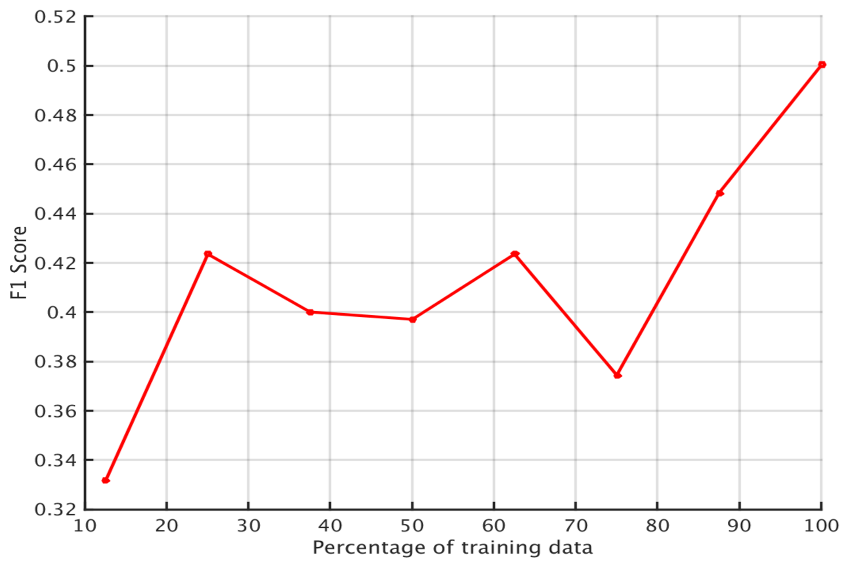


Figure 4.4: Learning curve for training on 2013 and testing on 2014 common systems dataset

F1 score drops which we suspect is due to overfitting 2013 data, there is still a strong correlation between the 2013 training data size and F1 score on the 2014 dataset. Thus we can infer that training on 2013 data is useful even though the 2013 and 2014 data are fairly different. Although the queries change, the common systems remain more-or-less the same and stacking enables a meta-classifier to weigh those common systems based on their 2013 performance.

The TAC KBP official scoring key for the ESF task includes human annotated slot fills along with the pooled slot fills obtained by all participating systems. However, Sammons et al. (2014) use an unofficial scoring key in their paper that does not include human annotated slot fills. In order to compare to their results, we also present results using the same unofficial key. Table 4.3 gives the performance of our baseline systems on the 2014 SFV dataset using the unofficial key for scoring. We note that our union does not produce a recall of 1.0 on the unofficial scorer due to our single-valued slot selection strategy for multiple systems. As discussed earlier for the single-valued slot, we include the slot-fill with the highest confidence

Baseline	**Precision**	**Recall**	**F1**
Union	0.054	**0.877**	0.101
Voting (threshold learned on 2013 data)	**0.637**	0.406	0.496
Voting (optimal threshold for 2014 data)	0.539	0.526	**0.533**

Table 4.3: Baseline performance on all 2014 SFV dataset (65 systems) using unofficial scorer

Approach	**Precision**	**Recall**	**F1**
Union	0.177	**0.922**	0.296
Voting (threshold learned on 2013 data)	**0.694**	0.256	0.374
Voting (optimal threshold for 2014 data)	0.507	0.543	0.525
Best published SFV result in 2014 (UIUC)	0.457	0.507	0.481
Stacking	0.613	0.562	0.586
Stacking + Relation	0.613	0.567	0.589
Stacking + Provenance(document)	0.498	0.688	0.578
Stacking + Provenance(document) + Slot-type	0.659	0.56	**0.606**
Stacking + Provenance(document and offset) + Slot-type	0.541	0.661	0.595

Table 4.4: Performance on the common systems dataset (10 systems) for various configurations using the unofficial scorer. All approaches except the UIUC system are our implementations.

(may not necessarily be correct) and thus may not match the unofficial scorer.

Table 4.4 gives the performance of all our supervised approaches along with the baselines on the common systems dataset using the unofficial key for scoring. UIUC is one of the two teams participating in the SFV 2014 task and the only team to report results, but they report 6 different system configurations and we show their best performance.

Our results indicate that stacking with provenance information and slot type gives the best performance. Our stacking approach that used the 10 systems common between 2013 and 2014 also outperformed the ensembling baselines that had the advantage of using *all 65* of the 2014 systems. Our stacking approach would have presumably performed even better if we had access to 2013 training data for all 2014 systems. Of course, the best-performing system for 2014 did not have access to the pooled slot fills of all participating systems. Although pooling the results has an advantage, naive pooling methods such as the ensembling baselines, in particular, the voting approach, do not perform as well as our stacked ensembles. Our best approach beats the best baseline for both the datasets by at least 6 F1 points using both the official and the unofficial scorer. As expected the *Union* baseline has the highest recall. Among the supervised approaches, stacking with document provenance produced the highest precision and is significantly higher (approximately 5%) than the approach that produced the second highest precision, stacking with just the slot-type auxiliary feature.

4.3.6 Experimental Evaluation on Cold Start Slot Filling (CSSF)

In 2015, NIST replaced the slot-filling task with the cold start slot filling (CSSF) task. The task became more challenging because the queries used were entities that did not have a Wikipedia entry. The CSSF task also introduced inverse of each slot type in the ontology and evaluation queries could take one of the two forms – single-hop as in the original slot-filling task or multiple-hops. Further, in 2016, the CSSF task became cross-lingual and was extended to two new languages, Spanish and Chinese, apart from English. We participated in both 2015 and 2016 versions of the CSSF task using our SWAF approach. We used the output of the shared systems from the previous years' iteration of the competition for training and the current year's output as the test. We had 10 shared systems between 2014 and 2015 and 8 shared systems between 2015 and 2016.

The 2015 CSSF task had a much smaller corpus of shorter documents compared to the previous year's slot-filling corpus (Ellis et al., 2015; Surdeanu and Ji, 2014). Thus, the provenance feature of Rajani et al. (2015) did not sufficiently capture the reliability of a slot fill based on where it was extracted. So, we introduced a new auxiliary feature. Slot filling queries were provided to participants in an XML format that included the query entity's ID, name, entity type, the document where the entity appears, and beginning and end offsets in the document where that entity appears. This allowed the participants to disambiguate query entities that could potentially have the same name but refer to different entities. Below is a sample query

Method	Precision	Recall	F1
Mixtures of Experts (ME) model	0.479	0.184	0.266
Oracle Voting baseline (3 or more systems must agree)	0.438	0.272	0.336
Top ranked CSSF system in 2015 (Angeli et al., 2015)	0.399	0.306	0.346
Stacking without auxiliary features	0.497	0.282	0.359
Stacking with just instance auxiliary features	0.498	0.284	0.360
Stacking with just provenance auxiliary features	**0.508**	0.286	0.366
Stacking with both provenance + instance auxiliary features	0.466	**0.331**	**0.387**

Table 4.5: Results on 2015 Cold Start Slot Filling (CSSF) task using the official NIST scorer

from the 2015 task:

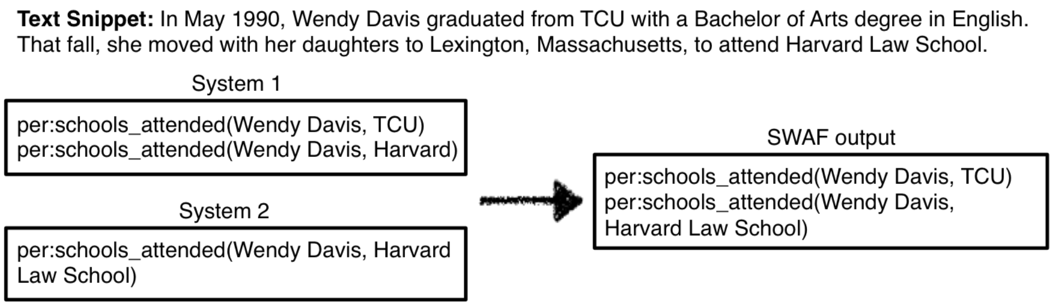
```
<query id="CSSF15_ENG_006">
  <name>Walmart</name>
  <docid>435</docid>
  <beg>232</beg>
  <end>238</end>
  <enttype>org</enttype>
  <slot0>org:date_dissolved</slot0>
</query>
```

The ⟨docid⟩ tag refers to the document where the query entity appears, which we will call the *query document*. We used an auxiliary feature that involved measuring the similarity between this *query document* and the *provenance document* that is provided by a given system. We represented the query and provenance documents as standard TF-IDF weighted vectors and used cosine similarity to compare documents. Therefore, every system that provided a slot fill, also provided the provenance for the fill and thus had a similarity score with the query document. If a system did not provide a particular slot fill then its document similarity score is simply zero. This feature is intended to measure the degree to which the system's provenance document is referencing the correct query entity.

Table 4.5 shows the performance on the 2015 version of the task. We evaluated and compared various ablations of the auxiliary features for stacking and found

Method	Precision	Recall	F1
Mixtures of Experts (ME) model	0.168	0.321	0.180
Oracle Voting baseline (4 or more systems must agree)	0.191	0.379	0.206
Top ranked CSSF system in 2016 (Zhang et al., 2016)	0.265	0.302	0.260
Stacking without auxiliary features	**0.311**	0.253	0.279
Stacking with just instance auxiliary features	0.257	0.346	0.295
Stacking with just provenance auxiliary features	0.252	0.377	0.302
Stacking with both provenance + instance auxiliary features	0.258	**0.439**	**0.324**

Table 4.6: Results on 2016 Cold Start Slot Filling (CSSF) task using the official NIST scorer


Text Snippet: In May 1990, Wendy Davis graduated from TCU with a Bachelor of Arts degree in English. That fall, she moved with her daughters to Lexington, Massachusetts, to attend Harvard Law School.

System 1
per:schools_attended(Wendy Davis, TCU)
per:schools_attended(Wendy Davis, Harvard)
System 2
per:schools_attended(Wendy Davis, Harvard Law School)
SWAF output
per:schools_attended(Wendy Davis, TCU)
per:schools_attended(Wendy Davis, Harvard Law School)

Figure 4.5: Illustration of our SWAF approach on an instance of the CSSF dataset. SWAF uses the auxiliary features including the text snippet above as provenance for classifying the instance.

that using both the instance and provenance auxiliary features gives us the best performance. We also compared our results to the Mixture of Experts (ME) ensembling approach that has a similar intuition as the instance auxiliary features of SWAF, i.e., some systems are better at certain instance types than other systems. We note that the dramatic drop in our performance when compared to the 2014 version of the task, is because of the challenges posed by the introduction of the CSSF task. Also, the NIST scorer had a very strict way of evaluating multiple-hop queries. So, for evaluating responses for such multi-hop queries, if a system generated an incorrect `hop-0` response then all its `hop-1` responses will be treated as incorrect even if they were actually correct.

Table 4.6 shows the results obtained on the 2016 CSSF task. We observed trends very similar to the 2015 results on the task. The CSSF task got more challenging each year and we found that the performance of all the systems was affected. However, our SWAF approach consistently beat the top ranking individual system. Figure 4.5 illustrates the SWAF approach on an actual instance of the CSSF dataset. Two systems extract the fill 'NY Red Bulls' for the entity 'Arsenal' and relation 'org:member_of' while one system extracts the fill 'Leeds United' for the same entity and relation. SWAF has learned to rely on system 2 based on the instance and provenance auxiliary features.

We note that the top ranked systems we compare to in Table 4.5 and Table 4.6 are not restricted to individual systems and could also be an ensemble. The top-ranked indicates the best performance obtained amongst all individual and ensemble systems in the competition.

4.4 Stacking with Auxiliary Features for Entity Linking

The knowledge base population (KBP) track of the Text Analysis Conference (TAC) conducted annually by NIST consisted of the entity discovery and linking (EDL) task along with slot-filling. The EDL task was cross-lingual including two foreign languages, Spanish and Chinese, along with English and was thus called the tri-lingual entity discovery and linking (TEDL). We will refer to the TEDL task generally as EDL unless specifically required. The objective of the EDL task is to discover entities based on a supplied text corpus as well as link these entities to an existing English knowledge base (KB) or cluster the mention with a NIL ID if the system could not successfully link the mention to any entity in the KB (Ji et al., 2015, 2016). A version of FreeBase (Bollacker et al., 2008) was used as the KB.

Our stacking with auxiliary features (SWAF) approach to EDL uses the KB ID (or NIL ID) as the handle for ensembling across system outputs. Systems link each detected entity *mention*, i.e., a string that references that entity in the text, to a KB or NIL ID. EDL systems optionally provide confidence scores as part of the output. In case a system does not provide confidence then we use a confidence score of 1.0. Also, if a system did not detect a particular entity mention then we use a confidence score of zero. For ensembling, we also need to define criteria for what counts as the *same* entity mention when multiple systems detect it. So, two systems are said to have detected and linked the same mention if the entity mentions overlap to any extent.

4.4.1 Provenance Features

For provenance features, we use the provenance information for each generated output instance. Provenance indicates “where” the system found the output in the source corpus. For EDL, if a system successfully links a mention to a KB ID, then it must provide the mention provenance indicating the origin of the mention in the corpus. The provenance is in the form of *docid* and *start offset – end offset* that gives the source document in the corpus and offsets in that document.

The provenance feature for the EDL task is similar to the provenance feature of the slot-filling task. It measures the substring overlap of entity mentions across systems using the Jaccard similarity coefficient. We used the aforementioned criteria for deciding if a mention is same or different across multiple systems for generating the provenance features.

4.4.2 Instance Features

We used a one-hot encoding of the entity type as instance features. There were a total of five pre-defined entity types for the EDL task. They were person (PER), organization (ORG), geo-political entity (GPE), facility (FAC) and location (LOC).

Another instance feature we used is similar to the new auxiliary feature for the CSSF task. The similarity measure between the entity’s *KB document* and its *mention document* are used as features. For every KB ID that ever occurred in the output, we created a pseudo-document consisting of the entity’s KB description as well as all relations involving the entity that exist in the KB. The document that systems provided as provenance was considered the mention document. We used cosine similarity measure to compare an entity’s KB and mention document

Method	Precision	Recall	F1
Oracle Voting baseline (4 or more systems must agree)	0.514	**0.601**	0.554
Top ranked TEDL system in 2015 (Sil et al., 2015)	0.693	0.547	0.611
Stacking without auxiliary features	0.729	0.528	0.613
Stacking with just instance auxiliary features	0.783	0.511	0.619
Stacking with just provenance auxiliary features	0.814	0.508	0.625
Stacking with both provenance + instance auxiliary features	**0.814**	0.515	**0.630**

Table 4.7: Results on 2015 Tri-lingual Entity Discovery and Linking (TEDL) task using the official NIST scorer and the CEAFm metric

represented as standard TF-IDF weighted vectors.

4.4.3 Post-processing

Once we obtain the decision on each input instance from the stacker, we perform some final post-processing to produce output that is in the same format as that generated by an individual system. For each entity mention link that is classified as correct, if the link is a KB ID then we include it in the final output, but if the link is a NIL ID then we keep it aside until all mention links are processed. Thereafter, we resolve the NIL IDs across systems since NIL IDs for each system are unique. We merge NIL clusters across systems into one if there is at least one common entity mention among them. Finally, we give a new NIL ID for these newly merged clusters.

4.4.4 Experimental Evaluation

We participated in the 2015 and 2016 versions of the EDL tasks. In both years, there were 6 shared systems with the previous year's iteration of the task. We trained our meta-classifier on the previous year's output of the common systems. A $L1$ regularized linear SVM weighted by the number of instances for each class worked best as a meta-classifier for stacking. The EDL evaluation uses the mention CEAF metric (Ji et al., 2015, 2016) for measuring precision, recall and F1. This metric finds the optimal alignment between system and gold standard clusters and

Method	Precision	Recall	F1
Oracle Voting baseline (4 or more systems must agree)	0.588	0.412	0.485
Mixtures of Experts (ME) model	0.721	0.494	0.587
Top ranked EDL system in 2016 (Sil et al., 2016)	0.717	0.517	0.601
Stacking without auxiliary features	0.723	0.537	0.616
Stacking with just instance auxiliary features	0.752	0.542	0.630
Stacking with just provenance auxiliary features	**0.767**	0.544	0.637
Stacking with both provenance + instance auxiliary features	0.739	**0.600**	**0.662**

Table 4.8: Results on 2016 Tri-lingual Entity Discovery and Linking (EDL) task using the official NIST scorer and the CEAFm metric

then evaluates precision and recall micro-averaged over mentions.

Table 4.7 and Table 4.8 show the results obtained on the 2015 and 2016 TEDL tasks respectively. Overall, we observe similar trends in performance in both years. Unlike the CSSF task, the ME ensemble is able to beat the best voting baseline for the TEDL task but it still does not beat the top-ranked system in the competition. The performance of the ME algorithm deteriorates as the number of component system increases. Thus, we can conclude that it is not robust to the number of component systems. Figure 4.6 illustrates the SWAF approach on an actual TEDL instance. Our approach has learned to rely on system 1 for the given instance based on the instance and provenance auxiliary features.

We note that for Slot-Filling, adding the auxiliary feature lowers precision but improves recall and is vice-versa for EDL. This trend is not much related to SWAF but mainly depends on the number of component systems and input instances. We have more component systems for slot-filling than EDL and these systems themselves have high variation of precision and recall. Similarly, the number of input instances is much higher for EDL than slot-filling.

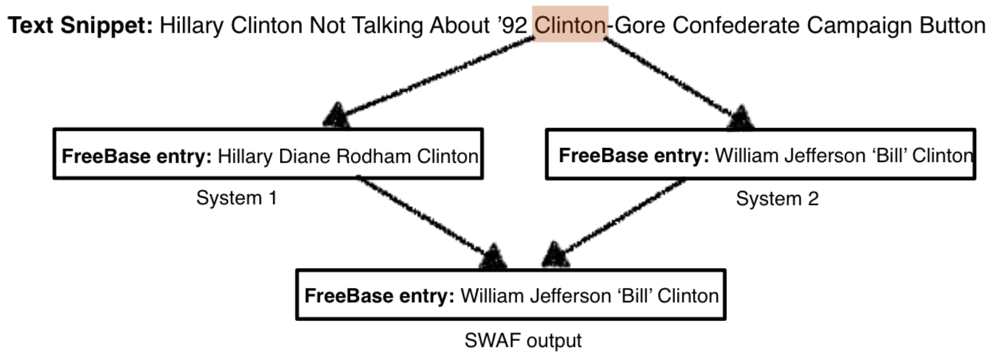

Text Snippet: Hillary Clinton Not Talking About '92 Clinton-Gore Confederate Campaign Button
FreeBase entry: Hillary Diane Rodham Clinton
System 1
FreeBase entry: William Jefferson 'Bill' Clinton
System 2
FreeBase entry: William Jefferson 'Bill' Clinton
SWAF output

Figure 4.6: Illustration of our SWAF approach on an instance of the TEDL dataset. One system linked the mention to a FreeBase entry for Hillary Clinton whereas the other system linked the same mention to Bill Clinton. SWAF uses the auxiliary features including the text snippet above as provenance for classifying the instance.

4.5 Combining Supervised and Unsupervised Ensembles for Knowledge Base Population

In our approach to using SWAF for the two knowledge base population (KBP) subtasks, we were only able to use shared systems that had historical training data. However, in some situations, we would like to ensemble systems for which we have no historical performance data. Towards this end, we use the variant of SWAF proposed in Section 3.4 that first uses unsupervised ensembling to combine systems without training data, and then uses stacking to combine this ensembled system with other systems for which training data is available.

Using this new SWAF variant, we demonstrated new state-of-the-art results on the two KBP subtasks – *Cold Start Slot-Filling* CSSF)[2] and the *Tri-lingual Entity Discovery and Linking* (TEDL) (Ji et al., 2015). Our approach outperformed the best individual system as well as other ensembling methods such as stacking only the shared systems, which was our previous supervised approach (Rajani and Mooney, 2017b). As part of this work, we also proposed a new auxiliary feature for the CSSF and TEDL tasks and verified that incorporating them in the combined

[2]`http://www.nist.gov/tac/2015/KBP/ColdStart/guidelines.html`

approach improved performance.

4.5.1 Unsupervised Ensembling Approach

Only 38 of the 70 systems that participated in CSSF 2015 also participated in 2014, and only 24 of the 34 systems that participated in TEDL 2015 also participated in the 2014 EDL task. Therefore, many KBP systems in 2015 were new and did not have past training data needed for the supervised approach. In fact, some of the new systems performed better than the shared systems, for example, the *hltcoe* system did not participate in 2014 but was ranked 4^{th} in the 2015 TEDL task (Ji et al., 2015). We first ensembled these unsupervised systems using the constrained optimization approach described by Wang et al. (2013). Their approach is specific to the English slot-filling task and also relies a bit on past data for identifying certain parameter values. Below we describe our modifications to their approach so that it can be applied to both KBP tasks in a purely unsupervised manner.

The approach in Wang et al. (2013) aggregates the "raw" confidence values produced by individual KBP systems to arrive at a single aggregated confidence value for each output instance. Suppose that $V_1, \ldots, V_M$ are the M distinct values produced by the systems and N_i is the number of times the value V_i is produced by the systems. Then the aggregated confidence value is produced by solving the following optimization problem:

$$\min_{0 \leq x_i \leq 1} \sum_{i=1}^{M} \sum_{j=1}^{N_i} w_{ij} \left(x_i - c_i\left(j\right)\right)^2 \tag{4.2}$$

where c_i denotes the raw confidence score, x_i denotes the aggregated confidence score for V_i and $w_{ij} \geq 0$ is a non-negative weight assigned to each instance. Equation 4.2 ensures that the aggregated confidence score is close to the raw score as well as proportional to the agreement among systems on a given output instance. Thus if a system's output instance is also produced by multiple other systems, it would have a higher score than if it were not produced by any other system. Wang et al. (2013) used the inverse ranking of the average precision previously achieved by individual systems as the weights in the above equation. However, since we used this approach for systems that did not have historical training data, we used uniform weights across all unsupervised systems for both the tasks.

Equation 4.2 is subject to certain constraints on the confidence values depending on the task. For the slot-filling task, there are two different constraints based on whether the slot type is single-valued or list-valued. For single-valued slot types, only one slot value can be correct and thus the constraint is based on the mutual exclusion property of the slot values:

$$P(V_1) + P(V_2) + \cdots + P(V_M) \leq 1 \tag{4.3}$$

This constraint allows only one of the slot values to have a substantially higher probability compared to rest. On the other hand, for list-valued slot types, the RHS in the above equation is replaced by the value $\frac{n_c}{n}$ where n_c is the average number of correct slot fills for that slot type across all entities in the previous year and n is the total number of slot fills for that slot type across all entities. This approach to estimating the number of correct values can be thought of as *collective precision* for the slot type achieved by the set of systems. For the newly introduced slot inverses in 2015, we used the same ratio as that of the corresponding original slot type. Thus the slot type *per:parents* (new slot type) would have the same ratio as that of *per:children*.

For the TEDL task, we used the KB ID for identifying an instance and therefore use the entity type for defining the constraint on the entity mentions. For each of the entity types (PER, ORG, and GPE) we replaced the quantity on the right-hand side in Equation 4.3 by the ratio of the average number of correct mentions for that entity type in 2014 to the total number of mentions for that entity type, across all entities. For the two new entity types introduced in 2015 (FAC and LOC), we used the same ratio as that of GPE because of their semantic similarities.

The output from this approach for both tasks is a set of unique instances with aggregated confidence scores across all unsupervised systems which go directly into the stacker. Using the aggregation approach as opposed to directly using the raw confidence scores allows the classifier to meaningfully compare confidence scores across multiple systems although they are produced by very diverse systems.

Another technique for unsupervised ensembling that we experimented with in place of the constrained optimization approach is the Bipartite Graph-based Consensus Maximization (BGCM) approach by Gao et al. (2009) which was discussed in detail in Section 2.2.

4.5.2 Combining Supervised and Unsupervised Ensembles

We combine the supervised and unsupervised methods using a stacked meta-classifier as the final arbiter on a given input instance. The outputs from the supervised and unsupervised systems are fed into the stacker in a consistent format so that there is a unique input tuple. For the CSSF task, it is in the form $\langle$query entity $+$ relation, slot fill$\rangle$ and for the EDL task it is $\langle$KB or NIL ID, mention$\rangle$. Most KBP teams submit multiple variations of their system. Before ensembling, we first combine multiple runs of the same team into one. In this way, for the CSSF task, we obtained 10 systems (one for each team) for which we have supervised data for training stacking and 13 systems for which we used unsupervised ensembling. Similarly, for the TEDL task, we obtained 6 teams that had 2014 training data and 4 teams that did not have training data.

The unsupervised method produced aggregated, calibrated confidence scores which go directly into our final meta-classifier. We treat this combination as a single system called the *unsupervised ensemble*. We add the unsupervised ensemble as *one* additional system to the stacker giving us a total of 11 CSSF and 7 TEDL systems respectively. Once we have extracted the auxiliary features for each of the supervised systems and the unsupervised ensemble for both years, we trained the stacker on 2014 systems and tested on the 2015 systems. The unsupervised ensemble for each year is composed of different systems, but hopefully, the stacker learns to combine a generic unsupervised ensemble with the supervised systems that are shared across years. This allows the stacker to arbitrate the final correctness of an input instance tuple, combining systems for which we have no historical data with systems for which training data *is* available. To learn the meta-classifier, we used an L1-regularized SVM with a linear kernel (Fan et al., 2008) (other classifiers gave similar results).

Once we obtained the decisions from the stacker on every output instance, we performed a final post-processing to get the aggregated output that looks like it was produced by a single system and there are no conflicts. All the instances classified as correct by the classifier are kept while those that are classified as incorrect are discarded. The correct instances are then processed based on the task at hand and the process is exactly the same as with the SWAF approach.

4.5.3 New Auxiliary Feature

Our new auxiliary feature measures the document similarity between the *provenance documents* that different systems provide. We note that this is different from the auxiliary features introduced in Section 4.3.6 that measure the similarity between query and provenance documents. Suppose for a given query and slot type, n systems provide the same slot fill. For each of the n systems, we measure the average document cosine similarity between the system's provenance document and those of the other $n-1$ systems. Our previous document provenance feature simply measured whether systems agreed on the exact provenance document. By softening this to take into account the *similarity* of provenance documents, we hope to have a more flexible measure of provenance agreement between systems. We used these similarity features for both the CSSF as well as the TEDL tasks, except that the TEDL task does not have a query document and so we used the FreeBase entry for the mention. We created a *pseudo* document for each KB ID by appending the FreeBase definition, description, and relations of the entity. In this way, the new auxiliary feature captures similarity between the provenance document and the *pseudo* FreeBase document.

4.5.4 Experimental Evaluation

Table 4.9 and Table 4.10 show the results obtained using stacking for combining supervised and unsupervised ensembles on the CSSF and TEDL tasks respectively. Our full system, which combines supervised and unsupervised ensembling performed the best on both tasks. The 2015 iteration of TAC-KBP also in-

Methodology	**Precision**	**Recall**	**F1**
Constrained optimization approach (Wang et al., 2013)	0.1712	0.3998	0.2397
Oracle Voting baseline (3 or more systems must agree)	0.4384	0.2720	0.3357
Top ranked CSSF system in 2015 (Angeli et al., 2015)	0.3989	0.3058	0.3462
SWAF approach (Rajani and Mooney, 2017b)	0.4656	0.3312	0.3871
BGCM for combining supervised and unsupervised systems	0.4902	0.3363	0.3989
Stacking using BGCM instead of constrained optimization	**0.5901**	0.3021	0.3996
Top ranked SFV system in 2015 (Rodriguez et al., 2015)	0.4930	0.3910	0.4361
Combined stacking and constrained optimization	0.4679	**0.4314**	**0.4489**

Table 4.9: Results on 2015 Cold Start Slot Filling (CSSF) task using the official NIST scorer

cluded the Slot Filler Validation (SFV) task[3] where the goal is to ensemble/filter outputs from multiple slot filling systems. The top-ranked system in 2015 (Rodriguez et al., 2015) does substantially better than many of the other ensembling approaches, but it does not do as well as our best performing system. The purely supervised approach using auxiliary features of (Rajani and Mooney, 2017b) performs substantially worse, although still outperforming the top-ranked individual system in the 2015 competition. These approaches only use the common systems from 2014, thus ignoring approximately half of the systems. The approach of Wang et al. (2013) performs very poorly by itself, but when combined with stacking gives a boost to recall and thus the overall F1. The performance of the constrained optimization approach is obtained using purely unsupervised form of the method proposed by Wang et al. (2013) but because it is used in a supervised way by SWAF (by training on the unsupervised systems of the previous year), we obtain particularly high recall.

The oracle voting baseline also performs very poorly indicating that naive ensembling is not advantageous. For the TEDL task, the relative ranking of the approaches is similar to those obtained for CSSF, proving that our approach is very general and improves performance on two quite different and challenging problems.

Even though it is obvious that the boost in our recall was because of adding the unsupervised systems, it isn't clear how many new instance tuples were gen-

[3]`http://www.nist.gov/tac/2015/KBP/SFValidation/index.html`

Methodology	Precision	Recall	F1
Constrained optimization approach (Wang et al., 2013)	0.445	0.176	0.252
Oracle Voting baseline (4 or more systems must agree)	0.514	0.601	0.554
Top ranked TEDL system in 2015 (Sil et al., 2015)	0.693	0.547	0.611
SWAF approach (Rajani and Mooney, 2017b)	0.813	0.515	0.630
BGCM for combining supervised and unsupervised outputs	0.810	0.517	0.631
Stacking using BGCM instead of constrained optimization	0.803	0.525	0.635
Combined stacking and constrained optimization	0.686	**0.624**	**0.653**

Table 4.10: Results on 2015 Tri-lingual Entity Discovery and Linking (TEDL) using official NIST scorer and CEAF metric

erated by these systems. We, therefore, evaluated the contribution of the systems ensembled using the supervised approach and those ensembled using the unsupervised approach, to the final combination for both the tasks. Figure 4.7 shows the number of unique as well as common instance tuples that were contributed by each of the approaches. The unique instances are those that were produced by one approach but not the other and the common instances are those that were produced by both approaches. We found that approximately one-third of the input instances in the combination came from the unique instances produced just by the unsupervised systems for both the CSSF and TEDL tasks. Only about 22% and 15% of the total input instances were common between the two approaches for the CSSF and TEDL tasks respectively. Our findings highlight the importance of utilizing systems that do not have historical training data.

4.6 Chapter Summary

We demonstrated that our approach is a general, effective approach by applying it to two challenging NLP problems requiring structured output: relation extraction and entity linking. SWAF obtained very promising results on various versions of both the tasks, outperforming the best component systems as well as other ensembling methods. We achieved a new state-of-the-art on both the KBP tasks approach with an overall F1 score of 32.4% and CEAFm F1 of 66.2% on the

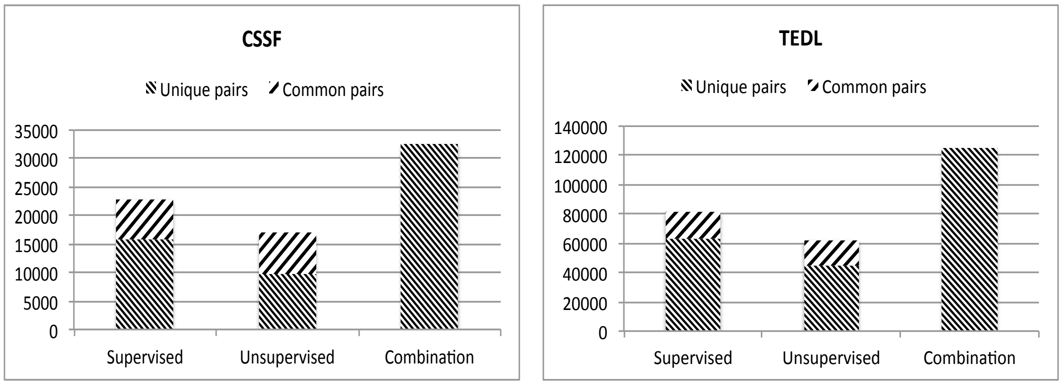


Figure 4.7: Total number of unique and common input pairs contributed by the supervised and unsupervised systems to the combination for the CSSF and TEDL tasks respectively.

2016 KBP CSSF and EDL tasks respectively.

We observed that the well known mixture-of-experts method is not robust because of its assumption that the underlying systems are trained on different feature sub-spaces, which is not always the case. On further analyzing the results obtained by SWAF, we found that it does better when the component outputs differ widely and have low confidences. The gain in performance from SWAF comes from output decisions that are difficult to make without context; however, using auxiliary features enables fusion of additional relevant information, allowing the stacker to make the right decision.

We also introduced a novel Stacking-based approach to ensembling both supervised and unsupervised systems and demonstrated promising results on the two KBP tasks. The approach outperformed the top-ranked systems from both 2015 competitions as well as several other ensembling methods, achieving a new state-of-the-art for both of these important, challenging tasks. We found that adding the unsupervised ensemble along with the shared systems specifically increased recall substantially.

Chapter 5
Stacking with Auxiliary Features for Computer Vision

This chapter demonstrates the application of stacking with auxiliary features on two well-known vision tasks – object detection and visual question answering. Parts of the work in this chapter have been published in (Rajani and Mooney, 2017b) and (Rajani and Mooney, 2018). All work in this chapter constitutes original contributions.

5.1 Chapter Overview

The previous chapter demonstrated the success of stacking with auxiliary features (SWAF) on two very important natural language understanding problems – information extraction and entity linking. In this chapter, we examine the SWAF approach on well-known computer vision problems. In object detection, algorithms produce a list of object categories present in the image along with an axis-aligned bounding box indicating the position and scale of every instance of each object category detected (Russakovsky et al., 2015). Object detection has been part of the annual ImageNet Large Scale Visual Recognition Competition (ILSVRC). Visual Question Answering (VQA) is also an annual competition at the intersection of vision and language (Antol et al., 2015). Given an image and a natural language question about the image, the task is to provide an accurate natural language answer.

SWAF (Rajani and Mooney, 2017b) is a type of ensembling algorithm which learns to combine outputs of multiple systems using features of the current problem as context. In this chapter, we use SWAF to more effectively combine several object detection and VQA models. Our approach extracts features from the images for the object detection problem and from image-question (IQ) pairs for the VQA problem, as well as the component models for both tasks and provides this information to the classifier. The meta-classifier then learns to predict whether the generated output is correct or not. We obtain significant improvements over the component models for both the 2015 object detection and the 2016 VQA competitions.

5.2 Prior Work

Prior work related to stacking and ensembling in general was discussed in the previous chapter. In this chapter, we focus on work related to object detection and visual question answering (VQA).

ImageNet Object detection: The scale of the ImageNet dataset is so large that almost all participating teams deploy deep learning for object detection. The top-performing team in the 2015 iteration of the competition used a deep residual net (He et al., 2016) and several other teams deployed a version of faster R-CNN (Region based Convolutional Neural Network) with selective search (Ren et al., 2015). Faster R-CNN is a more efficient variant of fast R-CNN (Girshick, 2015) that first uses Region Proposal Networks (RPN) to train an end-to-end network for generating region proposals. The region proposals are then used by the R-CNN for detection. In our work, we also use the deformable parts model based on histogram of oriented gradients (HOG) (Dalal and Triggs, 2005), as a component model for object detection. There has been some work on stacking for multi-layer object recognition (Peppoloni et al., 2014) but our work is the first to use stacking for ensembling object *detectors* and we obtain significant improvements over the component systems.

Visual question answering: A variety of methods to address the VQA challenges of image understanding, language grounding and common-sense knowledge capabilities have been developed in recent years (Andreas et al., 2016a; Fukui et al., 2016; Xu and Saenko, 2016; Lu et al., 2016; Chen et al., 2015). The vision component of a typical VQA system extracts visual features using a deep convolutional neural network (CNN), and the linguistic component encodes the question into a semantic vector using a recurrent neural network (RNN). An answer is then generated conditioned on the visual features and the question vector. The top-performing VQA systems are ensembles of neural networks that perform significantly better than any of the underlying individual models.

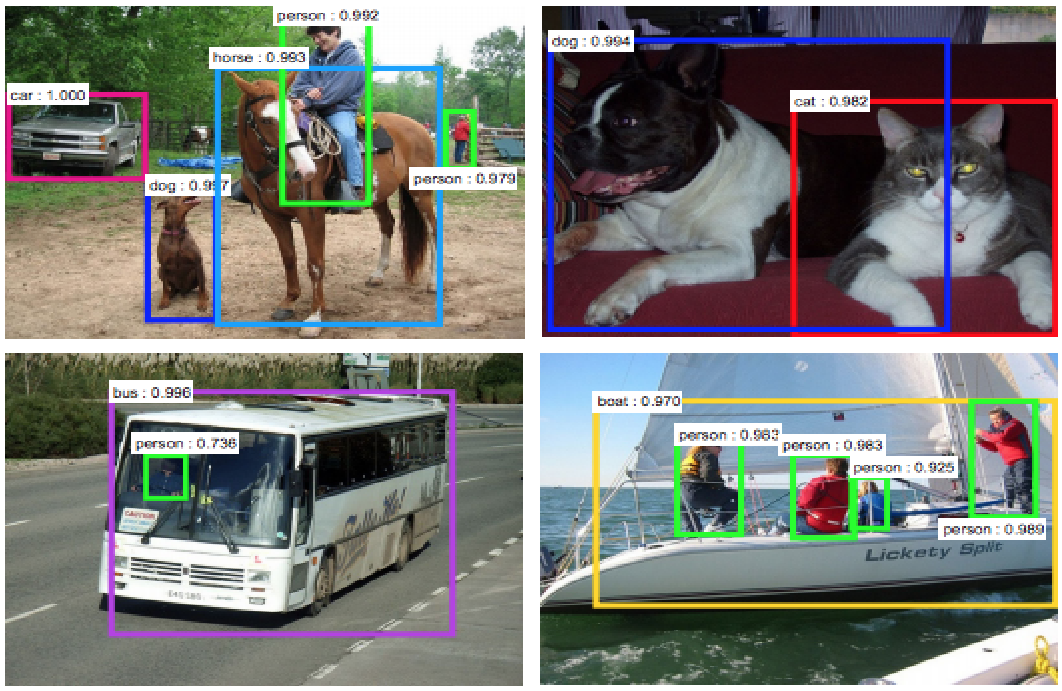


Figure 5.1: ImageNet Object detection sample images with bounding boxes around object categories.

5.3 Stacking with Auxiliary Features for Object Detection

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a widely known annual competition in Computer Vision and has become the standard benchmark for large-scale object recognition (Russakovsky et al., 2015). The goal of the ImageNet object detection task is to detect all instances of object categories (out of the 200 predefined categories) present in the image and localize them by providing coordinates of the axis-aligned bounding boxes for each instance. The output for the task is the image ID, the object category (1-200), a confidence score, and the coordinates of the bounding box. In case of multiple instances in the same image, each instance is mentioned on a separate line. Figure 5.1 shows a random sample of images with bounding boxes indicating the location and scale of the detected objects.

5.3.1 Auxiliary features for object detection

We use two types of auxiliary features for the object detection task – the provenance and the instance auxiliary features.

Provenance features

Recollect that the intuition behind using provenance information in auxiliary features is that an output is more reliable if multiple systems agree not only on the decision itself, but also on its provenance. For the provenance feature, we used the Jaccard coefficient to measure the overlap between bounding boxes across systems. If a system successfully detects a target category then it must provide a bounding box localizing the object in the image. The bounding box is in the form $\langle x_{min}, y_{min}, x_{max}, y_{max} \rangle$ and is similar to provenance for the KBP tasks as discussed in the previous chapter, giving *where* in the input is the information supporting the conclusion. For a given image ID, if N systems detect the same object instance, then the bounding box overlap (BBO) score for a system n is calculated as the intersection of the areas of bounding boxes, divided by their union:

$$\text{BBO(n)} = \frac{1}{|\text{N}| - 1} \times \sum_{i \in N, i \neq n} \frac{|\text{Area(i)} \cap \text{Area(n)}|}{|\text{Area(i)} \cup \text{Area(n)}|} \tag{5.1}$$

Figure 5.2 shows a cartoon example of how the provenance feature is calculated based on an object instance in an image. We note that the use of provenance as features does not require access to the large corpus of images and is thus computationally inexpensive.

Instance features

The idea behind using the *instance auxiliary features* is that some models learn to better localize object categories with certain attributes (e.g.: furry objects) than other systems. For example, a classifier could learn not to trust an object detection output for the target class ‘dog’ from a system that is not good at detecting dogs. The instance features enable the stacker to learn such patterns using task-specific

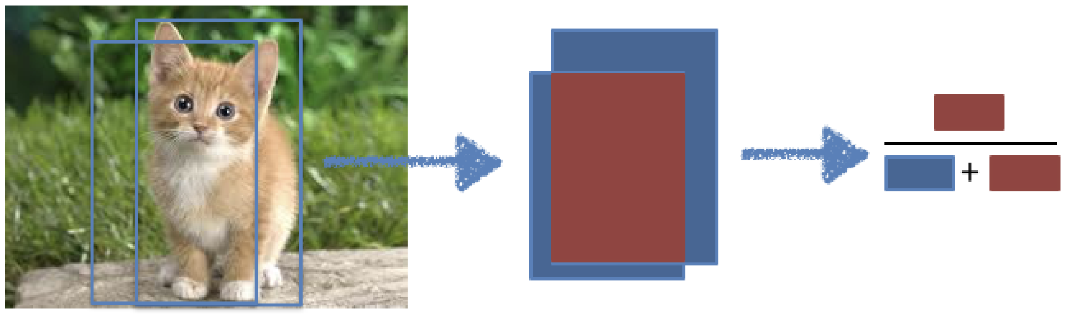


Figure 5.2: The provenance feature for a system is calculated as the intersection (red area) over union (red plus blue areas) of the area of its bounding box with every other system's bounding box for an object instance.

information in conjunction with the provenance features. We used two types of instance features. First, we used a one-hot encoding of the object category (total 200) as instance features. Second, we used a bag of visual words for the image using Scale-Invariant Feature Transform (SIFT) as the feature descriptor (Lowe, 2004) as well as the $4,096$ features from VGGNet's $fc7$ layer (Simonyan and Zisserman). Note that some underlying object detection systems also used these features for classification and we show that using them for learning the top-level meta-classifier further boosts the performance.

Instance formation

We used the image ID as a handle for ensembling object instances in an image from multiple systems. The component systems each could possibly detect multiple instances of an object category and it is important to discriminate between each of the instances in order to have a coherent output. Towards this end, we defined what constituted the same instance of an object across multiple system' outputs as follows. For a given image, two systems are said to have detected the same object instance if the intersection over union (IOU) of the areas of their bounding boxes (BB) is greater than 0.5. If the outputs don't meet this criteria for a given image, then they are considered to be two different instances of the same object in that image. The choice for the IOU of the BB to be greater than 0.5 to be considered the same instance is based on the evaluation metric of the object detection competition, i.e., a system is said to have detected an object instance correctly if the IOU of it's BB with the ground truth exceeds a threshold of 0.5.

The confidence scores along with the provenance and instance auxiliary features of every component system form the input to the stacker. Each object instance that is classified as correct by the stacker is included in the final output. The bounding box of the ensemble is calculated as follows. If multiple systems detected the object instance, then we sum the overlapping areas between a system's bounding box and that of every *other* system that also detected the exact same instance and we do this for every such system. The system with the *maximum* sum has a bounding box with the maximum overlap with other systems, and thus is used as the bounding box for the ensemble. When there are only two systems that produced an output, this method does not discriminate so we use the bounding box produced by the system with the higher confidence score. The reason we did not average the bounding boxes of the component system to produce the bounding box of the ensemble is that if a component system's bounding box does not overlap with the other systems' bounding boxes, averaging mostly results in a bounding box which does not localize the object instance correctly. We also experimented with using the *union*, *intersection*, and average of the bounding boxes across systems as the aggregate bounding box for the ensemble, but this approach is heavily penalized by the ImageNet evaluation metric. The union and the intersection mainly failed because they localized too much or too little of the object instance and did not meet the IOU threshold of 0.5 during evaluation as discussed above. The average mostly failed when one of the system's bounding box had little overlap with the bounding boxes of all the other system's and this led to the average bounding box being offset as well and therefore not meeting the evaluation threshold. A weighted average approach to combining bounding boxes would be an interesting future direction to explore.

5.3.2 Component object detection models

Our object detection ensemble using stacking with auxiliary features (SWAF) is comprised of three individual component models. We used two state-of-the-art deep neural models trained on the ImageNet object-detection training set, the ZF and the VGG models (Ren et al., 2015). We also use the Deformable Parts Model (DPM) (Felzenszwalb et al., 2010) with selective search for object detection, to produce a final ensemble of three systems. We ran these models on the validation set using the faster-RCNN method (Ren et al., 2015) with selective search (Uijlings et al., 2013) using the Caffe system (Jia et al., 2014). DPM is very slow to test and was unable to process the entire test set on all 200 categories. Therefore, we performed 10-fold cross-validation on the validation set, testing our approach on a total of about $20K$ images. We have discussed each of these three models in detail below.

The ZFNet model

The Zeiler and Fergus (ZF) (Zeiler and Fergus, 2014) is a well known convolutional neural network model in computer vision. It is an improvement on AlexNet obtained by tuning the architecture hyperparameters, in particular, the ZFNet uses filters of size 7×7 in the first layer instead of the 11×11 sized filters of the AlexNet. A smaller filter size in the first convolutional layer helps retain a lot of original pixel information. The ZFNet has 5 convolutional layers followed by three fully connected layers.

The VGGNet

The VGGNet (Simonyan and Zisserman) is a 16 layer convolutional neural network that uses a 3×3 filter size. It has 13 convolutional layers and the last three fully connected layers. The VGGNet was very influential because it emphasized that CNNs should have a deep network of layers for hierarchical representation of visual data, such as images, to actually work.

Both the ZF and VGGNet networks are used for object detection in a similar way by first generating region proposals using a region proposal network (RPN) (Ren et al., 2015) which are then used by a region-based convolutional neural network (R-CNN) (Girshick, 2015) for detection. The RPN is a fully connected convolutional network that simultaneously predicts object bounds and objectness scores at each position in an image. The RPN is trained end-to-end to generate high-quality region proposals which are used by the R-CNN. The RPN and the R-CNN are merged into a single network by sharing their convolutional features using *attention* which is a way for the RPN component to tell the unified network where to look. We use a faster version of R-CNN called the Faster R-CNN (Ren et al., 2015) for more efficient object detection.

Deformable Parts Model (DPM)

The DPM is a type of Markov random field that models an object as a collection of spatially constrained parts (Felzenszwalb et al., 2010). An object detector based on DPM will first find a match for the whole object and then model the parts to fine-tune the result. A DPM based object detector uses a sliding window approach, where a filter is applied at all positions and scales of an image. The detector is like a classifier which takes as input an image, a position within that image, and a scale and determines whether or not there is an instance of the target category at the given position and scale. The object detector uses low-level features based on the histogram of oriented gradients (HOG) to represent an object category.

The DPM was a “non-deep” and more traditional and computationally slow model used in our ensemble. Although more recently Girshick et al. (2015) showed that a DPM could be formulated as a CNN. The authors unrolled the DPM inference algorithm and mapped each step to an equivalent CNN layer. We, however, use the traditional DPM that is a graphical model and not a neural network. The reasoning is that the errors between the DPM and the CNN networks are de-correlated and we hypothesized that our meta-classifier could exploit this information to significantly outperform even while using a “non-deep” object detector.

5.3.3 Experimental Results

In this section, we present experimental results on using SWAF for object detection and compare our results to other individual as well as ensemble systems. We used an SVM with an RBF kernel as a meta-classifier. A small random sample of the training set (10%) was used as validation data to set the hyper-parameters. The ImageNet challenge evaluates the object detection task using average precision (AP) on a precision-recall curve. The predicted bounding box for detection is considered correct if its intersection over union with the ground truth exceeds a threshold of 0.5 (Russakovsky et al., 2015). The official scorer gives the AP for each of the 200 classes along with the median and mean AP. We report the median AP and mean AP (mAP). We compare our results to stacking without using any auxiliary features and various ablations of the provenance and instance auxiliary features. We were unable to obtain the state-of-the-art system and thus it was not part of our ensemble. For this reason, we compare our results to the best performing *component* system. Our results also include the "oracle" voting baseline for ensembling the system outputs. For this approach, we vary the threshold on the number of systems that must agree to identify an "oracle" threshold that results in the highest F1 score by plotting a precision-recall curve and finding the best F1. An optimal threshold of 1 gave the best voting performance on the detection task. We note that oracle voting is "cheating" to give the best possible voting baseline. We also compare to the Mixture of Experts (ME) (Jacobs et al., 1991) ensemble system that was discussed in detail in the previous chapter. Recollect that the reason we compare to the ME ensembling approach is because of its similarity to the instance features for our SWAF approach, i.e., some models are better at certain instance types than other models.

Table 5.1 shows the results for the ImageNet 2015 object detection task. Using stacking with both types of auxiliary features beats the best individual component system as well as the oracle voting baseline significantly. For the voting baseline, we consider an object instance to be the same if the systems' bounding boxes have IOU greater than 0.5. If we were able to use the top-ranked system

Method	Mean AP	Median AP
Oracle Voting baseline (1 or more systems must agree)	0.366	0.368
Best component system (VGG + selective search)	0.434	0.430
Stacking without auxiliary features	0.451	0.441
Stacking with just instance features	0.461	0.450
Mixtures of experts (ME) model	0.494	0.489
Stacking with just provenance auxiliary features	0.502	0.494
Stacking with provenance + instance auxiliary features	**0.506**	**0.497**

Table 5.1: Results on 2015 ImageNet object detection task using the official ImageNet scorer.

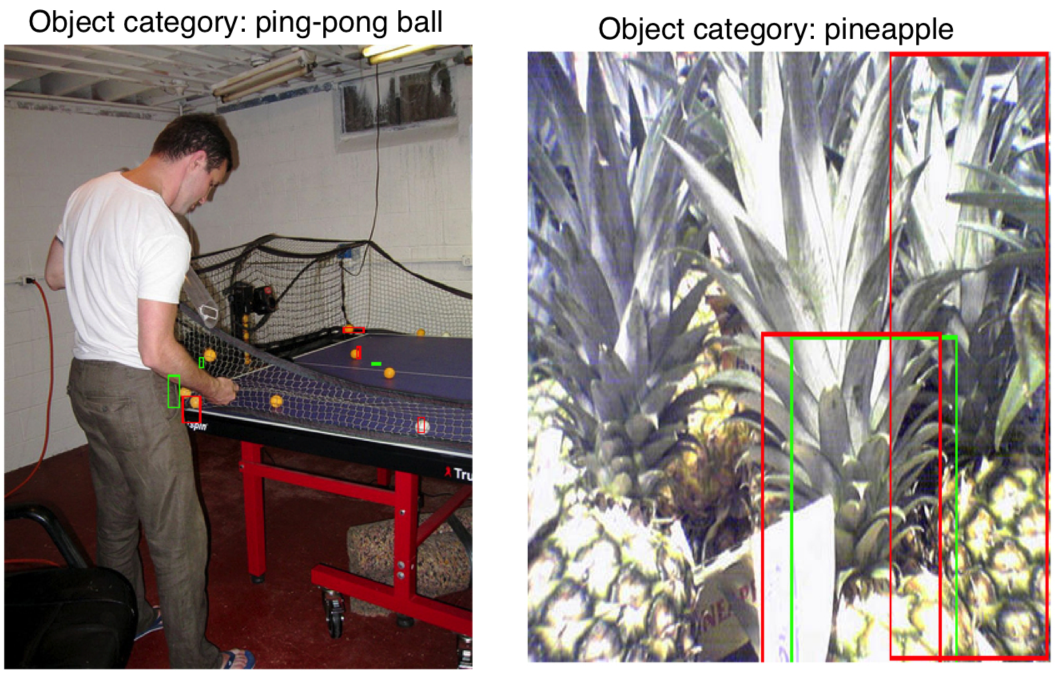
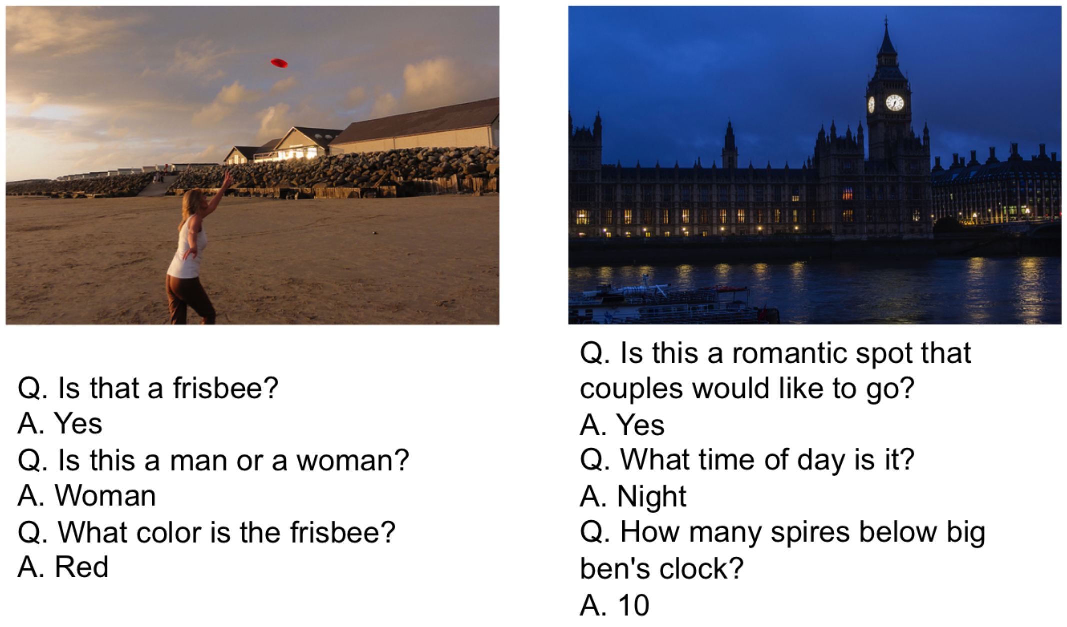


Figure 5.3: Random sample of outputs obtained on the 2015 ImageNet object detection task. The green bounding boxes are generated by the individual systems and among those, SWAF is able to identify the bounding boxes that are actually correct indicated by red.

from the competition as part of our ensemble, we would expect to obtain a new state-of-the-art result. Since we use cross-validation for obtaining these results, we performed a pairwise t-test with significance level 0.05 and found that using any ablation of stacking with auxiliary features is significantly better (p-value< 0.05) than using the best component system, although using stacking alone is not significantly worse compared to stacking with *any* auxiliary features. The ME algorithm performs significantly better than the individual components since it works well given the small number of component systems (i.e. 3). On analyzing the results, we found that the AP of several object classes differed widely across systems and even more so between the deep systems and DPM. Using SWAF, the meta-classifier learns to discriminate systems based on the auxiliary features and is thus able to leverage the best aspects of each individual system. An analysis of the results showed that SWAF particularly does well on localizing objects in images that have multiple instances of the *same* object, i.e. the image could be considered to be "cluttered". Combining diverse systems intelligently is one of the reasons why SWAF does particularly well on images with multiple instances of the same object category. Figure 5.3 demonstrates SWAF on a sample of the ImageNet object detection dataset.

5.4 Stacking with Auxiliary Features for VQA

VQA is the task of answering a natural language question about the content of an image by returning an appropriate word or phrase. Figure 5.4 shows a sample of images and questions from the VQA 2016 challenge. The dataset consists of images taken from the MS COCO dataset (Lin et al., 2014) with three questions and answers per image obtained through Mechanical Turk (Antol et al., 2015). Table 5.2 summarizes the splits in the VQA dataset. Several deep learning models have been developed that combine a computer vision component with a linguistic component in order to solve the VQA challenge. Some of these models also use data-augmentation for pre-training. To the best of our knowledge, there has been no prior work on stacking for VQA, and we are the first to show how model-specific explanations can serve as an auxiliary feature. The auxiliary features that we use are motivated by an analysis of the VQA dataset and also inspired by other related work such as using a Bayesian framework to predict the form of the answer from



Q. Is that a frisbee?
A. Yes
Q. Is this a man or a woman?
A. Woman
Q. What color is the frisbee?
A. Red

Q. Is this a romantic spot that couples would like to go?
A. Yes
Q. What time of day is it?
A. Night
Q. How many spires below big ben's clock?
A. 10

Figure 5.4: Random sample of images with questions and ground truth answers taken from the VQA dataset.

the question (Kafle and Kanan, 2016).

For stacking VQA systems, we first form unique question-answer pairs across all of the systems' outputs before passing them through the stacker. If a system generates a given output, then we use its probability estimate for that output, otherwise, the confidence is considered zero. If a question-answer pair is classified as correct by the stacker, then if there are other answers that are also classified as correct for the same question, the output with the highest meta-classifier confidence is chosen. For questions that do not have any answer classified as correct by the stacker, we choose the answer with lowest classifier confidence, which means it is least likely to be incorrect. This is because the VQA evaluation requires that participating systems submit "some" answer to every question in the test set.

The confidence scores along with other auxiliary features form the complete set of features used by the stacker. The auxiliary features are the backbone of the SWAF approach, enabling the stacker to intelligently learn to rely on systems' outputs conditioned on the supporting evidence. We use a total of four different cat-

	Images	Questions
Training	82,783	248,349
Validation	40,504	121,512
Test	81,434	244,302

Table 5.2: VQA dataset splits.

egories of auxiliary features for VQA. Three of these can be inferred directly from the image-question (IQ) pair and do not require querying the individual models. For the fourth category of auxiliary features, we generate *visual explanations* for the component models and use these to develop auxiliary features. The auxiliary features used in our VQA ensemble model are discussed below.

5.4.1 Auxiliary Features for VQA

This section discusses the auxiliary features inferred from the IQ pairs and the next section focuses on using explanation as auxiliary features.

Question and Answer Types

Antol et al. (2015) analyzed the VQA data and found that most questions fall into several types based on the first few words (e.g., questions beginning with "What is...", "Is there...", "How many...", or "Does the..."). Using the validation data, we discover such lexical patterns to define a set of question types. The questions were tokenized and a question type was formed by adding one token at a time, up to a maximum of five, to the current substring. The question "What is the color of the vase?" has the following types: "What", "What is", "What is the", "What is the color", "What is the color of". The prefixes that contain at least 500 questions were then retained as types. We added a final type "other" for questions that do not fall into any of the predefined types, resulting in a total of 70 question types. A 70-bit vector is used to encode the question type as a set of auxiliary features.

The original analysis of VQA answers found that they are 38% "yes/no" type and 12% numbers. There is clearly a pattern in the VQA answers as well, and we use the questions to infer some of these patterns. We considered three answer types – “yes/no,” “number,” and “other”. The answer-type auxiliary features are encoded using a one-hot vector. We classify all questions beginning with “Does”,“Is”,“Was”,“Are”, and “Has” as “yes/no”. Ones beginning with “How many”, “What time”, “What number” are assigned “number” type. These inferred answer types are not exhaustive but have good coverage. The intuition behind using the question and answer types as auxiliary features is that some VQA models may be better than others at predicting certain types of questions and/or answers. Making this information available at the time of classification aids the stacker in making a better decision.

Question Features

We also use a bag-of-words (BOW) representation of the question as auxiliary features. Words that occur at least five times in the validation set were included. The final sparse vector representing a question was normalized by the number of unique words in the question. In this way, we are able to embed the question into a single vector. Goyal et al. (2016) showed that attending to specific words in the question is important in VQA. Including a BOW of the words in the question as auxiliary feature equips the stacker to efficiently learn which words are important and can aid in classifying answers.

Image Features

We also used “deep visual features” of the image as additional auxiliary features. Specifically, we use the $4,096$ features from VGGNet’s (Simonyan and Zisserman) $fc7$ layer . This creates an embedding of the image in a single vector which is then used by the stacker. Using such image features enables the stacker to learn to rely on systems that are good at identifying answers for particular types of images. The individual VQA models fuse an embedding of the image along with an embedding of the question. By using the image embeddings at the meta-classifier level, the stacker learns to discriminate between the component models based on a deeper representation of the images.

5.4.2 Using Explanations

Recently, there has been some work on analyzing regions of the image that deep learning models focus on while making decisions (Goyal et al., 2016; Hendricks et al., 2016; Park et al., 2016). Their work shows that deep learning models attend to relevant parts of the image while making a decision. For VQA, the parts of images that the models focus on can be thought of as *visual* explanations for answering the question. We use these visual explanations to construct auxiliary features for SWAF. The idea behind using explanation as features is that it enables the stacker to learn to trust the agreement between systems when they also agree on the heat-map explanation by "looking" at the right region of the image when generating an answer.

Generating Explanations

We used the GradCAM algorithm (Selvaraju et al., 2017) to generate model-specific explanatory heat-maps for each IQ pair. The GradCAM approach generates a class-discriminative localization-map for a given model based on its respective predicted output class in the following way. First, the gradient of the score y^c for the predicted class c is computed before the softmax layer with respect to the feature maps A^k of a convolutional layer. Then, the gradients flowing back are global average pooled to obtain the neuron importance weights.

$$w_k^c = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{backprop gradients}} \tag{5.2}$$

The above weights capture the importance of a convolutional feature map k for the output class c. Z is the total number of pixels in the feature map. A ReLU over the weighted combination of the feature maps results in the required localization-map

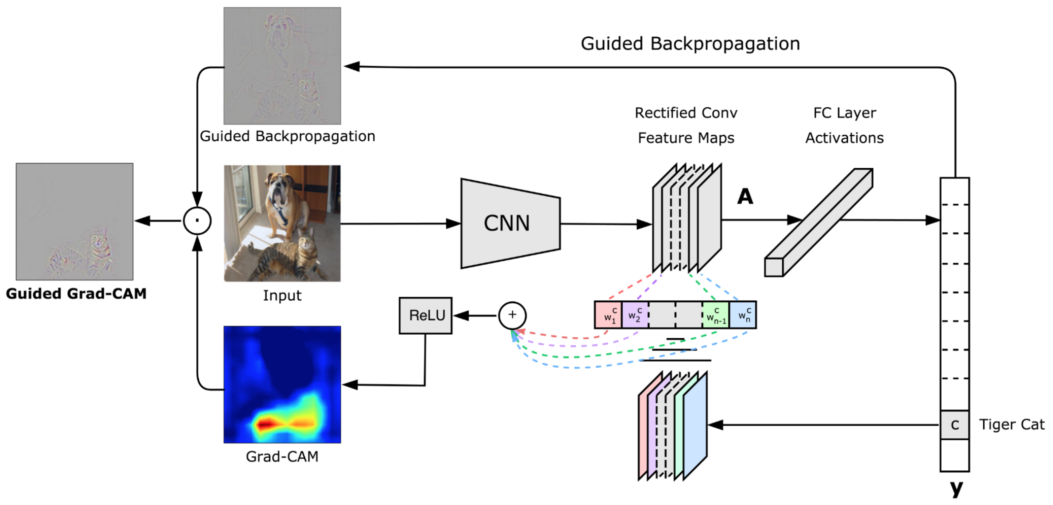


Figure 5.5: Given an image and a target class (*e.g.*"Tiger Cat"), GradCAM generates a visual explanation by backpropagating the gradients for that class to the convolutional feature maps of interest. The visual explanation represents where the model "looks" while making a particular decision.

for the output class as follows:

$$H^c = ReLU(\sum_k w_k^c A^k) \tag{5.3}$$

The heat-map H^c is of the same size as the convolutional feature maps of the network. The ReLU applied to the linear combination of maps results in features that have a positive influence on the output class, that is, the pixels whose intensity should be increased in order to increase y_c. H^c is up-sampled to the input image resolution using bi-linear interpolation. Figure 5.5 shows the end-to-end process of generating a visual explanation for a CNN.

For each of the component VQA models, we generate the localization-map to be used as auxiliary features for ensembling. The intuition behind using explanation as auxiliary features is that the agreement between systems is perhaps more trustworthy when they also agree on the heat-map explanation. Figure 5.6 shows a sample of IQ pairs from the VQA dataset and their respective heat-maps generated by three different VQA models discussed in Section 5.4.3. We propose algorithms

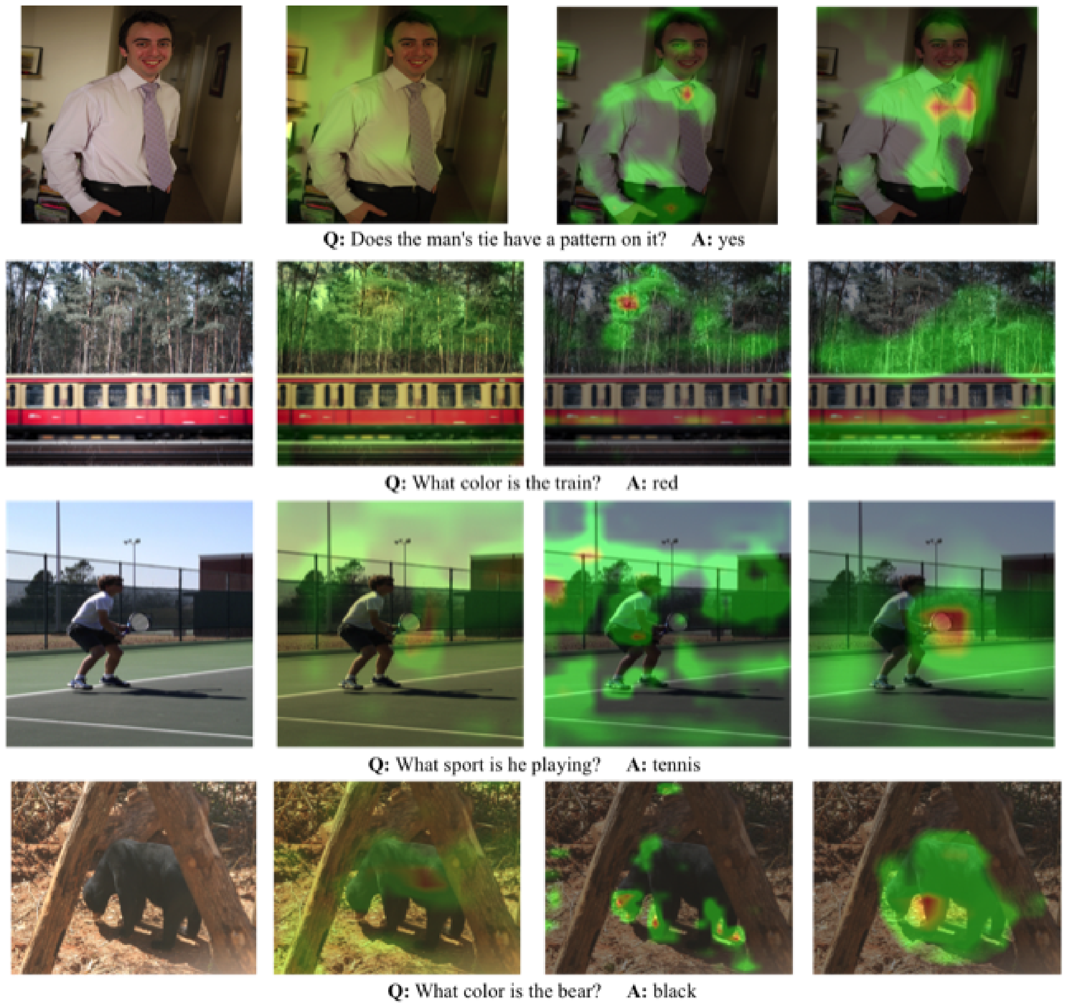

Q: Does the man's tie have a pattern on it? A: yes
Q: What color is the train? A: red
Q: What sport is he playing? A: tennis
Q: What color is the bear? A: black

Figure 5.6: Each row from left to right shows an image-question pair from the VQA dataset along with localization-maps overlaid on the image generated by the LSTM, HieCoAtt and MCB models respectively. The answers shown are those predicted by our ensemble.

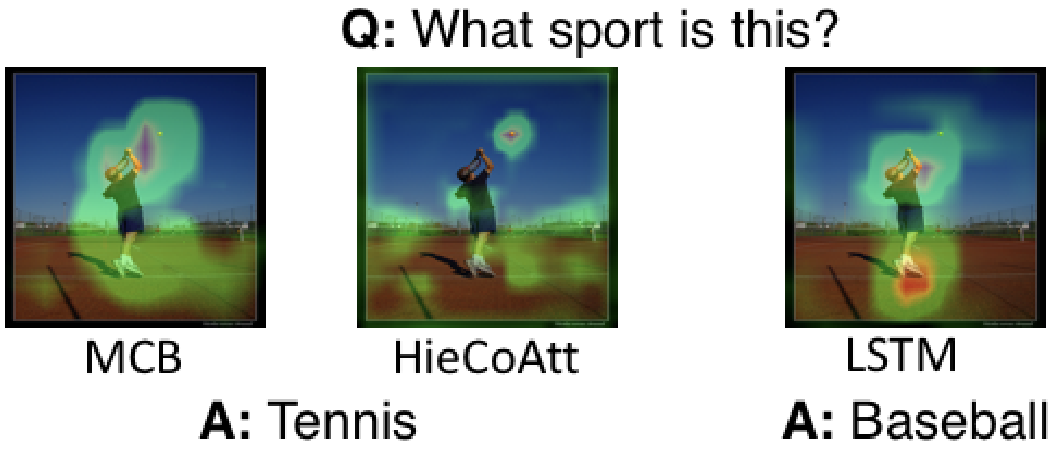


Figure 5.7: The MCB and HieCoAtt models generate the correct answer and have localization-maps that support their answer (i.e, "tennis") while the LSTM model focuses on the player's foot and ground area and produces an incorrect answer (i.e., "baseball").

for generating and evaluating *visual* explanations for ensembles in the next chapter.

Explanation as Auxiliary Features

The localization-map generated by each VQA model serves as a visual explanation for the predicted output of that model. We compare agreement between the localization-maps of the individual models to generate auxiliary features for SWAF. Figure 5.7 demonstrates that models that agree on an answer tend to have overlapping localization-maps. Figure 5.8 and Figure 5.9 show examples from VQA data where models with overlapping localization-maps produce the correct answer and those with barely any overlap between their localization-maps produce an incorrect answer. These examples indicate that systems with overlapping localization-maps produce outputs that are more reliable.

We use the rank-order correlation metric to measure the overlap between localization-maps. First, we take the absolute gray-scale value of the localization-maps in of each model and compute their mean rank-correlation with the localization-map of every other model. Then, we rank the pixels according to their spatial atten-

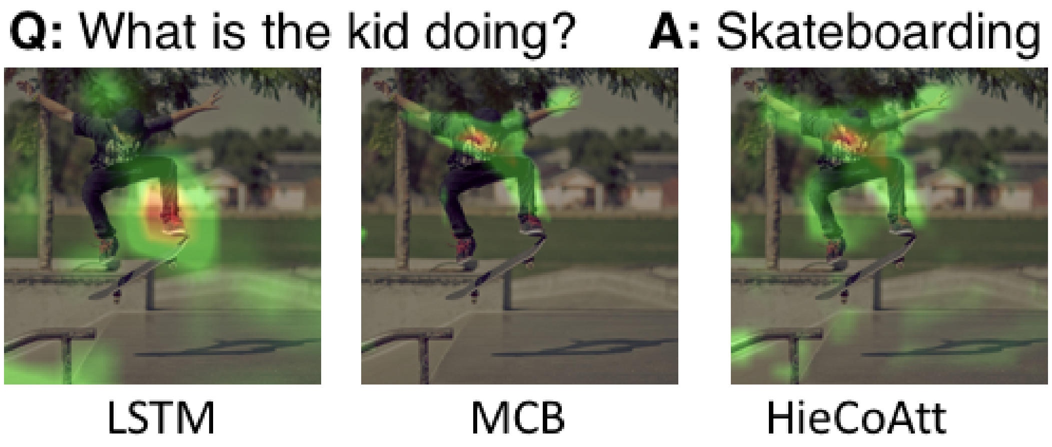


Figure 5.8: Example of an image-question pair where all three component models have overlapping localization-maps and produce the correct answer (i.e., “skateboarding”).

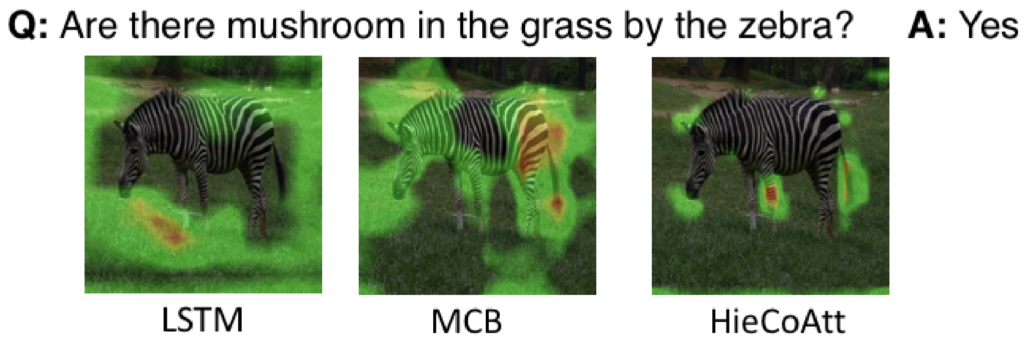


Figure 5.9: Example of an image-question pair where all three component models have barely any overlap between their localization-maps and produce the incorrect answer (i.e., “yes”).

tion and then compute the correlation between the two ranked lists. The rank correlation protocol has been used in the past to compare machine-generated and human attention-maps as described in (Das et al., 2016). We compare the localization-maps of the component VQA models pairwise and this generates $\binom{n}{2}$ "explanation agreement" features for SWAF, where n is the total number of models.

5.4.3 Component VQA models

We use SWAF to combine three diverse VQA systems such that the final ensemble performs better than any individual component model even on questions with a low agreement. All three component models are trained on just the VQA training set.

Long Short-Term Memory (LSTM)

The LSTM model (Antol et al., 2015) is one of the original baseline models used to establish a benchmark for the VQA dataset. A VGGNet (Simonyan and Zisserman) is used to obtain embeddings for the image which are combined with an LSTM (Hochreiter and Schmidhuber, 1997) embedding of each question. An LSTM with two hidden layers is used to obtain a $2,048$-dimensional embedding of the question, followed by a fully-connected layer with `tanh` non-linearity to transform the embedding to $1,024$ dimensions. The l_2 normalized activations from the last hidden layer of VGGNet are used as a $4,096$ dimensional image embedding. The image embedding is first transformed to $1,024$ dimensions by a fully-connected layer with `tanh` nonlinearity to match the dimensionality of the LSTM embedding of the question. The transformed image and LSTM embeddings are then fused via element-wise multiplication.

Hierarchical Question-Image Co-Attention (HieCoAtt)

The idea behind the HieCoAtt model is that in addition to using visual attention to focus on where to look, it is equally important to model what words to attend to in the question (question-attention) (Lu et al., 2016). This model jointly reasons about the visual and language components using “co-attention”. Question attention is modeled using a hierarchical architecture at word, phrase, and question levels.

HieCoAtt uses two types of co-attention – parallel and alternating. Parallel co-attention attends to the image and question simultaneously by calculating the similarity between image and question features at all pairs of image-locations and question-locations. Alternating co-attention sequentially alternates between generating image and question attention by attending to the image based on the question summary vector and then attending to the question based on the attended image features. Both co-attentions are executed at all three levels of the question hierarchy.

Multimodal Compact Bilinear pooling (MCB)

The MCB model combines the vision and language vector representations using an outer product instead of the traditional approach of using concatenation or element-wise product or sum of the two vectors (Fukui et al., 2016). Bilinear pooling computes the outer product between two vectors which, in contrast to the element-wise product, allows a multiplicative interaction between all elements of both vectors. To overcome the challenge of high dimensionality due to the outer product, the authors adopt the idea of using Multimodal Compact Bilinear pooling (MCB) (Gao et al., 2016) to efficiently and expressively combine multimodal features.

The MCB model extracts representations for the image using the 152-layer Residual Network (He et al., 2016) and an LSTM (Hochreiter and Schmidhuber, 1997) embedding of the question. The two vectors are pooled using MCB and the answer is obtained by treating the problem as a multi-class classification problem with $3,000$ possible classes. The MCB model won the VQA 2016 challenge by obtaining the best performance on the test set.

5.4.4 Experimental Results

We now present experimental results on the VQA challenge using the SWAF approach and compare it to various baselines, individual and ensemble VQA models, as well as ablations of our SWAF algorithm on the open-ended VQA test set. In addition to the three data splits given in Table 5.2, the VQA challenge divides the test set into *test-dev* and *test-standard*. Evaluation on either split requires submitting the output to the competition's online server.[1] However, there are fewer restrictions on the number of submissions that can be made to the *test-dev* compared to the *test-standard*. The *test-dev* is a subset of the standard test set consisting of randomly selected $60,864$ (25%) questions. We use the *test-dev* set to tune the parameters of the meta-classifier. All the individual VQA models that we ensemble are trained only on the VQA training set and the SWAF meta-classifier is trained on the VQA validation set.

For the meta-classifier, we used a $L1$-regularized SVM classifier for generic stacking and stacking with only question/answer types as auxiliary features. For the question, image, and explanation features, we found that a neural network with two hidden layers works best. The first hidden layer is fully connected and the second has approximately half the number of neurons as the first layer. The question and image features are high-dimensional and therefore a neural network classifier worked well. We found that using late fusion (Karpathy et al., 2014) to combine the auxiliary features for the neural network classifier worked slightly better. We used Keras with Tensorflow back-end (Chollet, 2015) for implementing the network. We compare our approach to a voting baseline which maximizes precision by only accepting an answer as correct if all the component systems predicted the exact same answer for a given question. For questions that do not have a consensus, the answer that has the maximum agreement is taken with ties being broken in the favor of systems with higher confidence scores. We also compare against other state-of-the-art VQA systems not used in our ensemble: iBowIMG (Zhou et al., 2015b), DPPNet (Noh et al., 2016) and the Neural Module Networks (NMNs) (Andreas et

[1] `www.visualqa.org/challenge.html`

Method	**All**	**Yes/No**	**Number**	**Other**
DPPNet (Noh et al., 2016)	57.36	80.28	36.92	42.24
iBOWIMG (Zhou et al., 2015b)	55.72	76.55	35.03	42.62
NMNs (Andreas et al., 2016b)	58.70	81.20	37.70	44.00
LSTM (Antol et al., 2015)	58.20	80.60	36.50	43.70
HieCoAtt (Lu et al., 2016)	61.80	79.70	38.70	51.70
MCB (Single system) (Fukui et al., 2016)	62.56	80.68	35.59	52.93
MCB (Ensemble) (Fukui et al., 2016)	66.50	**83.20**	**39.50**	58.00
Voting (MCB + HieCoAtt + LSTM)	60.31	80.22	34.92	48.83
Stacking	63.12	81.61	36.07	53.77
+ Q/A type features	65.25	82.01	36.50	57.15
+ Question features	65.50	82.26	38.21	57.35
+ Image features	65.54	82.28	38.63	57.32
+ Explanation features	**67.26**	82.62	**39.50**	**58.34**

Table 5.3: Accuracy results on the VQA *test-standard* set. The first block shows performance of a VQA model that use external data for pre-training, the second block shows single system VQA models, the third block shows an ensemble VQA model that also uses external data for pre-training, and the fourth block shows ensemble VQA models.

al., 2016b).

The iBowIMG concatenates the image features with the bag-of-word question embedding and feeds them into a softmax classifier to predict the answer, resulting in performance comparable to other models that use deep or recursive neural networks. The iBowIMG beats most VQA models considered in their paper. The DPPNet, on the other hand, learns a CNN with some parameters predicted from a separate parameter prediction network. Their parameter prediction network uses a Gated Recurrent Unit (GRU) to generate a question representation and maps the predicted weights to a CNN via hashing. The DPPNet uses external data (data-augmentation) in addition to the VQA dataset to pre-train the GRU. Another well-known VQA model is the Neural Module Network (NMN) that generates a neural network on the fly for each individual image and question. This is done by choosing from various sub-modules based on the question and composing these to generate the neural network, *e.g.*, the `find[x]` module outputs an attention map for detecting `x`. To arrange the modules, the question is first parsed into a symbolic expression and using these expressions, modules are composed into a sequence to answer the query. The whole system is trained end-to-end through backpropagation.

The VQA evaluation server, along with reporting accuracies on the full question set, also reports a break-down of accuracy based on three answer categories. The image-question (IQ) pairs that have answer type as "yes/no", those that have "number" as their answer type and finally those that do not belong to either of the first two categories are classified as "other". Table 5.3 shows the full and category-wise accuracies. All scores for the stacking models were obtained using the VQA *test-standard* server. The table shows results for both single system and ensemble MCB models. We used the single system MCB model as a component in our ensemble. The ensemble MCB system, however, was the top-ranked system in the VQA 2016 challenge and it is pre-trained on the Visual Genome dataset (Krishna et al., 2017) using pre-trained GloVe vectors (Pennington et al., 2014). On the other hand, our ensemble system does not use any external data and comprises of only three component models.

The SWAF approach obtained a new state-of-the-art result on the VQA task. The vanilla stacking approach itself beats the best *individual* model and adding the auxiliary features further boosts the performance. Our SWAF model with all four sets of auxiliary features related to IQ pairs did particularly well on the more difficult "other" answer category, indicating that the auxiliary features provide crucial information at classification time. Combining diverse systems with de-correlated errors intelligently is one of the reasons why SWAF shines on the "other" answer category. To further analyze the results using SWAF, we performed experiments with ablations of the auxiliary features. Figure 5.10 shows the results on the *test-dev* set obtained when ablating each of the auxiliary feature sets. We observed that deleting the Q/A type decreased performance the most and deleting the explanation features decreased performance the least.

The voting baseline does not perform very well even though it is able to

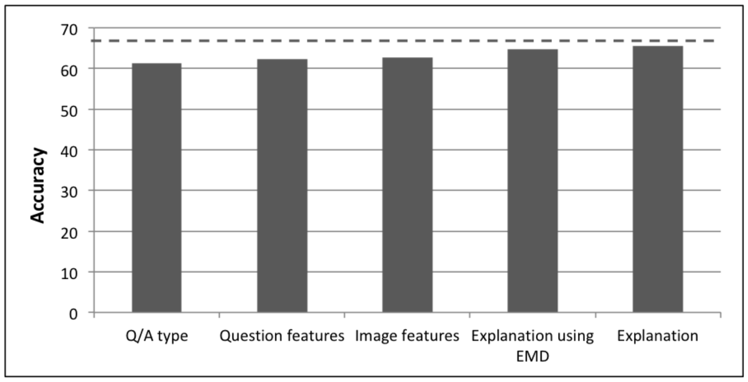


Figure 5.10: Results for auxiliary features ablations on the VQA *test-dev* set. The `x-axis` indicates the feature that was ablated from the final ensemble. Dotted line shows the performance of SWAF without any ablations.

beat one of the component models. The SWAF ablation results clearly indicate that there is an advantage in using each of the auxiliary features. How useful each of these auxiliary features is varies based on the IQ pair and answer types. Each of the auxiliary features contributed towards the final ensemble's performance, which is clear from Table 5.3. The voting and the "vanilla stacking" ensembling approaches did not perform as well as the auxiliary features. This led us to conclude that the performance gain is actually obtained from using these auxiliary features. The results reported for the MCB approach is an ensemble of *seven* MCB models with attention, as described in (Fukui et al., 2016).

Using explanations generated by various deep learning models as an auxiliary feature gave us insights into their decision-making process. We observed that the localization-maps generated were fairly noisy and these maps are still not good enough to develop human trust in those systems which is evident from Figure 5.6. Although the individual component systems agreed on an answer for many of the IQ pairs, the regions of an image they attend to varied significantly. However, the rank correlation metric made the localization-maps comparable and further using those as auxiliary features alleviated the noise. This is because the stacker *learns* a weighing of the auxiliary features including those obtained by using explanation maps when we train on the validation set. In this way, it learns to trust only the localization-maps that are actually useful. We also observed that there was a high positive correlation between the explanation maps generated by the HieCoAtt and MCB models, followed by the LSTM and MCB models and then the LSTM and HieCoAtt models with several of the maps even negatively correlated. We also experimented with using the Earth Mover's Distance (EMD) for comparing heat-maps and found that it worked better than the rank-order correlation, however, it came at a cost of high computational complexity ($\mathcal{O}(n^3)$ vs. $\mathcal{O}(n)$). Figure 5.10 shows the difference in performance obtained when the explanation feature calculated using the EMD metric and the rank-order correlation metric is ablated from the final ensemble. Clearly, using EMD for comparing explanation maps is better and shows improvement in performance. As shown in the past, our findings also confirmed that EMD metric provides a finer-grained comparison between the attention maps (Bylinskii et al., 2018). Our work shows that explanation generated by deep learning models does not have to be restricted to developing human trust or making these models more transparent but explanation can also be used effectively for improving performance on a challenging task.

5.5 Chapter Summary

We have presented results for using stacking with auxiliary features (SWAF) on two challenging vision problems – object detection and VQA. SWAF obtained very promising results on both tasks, significantly outperforming the best component systems as well as other ensembling methods. Using SWAF, we obtained an overall mAP of 50.6% on the ImageNet object detection problem and an overall accuracy of 67.26% on the VQA problem. On analyzing the results obtained by SWAF, we found that it does better when the component outputs differ widely and have low confidences. The gain in performance from SWAF comes from output decisions that are difficult to make without context; however, using auxiliary features enabled fusion of additional relevant information, allowing the stacker to make the right decision.

We proposed two and four different categories of auxiliary features for the object detection and VQA problems respectively. For the VQA task, we proposed and evaluated the novel idea of using explanations to improve ensembling of multiple systems. We demonstrated how visual explanations for VQA (represented as localization-maps) can be used to aid stacking with auxiliary features. This approach effectively utilized information on the degree to which systems agree on the *explanation* of their answers. We showed that the combination of all of these categories of auxiliary features, including explanation, gives the best results. Our work demonstrated that explanations along with developing human trust can also be used for improving performance on a challenging problem.

Chapter 6
Ensembling and Evaluating Explanations

This chapter discusses techniques for generating visual explanations for ensemble models and shows two ways to evaluate explanation that does not depend on human annotated ground truth. The work in this chapter has been published in (Rajani and Mooney, 2017a), and all work in this chapter constitutes original contributions.

6.1 Chapter Overview

Many machine learning systems deployed for real-world applications such as recommender systems, image captioning, object detection, etc. are ensembles of multiple models. Also, the top-ranked systems in many data-mining and computer vision competitions use ensembles. Although ensembles are popular, they are opaque and hard to interpret, and there has been little work on generating explanations for ensembles. In this chapter, we propose two novel methods for ensembling visual explanations – the weighted average (WA) and the penalized weighted average (PWA). We demonstrate the success of these approaches on Visual Question Answering (VQA) by using the localization maps for the component systems. Our novel approach is scalable with the number of component models in the ensemble.

We also introduce two new approaches to evaluate explanations – the comparison metric and the uncovering metric. Our crowd-sourced human evaluation indicates that our ensemble visual explanation significantly qualitatively outperforms each of the individual system's visual explanation. We consider the same three VQA systems in this chapter as in the previous chapter – the LSTM (Antol et al., 2015), the HieCoAtt (Lu et al., 2016) and the MCB (Fukui et al., 2016).

6.2 Prior Work

Prior work related to Visual Question Answering (VQA) has been discussed in the previous chapter. In this chapter, we discuss prior work related to generating and evaluating explanations and focus primarily on visual explanations.

Deep learning models have been used widely on several vision and language problems. However, they are opaque and unable to explain their decisions (Selvaraju et al., 2017). There are several advantages of having AI systems that can generate explanations that support their predictions, both when AI systems perform better and worse than humans (Johns et al., 2015; Agrawal et al., 2016). These advantages have motivated recent work on explainable AI systems, particularly in computer vision (Antol et al., 2015; Goyal et al., 2016; Hendricks et al., 2016; Park et al., 2016). However, there has been no prior work on generating visual explanations for ensemble systems.

Evaluating explanations generated by AI models is another challenging problem and crucial for measuring the quality of explainable AI systems. Most of the prior work in this area relies on annotated ground truth explanations (Park et al., 2016; Goyal et al., 2016; Das et al., 2017a). Hendricks et al. (2016) used human experts on bird watching to evaluate explanations for fine-grained bird classification and asked them to rank the image-explanation pairs. On the other hand, Das et al. (2017a) collect human attention maps for VQA by instructing human subjects on Mechanical Turk (MTurk) to sharpen parts of a blurred image that are important for answering the questions accurately. Selvaraju et al. (2017) evaluated explanations for image captioning by instructing human subjects on MTurk to select if a machine generated explanation is reasonable or not based on the predicted output.

6.3 Ensembling Visual Explanation

Our goal is to generate visual explanations for an ensemble model of VQA. We do this by ensembling explanations of the component models and using heuristics to constrain the ensemble explanation such that it is faithful to and supports the ensemble's prediction. Our strategy depends on the individual component models' answer and visualization for a given Image-Question (IQ) pair. We first build an ensemble model that uses the approach discussed in the previous chapter, Stacking With Auxiliary Features (SWAF) (Rajani and Mooney, 2017b) to combine outputs of the three component systems. We then generate an explanation for the ensemble by combining the visual explanations of the component systems.

We first generate model-specific explanation maps for each IQ pair using the Grad-CAM approach discussed in the previous chapter. These explanation maps have high and low-intensity regions depending on the parts of the image the model is attending to while deciding for the IQ pair under consideration. We generate such explanation-maps for each of the component VQA models used in the ensemble. Thereafter, we ensemble these explanation-maps of individual models to create an explanation for the ensemble. We now discuss two of our approaches for generating an ensembled visual-explanation that reflects the behavior of the ensemble – the weighted average (WA) and penalized weighted average (PWA).

6.3.1 Weighted Average Ensemble Explanation

As the name suggests, the ensemble explanation is generated by averaging the explanations of the component models proportional to their weights. The Weighted Average (WA) ensemble explanation is calculated as follows:

$$E_{i,j} = \begin{cases} \frac{1}{|K|}\sum_{k\in K} w_k A^k_{i,j}, & \text{if } w_k A^k_{i,j} \geq t \\ 0, & \text{otherwise} \end{cases} \qquad \text{subject to } \sum_{k\in K} w_k = 1 \tag{6.1}$$

Here, E is the explanation map of the ensemble, i and j are used to index into the explanation map entries, K is the set of component systems, w^k and A^k are the weights and explanation maps respectively for each of the component systems, and t is a thresholding value.

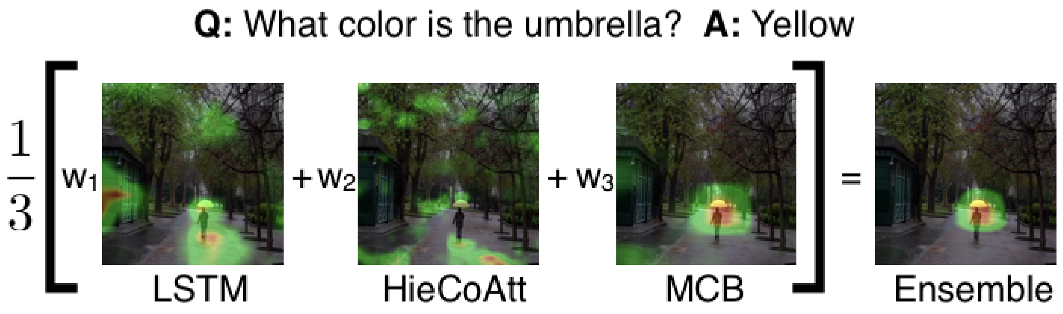


Figure 6.1: The Weighted Average (WA) ensemble explanation approach on an instance of the VQA data. In this example, all three component systems as well as the ensemble agree on the same answer (i.e., "yellow").

Thresholding the pixel values for the maps before or after averaging worked well for reducing noise as well as eliminating several low-intensity regions that arose as a result of combining multiple noisy maps. A weighted combination of the component feature maps worked better than using equal weights across all component systems. We weight the maps of the component systems proportional to their performance on the validation set, subject to the constraint that the weights sum to one. Figure 6.1 demonstrates the weighted average approach on an image-question pair.

The weighted average ensemble explanation only combines maps of individual systems that agree with the ensemble on the answer. If some component systems do not agree with the ensemble, this approach ignores them. However, information from the explanation maps of such disagreeing systems can be used to adjust the ensemble explanation, as in the following approach.

6.3.2 Penalized Weighted Average Ensemble Explanation

Component VQA systems that agree with the ensemble on the answer for an IQ pair have relevant explanation maps that reflect how the model arrived at its prediction. On the other hand, component systems that do not agree with the ensemble's output answer have explanation maps that are potentially irrelevant to the ensemble's answer and can be discounted from the ensemble's explanation. The

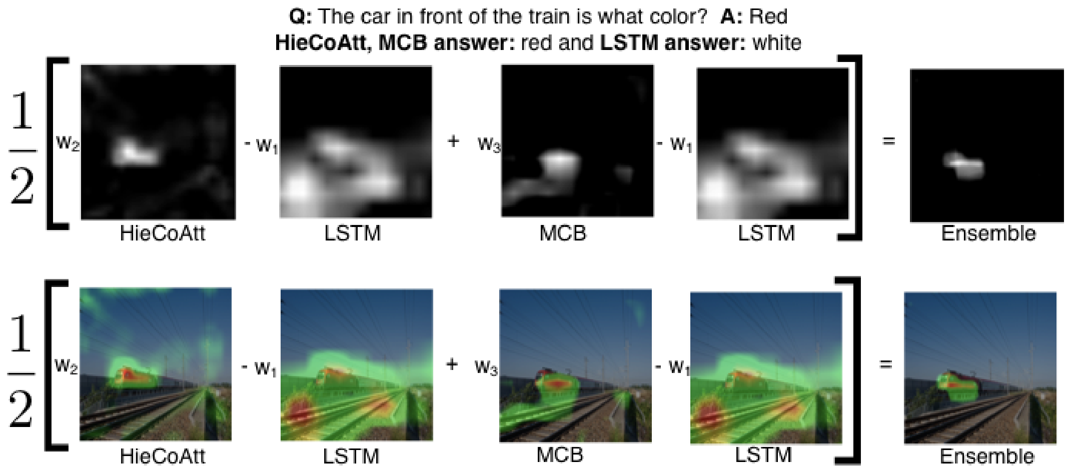

Q: The car in front of the train is what color? A: Red
HieCoAtt, MCB answer: red and LSTM answer: white
1/2
w2
- w1
+ w3
- w1
=
HieCoAtt
LSTM
MCB
LSTM
Ensemble

Figure 6.2: The top row shows the process of ensembling visual explanation for an IQ pair when the ensemble model agrees with the MCB and HieCoAtt models (ans: "red") and disagrees with the LSTM model (ans: "white"). The bottom row shows the reference IQ pair and the MCB vs ensemble visual explanation. The explanation map is normalized to obtain the final ensemble visualization.

Penalized Weighted Average (PWA) ensemble explanation is calculated as follows:

$$E_{i,j} = \begin{cases} \frac{1}{|K|} \sum_{k \in K} \sum_{m \in M} \overbrace{w_k A^k_{i,j} - w_m I^m_{i,j}}^{\text{p}}, & \text{if } p \geq t \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$
$$\text{subject to } \sum_{k \in K} w_k + \sum_{m \in M} w_m = 1$$

Here, I^m is an explanation map of a component system that does not agree with the ensemble and M is the total number of such systems. This assumes that a system that does not agree with the ensemble's answer is highlighting regions of the image that are *not* relevant, so we down-weight those regions in the explanation map for the ensemble. Another variation we explored is forcing a component model that does not agree with the ensemble to produce an explanation map for the alternate answer picked by the ensemble. We then calculate the ensemble explanation as in

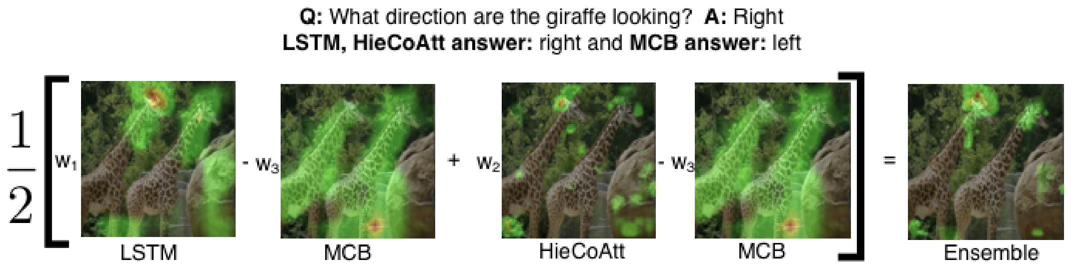

Q: What direction are the giraffe looking? A: Right
LSTM, HieCoAtt answer: right and MCB answer: left
1/2
W1
- W3
+ W2
- W3
=
LSTM
MCB
HieCoAtt
MCB
Ensemble

Figure 6.3: The Penalized Weighted Average (PWA) ensemble explanation approach on an instance of the VQA data. In this case, the LSTM and HieCoAtt systems agree with the ensemble's answer (i.e., "right") while the MCB disagrees.

the previous section, where all systems agree on the output. The forced version of WA performed consistently worse than the corresponding PWA approach. We, therefore, do not include it in our discussion of the results in Section 6.5. The in-depth analysis of our results in Section 6.6, however, includes the results for the forced version of WA.

Figure 6.2 and Figure 6.3 demonstrate the process of ensembling visual explanations using the PWA approach on instances from the VQA dataset. We use crowd-sourced hyper-parameter tuning to set the threshold value for each of the VQA systems and is discussed in detail later in this chapter in Section 6.3.4.

6.3.3 Agreement with N systems

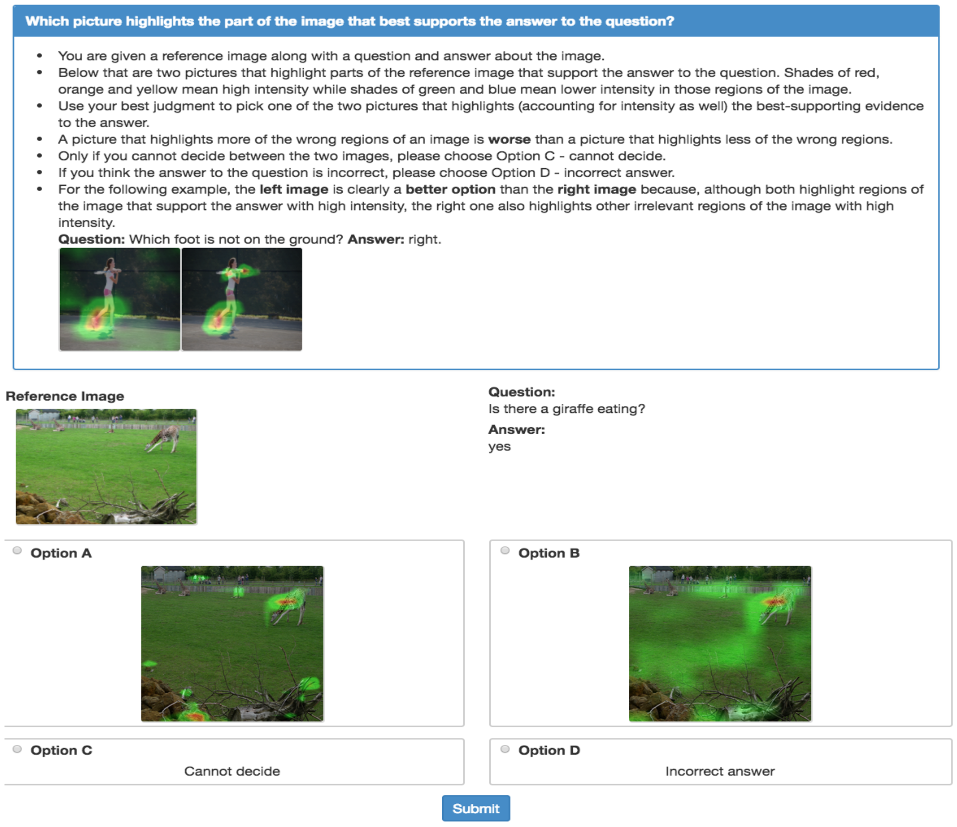
A visual explanation ensemble can be generated for N component models using Equations 6.1 and 6.2 and it scales with N. We consider three component VQA systems and there are three scenarios that arise depending on whether the ensemble model agrees with all three, any two, or only one of the component systems. For all the scenarios, we first generate a gray-scale GradCam visualization for each of the component systems. Thereafter, we generate the ensemble explanation using the aforementioned approaches depending on the scenario under consideration.

We observed that our ensemble model agreed with all *three* systems on approximately half of the VQA test set. In this case, we use the WA approach described in Section 6.3.1 to generate the ensemble explanation map. The MCB component model (Fukui et al., 2016), which uses the 152-layer ResNet network had the highest weight followed by the HieCoAtt (Lu et al., 2016) and the LSTM (Antol et al., 2015) models, that use VGGNet. For approximately one-fourth of the test set, our ensemble model agreed with exactly *two* component systems. For this scenario, we combine the explanation maps using both WA and PWA approaches by, respectively, ignoring or down-weighting the system that does not agree with the ensemble's answer. When the ensemble model agreed with only *one* component system's output, we generated the ensemble explanation map in two ways. First, the ensemble explanation was set equal to the explanation of the system it agrees with, minus the explanation of the systems it does not agree with, as in Equation 6.2. Second, we force the systems that do not agree with the ensemble to produce explanation maps for the answer produced by the ensemble and then use those maps to calculate the ensemble explanation map using Equation 6.1.

6.3.4 Crowd-sourced hyper-parameter tuning

The weighted average and the penalized weighted average methods for ensembling explanation maps depend on the parameter t which thresholds the pixel values for the maps. We also use the threshold parameter for getting rid of the noise in the explanation maps of individual systems. We use crowd-sourcing to determine the value of t. The idea is to optimize the explanation map generation based on the evaluation metric which as discussed above uses human judgment. Very recently, crowd-sourcing was used to tune the hyper-parameters of a policy network for a reinforcement learning agent (Fridman et al., 2018).

We use Mechanical Turk to search for a good value of the parameter t for each of the individual as well as the ensemble systems. We chose 50 random instances from the VQA validation set for judging the value of the threshold parameter. The Turkers had to select the image that highlighted the right amount of the appropriate regions to answer the given question. They were shown images with explanation maps thresholded in steps of $0.1, 0.15, 0.2, 0.25$. The human judges also had an option that the highlighted regions is not appropriate. We found that a threshold of less than 0.1 or more than 0.25 generated maps that were too noisy or not sufficiently highlighted respectively. The outcome of our experiment on crowd-sourcing the threshold parameter was that a threshold of 0.2 worked well for all the ensemble and individual systems except the HieCoAtt. The optimal threshold for HieCoAtt was 0.15. The pixel intensities $> t$ are normalized to lie between zero and one. We used the threshold parameters obtained using crowd-sourcing for all our evaluations. Our results improved by searching the right threshold values for each of the systems when compared to using a uniform threshold for all systems.


Which picture highlights the part of the image that best supports the answer to the question?

- You are given a reference image along with a question and answer about the image.
- Below that are two pictures that highlight parts of the reference image that support the answer to the question. Shades of red, orange and yellow mean high intensity while shades of green and blue mean lower intensity in those regions of the image.
- Use your best judgment to pick one of the two pictures that highlights (accounting for intensity as well) the best-supporting evidence to the answer.
- A picture that highlights more of the wrong regions of an image is **worse** than a picture that highlights less of the wrong regions.
- Only if you cannot decide between the two images, please choose Option C - cannot decide.
- If you think the answer to the question is incorrect, please choose Option D - incorrect answer.
- For the following example, the **left image** is clearly a **better option** than the **right image** because, although both highlight regions of the image that support the answer with high intensity, the right one also highlights other irrelevant regions of the image with high intensity.
  Question: Which foot is not on the ground? **Answer:** right.

Reference Image

Question:
Is there a giraffe eating?
Answer:
yes

Figure 6.4: AMT interface for evaluating visual explanations.

6.4 Evaluating Visual Explanations

A good explanation evaluation metric tests how well an explanation supports the decision made by the system. In light of this, we propose two new crowd-sourced human evaluation metrics and use them to assess our ensemble explanation. The first metric asks human judges to compare two machine-generated explanations and is called the *comparison* metric, while the second metric determines if the input evidence highlighted by an explanation is sufficient to allow a human judge to independently arrive at the same prediction and is called the *uncovering* metric.

6.4.1 Comparison metric

For the comparison metric, we showed two visual explanations side-by-side to workers on Amazon Mechanical Turk (AMT) along with the image-question (IQ) pair as well as the ensemble model's answer and ask them "Which picture highlights the part of the image that best supports the answer to the question?". One image is the explanation map of the ensemble while the other is the explanation map of an individual system. We provide detailed instructions along with an example to show what a good visualization looks like. Apart from picking one of the two images as more interpretable, we also give two more options – "cannot decide" and "wrong answer" (for when the judge believes the given answer is incorrect).

Since both visual explanations are machine generated, there is no question of judging explanations based on their similarity to human ones. The evaluation simply compares explanations based on whether they highlight regions of the image that support the answer. We note that the ensemble explanation is compared to an individual system's explanation map only if that system produced the same output answer as the ensemble. When multiple individual systems agree with the ensemble on an output for an IQ pair, then the ensemble explanation is compared to one of the individual system's explanation chosen randomly with equal probability.

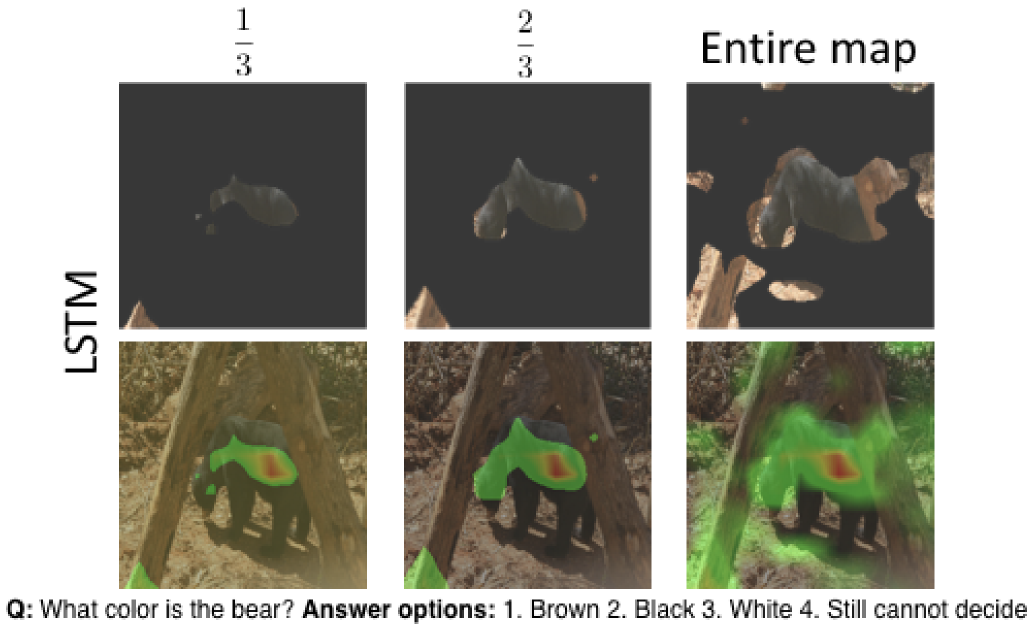

$\frac{1}{3}$
$\frac{2}{3}$
Entire map
LSTM
Q: What color is the bear? **Answer options:** 1. Brown 2. Black 3. White 4. Still cannot decide

Figure 6.5: The top row shows the uncovering of the explanation map step-by-step from left-to-right on an instance of the VQA data generated by the LSTM model. The second row shows the corresponding reference heat-maps beginning from one-third of the "hottest" region of the explanation going all the way to uncovering the entire explanation map.

6.4.2 Uncovering metric

The uncovering metric tests whether the input evidence highlighted by a visual explanation is sufficient to allow a human judge to independently arrive at the same prediction as the model that produced it. A judge is shown the question and a partially uncovered image and asked to pick an answer from a multiple choice list (taken from the multiple-choice version of the standard VQA evaluation (Antol et al., 2015)). A fraction of the most heavily weighted pixels in the visual explanation are "uncovered" and the judge is asked to pick the answer, or choose "cannot decide" when the partially uncovered image does not support an answer.

First, only the top one-third most intense pixels are uncovered, followed by the top two-thirds and finally, the entire explanation map is uncovered. The regions of the image that are not part of the explanation (zero-weighted pixels) are never

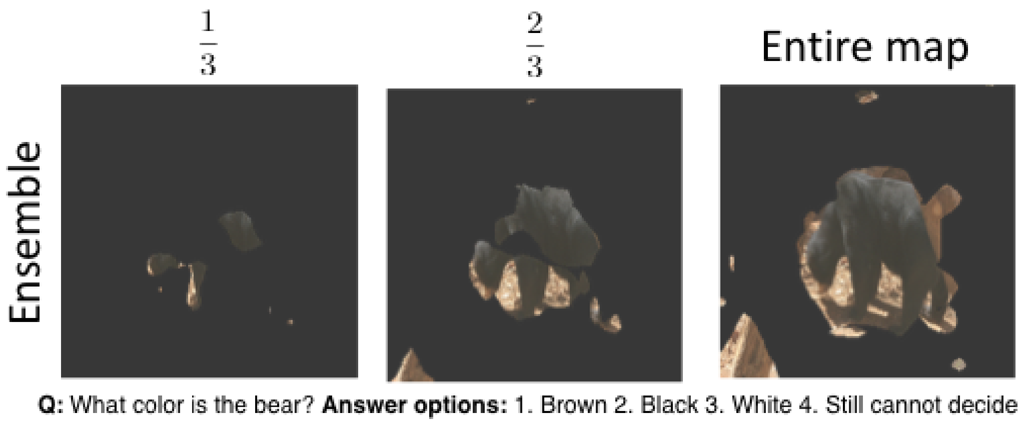

$\frac{1}{3}$
$\frac{2}{3}$
Entire map
Ensemble
Q: What color is the bear? **Answer options:** 1. Brown 2. Black 3. White 4. Still cannot decide

Figure 6.6: Example of the uncovering metric on the ensemble explanation.

exposed. The three images appear one after the other and turkers have to select the first partially uncovered image that is sufficient to pick the answer from the available choices. The Turkers were asked to complete two parts for each instance and given the following instruction, ”Part 1: Select an answer based on the question and set of partially visible images; Part 2: Select the first image from the set that was sufficient to arrive at the answer.” In this way, we evaluate both the ensemble's explanation as well those of the individual component systems and compare the percentage of the explanation map that was uncovered versus the accuracy of the answers selected by the human subjects. An explanation is better to the extent that it allows humans to pick the correct answer from a partially uncovered image showing just the most highly-weighted evidence used by the system. Figure 6.5 and Figure 6.6 show the step-by-step uncovering metric on the explanation maps for the LSTM and ensemble models respectively. As evident, the ensemble explanation maps are more precise than those of the LSTM model's explanation maps even though it highlights a bigger region of the image. The second image in Figure 6.6 is sufficient to arrive at the answer but even the third image in Figure 6.5 is barely enough to decide on an answer, since it is not clear if that object is actually a bear.

A drawback of the aforementioned approach is that if a system tends to highlight a larger proportion of the overall image, then it would have an undue advantage over other systems since it would tend to cause a larger fraction of the overall image to be revealed. This is because the evaluation metric discussed above is based on uncovering some fraction of the *non-zero-weighted region* of the image. To overcome this drawback, we also measured an alternate *normalized* version of the uncovering metric that revealed a fraction of the *entire image* as opposed to just the non-zero portion highlighted in the explanation map. So, at each step, we showed Turkers images that uncovered one-fourth, one-half, and three-fourths of the highest-weighted pixels in the entire image. In order maintain the ratio, we frequently had to uncover a number of zero-intensity pixels. The zero intensity pixels uncovered are randomly chosen from the entire image, giving rise to a "snow like" effect as shown in Figure 6.7. Arguably, this approach gives a more fair comparison between explanation maps of various systems. Using the normalized pixel ratio, sometimes the entire explanation map is revealed in the very first step of revealing one-fourth of the entire image, as evident from Figure 6.7, while other times the explanation map covers more than one-fourth of all the pixels and therefore is completely revealed in further steps of the normalized uncovering metric, as shown in Figure 6.8.

6.5 Experimental Results and Discussion

We evaluated our visual explanation maps for the VQA ensemble using the aforementioned comparison and uncovering metrics. The image-question (IQ) pairs used for generating and evaluating explanations were taken from the test-set of the VQA challenge. Three workers evaluated each of 200 random test IQ pairs for each of the different explanation ensembling methods discussed in Section 6.3 for each of the metrics. We then aggregated the Turkers decisions using voting, and when there is no agreement among workers, we classified those instances under a "no agreement" category and we ignored the instances for which the majority of Turkers thought the ensemble's answer was incorrect. For the comparison and uncovering metrics, we obtained inter-annotator agreement of 88% and 79% respectively.

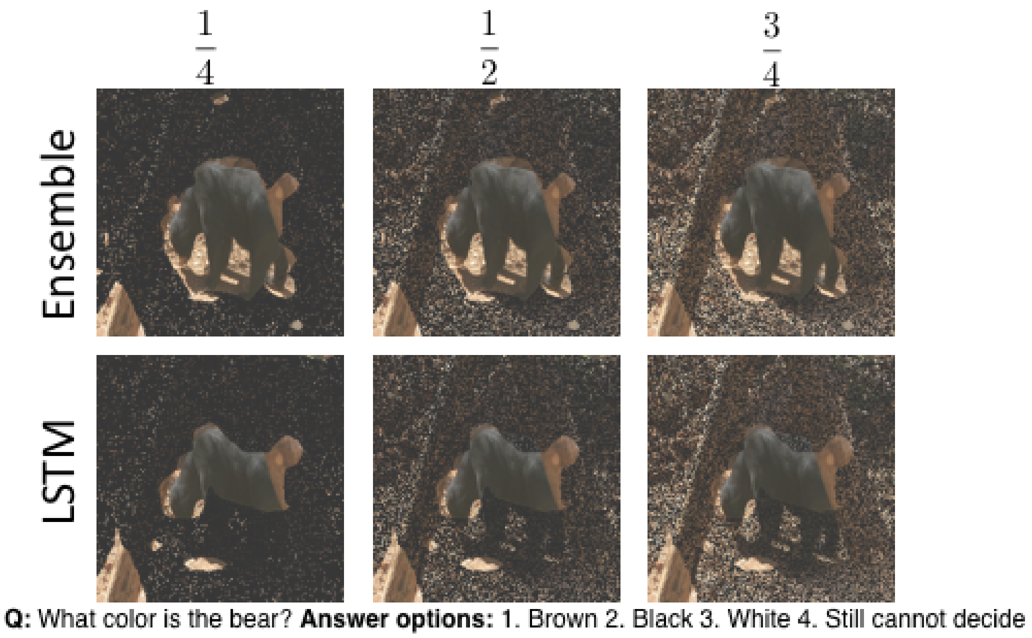

$\frac{1}{4}$
$\frac{1}{2}$
$\frac{3}{4}$
Ensemble
LSTM
Q: What color is the bear? **Answer options:** 1. Brown 2. Black 3. White 4. Still cannot decide

Figure 6.7: The top and bottom rows show the step-by-step uncovering metric using the normalized pixel ratio on the explanation map from left-to-right for the ensemble and LSTM models respectively.

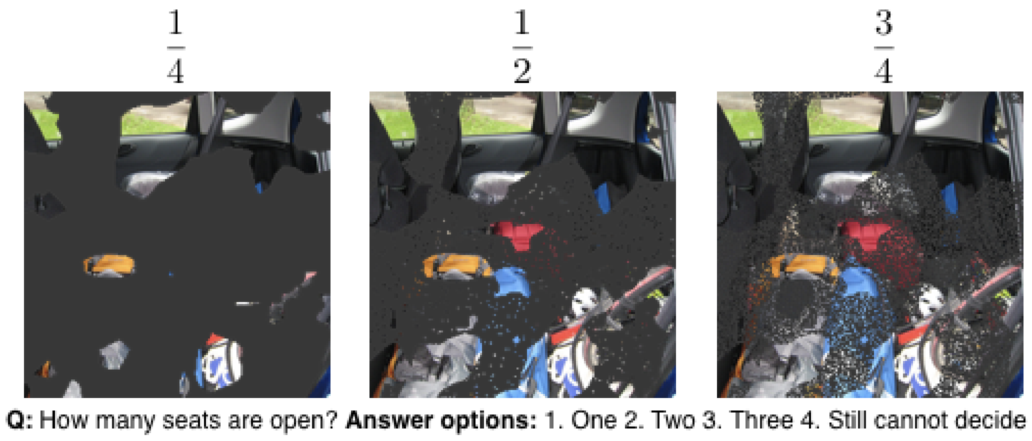

$\frac{1}{4}$
$\frac{1}{2}$
$\frac{3}{4}$
Q: How many seats are open? **Answer options:** 1. One 2. Two 3. Three 4. Still cannot decide

Figure 6.8: The top and bottom rows show the step-by-step uncovering metric using the normalized pixel ratio on the explanation map from left-to-right for the ensemble and LSTM models respectively.

Approach	Ensemble	Single System	Cannot decide
	Ensemble (WA)		
LSTM	**58**	36	3
HieCoAtt	**62**	27	6
MCB	52	41	2
	Ensemble (PWA)		
LSTM	**64**	28	3
HieCoAtt	**69**	26	1
MCB	**61**	35	1

Table 6.1: Results obtained using the comparison metric for evaluating the ensemble explanation map in terms of the percentage of cases a system's explanation was preferred, averaged for each ensembling approach. The remaining percentage of the time there was no majority agreement among human subjects. The bold figures imply statistical significance (p-value< 0.05).

6.5.1 Comparison metric

Table 6.1 shows the results obtained when comparing the ensemble explanation using the weighted average (WA) and the penalized weighted average (PWA) approaches with the individual systems' explanation. The results are averaged across instances of image-question pairs for each individual system. The rows in Table 6.1 show the percentage of time the Turkers found the single system vs. the ensemble explanation map to be qualitatively more interpretable. For a small percentage of cases, the Turkers were not able to decide if either the single system's or the ensemble's explanation was better, displayed in the third column and for the remaining percentage of time there was no majority agreement among the Turkers. We found that, on an average, the Turkers considered our ensemble's explanation more interpretable than an individual model's explanation 61% of the time.

We used the WA approach for generating the ensemble explanation when more than one system agreed with the ensemble's output prediction. We observed that the ensemble's explanation for an IQ pair was better than the LSTM, the HieCoAtt and the MCB models 58%, 62% and 52% of the time respectively. We performed a pairwise t-test with a significance level of 0.05 and found that the ensemble explanation using the WA approach was significantly better (p-value< 0.05) than the LSTM and the HieCoAtt systems' explanation.

When there was at least one individual system that did not agree with the ensemble on an output, we used the PWA approach for generating the explanation map. We observed that the ensemble's explanation for an IQ pair was better than the LSTM, the HieCoAtt and the MCB models 64%, 69% and 61% of the time respectively. We found that the ensemble explanation using the PWA approach was significantly better (p-value< 0.05) than all three individual systems' explanations on a pairwise t-test with significance level 0.05. We note that there were scenarios, like when two systems agreed with the ensemble when we compared both WA and PWA ensemble explanation maps to an individual system's explanation map. In such scenarios, we observed that PWA performed better than WA.

6.5.2 Uncovering metric

After each step of uncovering either $1/3$, $2/3$, or all of the explanation map for an image, we measured for what percentage of the test cases a human judge both decided they were able to answer the question and picked the correct answer. We found that, on an average, the penalized weighted average (PWA) ensemble explanation was sufficient 69% of the time and the weighted average (WA) ensemble explanation was sufficient 64% of the time to arrive at the correct answer from a set of answers for a given image-question pair. For the same IQ pairs and answer choices, the LSTM, the HieCoAtt, and the MCB models had explanation maps that were sufficient to arrive at the right answer only 42%, 38% and 46% of the time respectively. Table 6.2 shows the breakup of these percentages across the three partially uncovered images ranging from the least visible to entirely uncovered explanation maps for the ensemble and each of the individual models.

We observed that the Turkers were unable to decide on an answer based on just the PWA and WA ensemble explanation and required the entire image for about 14% and 17% of the questions respectively. However, for the individual models, the same fraction was 43% for the LSTM, 51% for the HieCoAtt and 42% for the MCB

System	One-third	Two-thirds	Entire map
Ensemble (PWA)	**29**	**35**	**69**
Ensemble (WA)	17	28	64
LSTM	10	22	42
HieCoAtt	9	19	38
MCB	11	20	46

Table 6.2: Results obtained using the uncovering metric averaged over image-question pairs. Shows the percentage of cases for which a partially revealed image was sufficient to arrive at the correct answer. Boldface indicates maximum in each column.

models. There was no agreement among the Turkers for the remaining percentage of cases. The ensemble explanation generated using PWA was significantly better (p-value< 0.05) than all three individual systems' explanation on a pairwise t-test with significance level 0.05. On the other hand, the ensemble explanation generated using WA exhibited trends towards statistical significance with all three individual systems' explanation. Also, the ensemble explanations generated by neither WA nor PWA were significantly better than the other.

System	One-fourth	One-half	Three-fourths
Ensemble (PWA)	**23**	**38**	**76**
Ensemble (WA)	21	34	71
LSTM	10	24	65
HieCoAtt	10	23	57
MCB	12	25	64

Table 6.3: Results obtained using the uncovering metric averaged over image-question pairs. Shows the percentage of cases for which a partially revealed image was sufficient to arrive at the correct answer based on the normalized uncovered pixel ratio. Boldface indicates maximum in each column.

We also experimented with uncovering a fraction of the *entire image* and not just the explanation map. The human judges were shown images that had $1/4$, $1/2$ and $3/4$ of the uncovered regions as shown in Figure 6.7. If the Turkers were still unable to decide on an answer even after revealing $3/4$ of the entire image then they chose the option “still cannot decide”. Table 6.3 shows the results obtained when uncovering with respect to the entire image and not just the explanation map. We found that, on an average, the PWA and the WA ensemble explanation were sufficient 76% and 71% of the time to arrive at the correct answer for a given image-question pair. For the same image-question pairs and answer choices, the LSTM, the HieCoAtt, and the MCB models had explanation maps that were sufficient to arrive at the right answer only 65%, 57% and 64% of the time respectively. The remaining percentage of time the Turkers were unable to decide on an answer even after looking at three-fourth of the entire image. We observed that the difference between the performance of the ensemble and individual systems when uncovering with respect to the entire image is not as pronounced as uncovering with respect to the explanation map; however, the overall trends in the results are very similar.

We found that the ensemble explanation using the PWA approach exhibited trends towards statistical significance with all three individual system's explanation. On the other hand, the ensemble explanation using the WA approach exhibited trends towards statistical significance with the HieCoAtt system's explanation. Also, the ensemble explanations generated by neither WA or PWA were significantly better than the other. Our choice for the fractions of the uncovered image at each step was decided so that the total number of images to be judged is neither too many (if a very small fraction is revealed) nor too few (if a very big fraction is revealed). The optimal number of fractions improves the effectiveness of the uncovering metric.

6.6 In-depth Analysis

To get further insights into the results obtained using our evaluation metrics, we performed further analysis by evaluating the results based on the number of systems that agreed with the ensemble on the answer of an IQ pair. We now discuss these results in detail below for each of the two evaluation metrics proposed.

Approach	Ensemble	Single system	Cannot decide
Case 1: Any *one* system agrees with the ensemble			
WA	55	37	4
PWA	**59**	36	2
Case 2: Any *two* systems agree with the ensemble			
WA	**62**	31	5
PWA	**71**	24	2
Case 3: All *three* systems agree with the ensemble			
WA	**61**	34	2

Table 6.4: Results obtained using the comparison metric for evaluating the ensemble explanation map in terms of the percentage of the time, averaged for each case. The ensemble's explanation maps were generated using both the Weighted Average (WA) and the Penalized Weighted Average (PWA) approaches when one or two systems agree with the ensemble's output. PWA approach is not applicable when all the systems agree with the ensemble's prediction. The bold figures imply statistical significance (p-value< 0.05).

6.6.1 Comparison metric:

Table 6.4 shows the results obtained when any one system, any two systems, and all three systems agree on the output of the ensemble. For the case when any one system agrees with the ensemble, we averaged the results over IQ pairs for each system that agreed with the ensemble's output. The weighted average (WA) combines the explanation map of the system that agrees with the ensemble on the output along with the other two systems' explanation maps generated by forcing them to produce maps for the answer predicted by the ensemble. We found that subtracting the thresholded explanation maps of the systems that did not agree with the ensemble's output using the penalized weighted average (PWA) approach worked slightly better than using the weighted average approach. For the case when any two systems agree with the ensemble, we average the results over IQ pairs for every set of two systems that agreed with the ensemble's output and compare results for both WA and PWA scenarios for generating the ensemble explanation maps. For the case when all three systems agree with the ensemble, we found that using the weighted average worked better than using equal weights. The PWA approach is not applicable in this scenario since there is no system that disagrees with the ensemble's answer prediction.

For each of the three cases, the Turkers are instructed to compare the ensemble explanation to the explanation generated by a random single system that agrees with the ensemble's output. For example, for the case when two systems agree with the ensemble on an IQ pair, say LSTM and HieCoAtt, then the explanation of the ensemble is compared to the explanation of the LSTM or HieCoAtt systems with a probability of 0.5. For Table 6.4, the remaining percentage of the time, there was no majority agreement among the Turkers. We also experimented with taking the union and intersection of the component explanation maps for various scenarios but found that they were either too noisy or too minimal and thus do not report them. We performed a pairwise t-test with significance level 0.05 across several batches and found that the ensemble explanation generated using PWA approach was always significantly better (p-value< 0.05 than the single system explanation. The ensemble explanation generated using WA was significantly better (p-value< 0.05 when at least two systems agreed with the ensemble's answer.

System	**One-third**	**Two-thirds**	**Entire map**
Ensemble (WA)	**18**	**28**	**64**
LSTM	10	18	37
HieCoAtt	02	13	31
MCB	10	15	44

Table 6.5: Results obtained using the uncovering metric for the individual models and the weighted average ensemble when all three systems agree with the ensemble's answer. The figures show the percentage of cases for which a partially revealed image was sufficient to arrive at the correct answer. Boldface indicates maximum in each column.

System	One-third	Two-thirds	Entire map
Ensemble (PWA)	**24**	**31**	**69**
Ensemble (WA)	20	30	67
LSTM	10	24	46
HieCoAtt	15	25	45
MCB	11	25	48

Table 6.6: Results obtained using the uncovering metric for the individual models and the ensemble systems when any two systems agree with the ensemble's answer averaged over IQ pairs. Boldface indicates maximum in each column.

System	One-third	Two-thirds	Entire map
Ensemble (PWA)	**34**	**46**	**70**
Ensemble (forced WA)	12	25	60
LSTM	11	22	41
HieCoAtt	09	19	39
MCB	10	16	48

Table 6.7: Results obtained using the uncovering metric for the individual models and the ensemble systems when any one system agrees with the ensemble's answer, averaged over IQ pairs. Boldface indicates maximum in each column.

6.6.2 Uncovering metric:

After each step of uncovering either $1/3$, $2/3$, or all of the explanation map for an image, we measured for what percentage of the test cases a human judge both decided they were able to answer the question and picked the correct answer. We found that when all three systems agreed with the ensemble's answer prediction, the weighted average explanation ensemble map was sufficient 64% of the time to arrive at the correct answer for a question. For the same question and answer choices, the LSTM, the HieCoAtt, and the MCB models had explanation maps that were sufficient to arrive at the right answer only 37%, 31% and 44% of the time respectively. Table 6.5 shows the breakup of these percentages across the three partially uncovered images ranging from the least visible to entirely uncovered explanation maps for the ensemble and each of the individual models. We observed that the

Turkers were unable to decide on an answer based on just the ensemble explanation and required the entire image for about 17% of the questions. However, for the individual models, the same fraction was 43% for the LSTM, 51% for the HieCoAtt and 42% for the MCB models. There was no agreement among the Turkers for the remaining percentage of the cases. The penalized weighted average was not applicable in this scenario since there was no system that disagreed with the ensemble's prediction.

When any two systems agreed with the ensemble's prediction, we considered two cases – weighted average (WA) of the two systems' maps that agreed and penalized weighted average (PWA) of the two systems' map that agreed, discounting the system's map that disagreed. We averaged the results over IQ pairs for every set of two systems that agreed with the ensemble's output. Using the uncovering metric, we found that overall the ensemble explanation using the WA and PWA approaches was sufficient 67% and 69% of the time respectively, to arrive at the correct answer. The LSTM, the HieCoAtt and the MCB models' explanation maps were sufficient to answer a question correctly only about 46%, 45% and 48% of the time respectively. Table 6.6 shows the breakup of these percentages across the range of partially uncovered explanation maps. The Turkers were not able to decide on an answer based on just the ensemble explanation and required the entire image for the PWA and WA approaches for only 14% and 20% of the questions respectively. However, for the individual models, the same ratio was 39% for the LSTM, 42% for the HieCoAtt and 34% for the MCB models respectively. For the remaining percentage of cases, there was no agreement among the Turkers.

When any one system agreed with the ensemble's prediction, we considered two cases – weighted average (WA) of the system's map that agrees combined with forcibly generated explanation maps based on the ensemble's output for the other two systems and the penalized weighted average (PWA) of the agreeing system's map discounting the other two systems' maps that disagreed. We found that overall the ensemble explanation using the WA and PWA approaches was sufficient 70% and 60% of the time respectively, to arrive at the correct answer. The LSTM, the HieCoAtt and the MCB models' explanation maps were sufficient to answer a

System	One-fourth	One-half	Three-fourths
Ensemble (WA)	**19**	**30**	**71**
LSTM	10	19	59
HieCoAtt	04	16	49
MCB	13	20	51

Table 6.8: Results obtained using the uncovering metric for the individual models and the weighted average ensemble based on the normalized uncovered pixel ratio when all three systems agree with the ensemble's answer. Boldface indicates maximum in each column.

System	One-fourth	One-half	Three-fourths
Ensemble (PWA)	**26**	**49**	**78**
Ensemble (WA)	25	43	74
LSTM	13	27	69
HieCoAtt	12	26	61
MCB	15	30	67

Table 6.9: Results obtained using the uncovering metric for the individual models and ensemble systems based on the normalized uncovered pixel ratio when any two systems agree with the ensemble's answer. Boldface indicates maximum in each column.

question correctly only about 41%, 39% and 48% of the time respectively. Table 6.7 shows the breakup of these percentages across the range of partially uncovered explanation maps. The Turkers were not able to decide on an answer based on just the ensemble explanation and required the entire image for the PWA and WA approaches for only 10% and 24% of the questions respectively. However, for the individual models, the same ratio was 46% for the LSTM, 49% for the HieCoAtt and 38% for the MCB models respectively.

We also experimented with uncovering a fraction of the *entire image* and not just the explanation map. The human judges were shown images that had $1/4$, $1/2$ and $3/4$ of the uncovered regions as shown in Figure 6.7. These fractions are smaller than those used in the previous uncovering experiment because they

System	One-fourth	One-half	Three-fourths
Ensemble (PWA)	**20**	**31**	**73**
Ensemble (forced WA)	18	29	67
LSTM	09	24	63
HieCoAtt	08	19	57
MCB	11	26	63

Table 6.10: Results obtained using the uncovering metric for the individual models and ensemble systems based on the normalized uncovered pixel ratio when any one system agrees with the ensemble's answer. Boldface indicates maximum in each column.

uncover based on the entire image as opposed to just the highlighted regions of the image. This choice of the fractions was decided so that not too much or too little of the image is uncovered at each step. Tables 6.8, 6.9 and 6.10 show the results obtained for the cases when *all three* systems, *any two* systems and *any one* system agree with the ensemble respectively.

We observed that when all three systems agreed with the ensemble's answer prediction, the weighted average explanation ensemble map was sufficient 71% of the time to arrive at the correct answer for a question when the uncovering was based on the entire image. For the same question and answer choices, the LSTM, the HieCoAtt, and the MCB models had explanation maps that were sufficient to arrive at the right answer only 59%, 49% and 51% of the time respectively, as shown in Table 6.8. We observed that the Turkers were unable to decide on an answer based on just the ensemble explanation and required the entire image for about 17% of the questions. However, for the individual models, the same fraction was 25% for the LSTM, 32% for the HieCoAtt and 28% for the MCB models. There was no agreement among the Turkers for the remaining percentage of the cases. The penalized weighted average was not applicable in this scenario since there was no system that disagreed with the ensemble's prediction.

When any two systems agreed with the ensemble's prediction, using the uncovering metric based on the entire image, we found that overall the ensemble explanation using the WA and PWA approaches was sufficient 78% and 74% of the time respectively, to arrive at the correct answer. The LSTM, the HieCoAtt and the MCB models' explanation maps were sufficient to answer the question correctly only about 69%, 61% and 63% of the time respectively. Table 6.9 shows the breakup of these percentages across the range of partially uncovered explanation maps. The Turkers were not able to decide on an answer based on just the ensemble explanation and required the entire image for the PWA and WA approaches for only 8% and 11% of the questions respectively. However, for the individual models, the same ratio was 18% for the LSTM, 22% for the HieCoAtt and 19% for the MCB models respectively. For the remaining percentage of cases, there was no agreement among the Turkers.

When any one system agreed with the ensemble's prediction, we found that overall the ensemble explanation using the WA and PWA approaches was sufficient 73% and 67% of the time respectively, to arrive at the correct answer. The LSTM, the HieCoAtt and the MCB models' explanation maps were sufficient to answer a question correctly only about 63%, 57% and 63% of the time respectively. Table 6.10 shows the breakup of these percentages across the range of partially uncovered explanation maps. The Turkers were not able to decide on an answer based on just the ensemble explanation and required the entire image for the PWA and WA approaches for only 11% and 18% of the questions respectively. However, for the individual models, the same ratio was 24% for the LSTM, 33% for the HieCoAtt and 22% for the MCB models respectively.

6.7 Chapter Summary

In this chapter, we introduced two new methods for ensembling visual explanations: (i) the weighted average and (ii) the penalized weighted average. We also proposed two metrics for evaluating the quality of visual explanations: (i) the comparison metric and (ii) the uncovering metric. These metrics measure explanation quality without relying on ground-truth human-generated explanation. We demonstrated our metrics by evaluating explanation maps generated by our ensemble system as well as three other VQA models. On average, our ensemble's explanation was qualitatively better than individual component models' explanation 61% of the time using the comparison metric and was sufficient to allow humans to arrive at the correct answer at least 64% of the time as indicated by the uncovering metric.

Chapter 7
Future Directions

As with all areas of research in AI, the work in this thesis answers some questions, but it also raises many new ones. In this chapter, we discuss possible future direction for our research focusing on two main areas: (i) Stacking With Auxiliary Features (SWAF) and (ii) Explainable AI (XAI).

7.1 Stacking With Auxiliary Features (SWAF)

In Chapter 5, we demonstrated SWAF on VQA. One of the auxiliary features we proposed was using the *visual* explanation generated by individual deep neural networks. Goyal et al. (2016) showed that VQA models focused on appropriate words in the question while answering. Their findings revealed that a word's POS tag determines its importance in a question. Unsurprisingly, the wh-words are the most important followed by the adjectives and then nouns. This means that the language component of a VQA model is as important as its visual component. As an extension of our visual explanation work, we believe that the words in the question to which a system attends can also be used to improve ensembling. The words in a question that a system focuses on would inform how reliable is its output. The correlation between different systems' word attention along with visual localization map could form more powerful auxiliary features for SWAF.

Another direct extension of our work would be to evaluate SWAF on other problems in natural language understanding and computer vision. Textual question answering in context and activity recognition in videos are promising avenues for SWAF application. There are several datasets for textual question answering in a wide range of domains including MS MARCO (Nguyen et al., 2016), NewsQA (Trischler et al., 2016) and SQuAD (Rajpurkar et al., 2016). UT-interaction (Ryoo et al., 2010) and UCF action[1] are some of the datasets for human activity recognition

[1]`http://crcv.ucf.edu/data/UCF_Sports_Action.php`

in videos.

The SWAF algorithm takes a data instance as input and outputs a binary accept/reject decision. For the successful application of SWAF to many of the machine learning applications, our work included turning problems with structured outputs into a binary classification of outputs produced by component systems. An exciting research direction would be to explore SWAF to actually combine structured outputs from multiple systems. Another avenue of research is to apply SWAF for problems in textual and visual data generation. Visual dialog is a task that requires an AI agent to hold a meaningful dialog with humans in natural language about visual content (Das et al., 2017b). Another example of a dialog dataset is the Ubuntu corpus (Lowe et al., 2015) which consist of natural conversational type dialogs.

Recently, ensembles of deep learning models have gained huge popularity because of their ground-breaking success in many AI applications (Fukui et al., 2016; He et al., 2016; Yang et al., 2018). However, many of these systems are ensembles of neural models with different seeds. Because the neural architecture remains the same, these underlying models make almost the same predictions and have correlated errors. On the other hand, SWAF relies on diverse models with somewhat de-correlated errors along with auxiliary features for making good predictions. How good is SWAF for combining neural models with different seeds? This could be an interesting question to be explored in the future. We think that there are benefits in ensembling neural models with different architectures over those with the same architecture but different seeds.

7.2 Explainable AI (XAI)

Our work on explainable AI used *visual* explanations for VQA. An obvious research problem is to explore using *textual* explanations along with the visual explanations for VQA. Park et al. (2016) propose a pointing and justification model (PJ-X) that attempts to generate a textual explanation as well as point to the evidence in the image. However, they use features from their original model that generates answers for image-question pairs and feed it into a new LSTM model for generating the textual explanation. Therefore, the explanation is generated in a "post-hoc" manner. A more faithful explanation would be one that attempts to explain the concepts in a region of the image the model attends to, using natural language. One way of finding such natural-language concepts in sub-regions of the image that contributed to the system's answer is by using *network dissection* (Bau et al., 2017). In this method, the semantics of the hidden units in a CNN are scored on how well they detect a number of visual concepts including objects, parts, scenes, textures, materials, and colors. These natural-language concepts can then be used to generate a coherent explanatory sentence using either a template-based approach or a trained LSTM.

Generating textual explanations has multiple advantages. First, it can be used along with visual explanations to provide a more powerful and complete justification for the model's decision. Second, textual explanations from multiple models can be used as auxiliary features just like our work on using the visual explanation for improving VQA. The idea behind using textual explanations as auxiliary features is to trust agreement between systems when their explanations are also coherent just like in the case of visual explanation. Machine translation metrics such as BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) can used for comparing similarity between system generated textual explanations. Another interesting feature that captures the semantics of the text is to use embeddings of the words in the justification for calculating similarity. We hope that using the textual and visual explanations together as features would enable the stacker to learn to rely on systems whose justification is congruent with its classification.

Ensembling natural-language explanations obtained from individual component systems is another interesting and challenging future direction. Combining textual explanations from multiple models is in itself a challenging task. Several techniques have been deployed for combining structured prediction model outputs for Machine Translation (MT) and Parsing. Minimum Bayes-Risk decoding is a very well known technique for finding a consensus translation that has the minimum expected loss under the distribution given by a k-best list of translations

(Kumar and Byrne, 2004). DeNero et al. (2010) proposed a model combination technique by learning a consensus model over the n-gram features of multiple underlying component MT models. A hierarchical system combination strategy at the word-, phrase- and sentence- levels has also been used in the past (Huang and Papineni, 2007). In parsing, there have been several work for combining structured textual output from multiple models (Sagae and Lavie, 2006; Martins et al., 2008; Fossum and Knight, 2009). Some of these ideas from MT and parsing could be adopted for research in ensembling textual explanations.

The ensembled visual explanation generated using our approach described in Chapter 6 can be combined with the ensembled textual explanation so that the natural-language concepts in the textual explanation directly point to corresponding visual-explanation regions in the image. By ensembling textual explanation along with the visual explanation, we would obtain an ensemble that is not just good in performance but also generates a consensus explanation for the ensemble.

Even though visual explanations generated by deep learning models are noisy and many times unconvincing to a human, we have seen that they can prove to be useful for classification in the form of auxiliary features to the stacker as demonstrated in Chapter 5. An interesting future direction would be to explore if there is any correlation between explanations that are rated qualitatively better by humans and those that are actually useful to the stacker.

Evaluating explanations is a challenging and an ongoing area of research. The metrics proposed by us in this thesis in Chapter 6 solves the problem of having to compare machine generated explanation to human-generated explanation. However, it is still not perfect and raises the question of whether the metric actually judges if the explanation is explaining what the model is doing. Arguably, the uncovering metric only judges if a given explanation is plausible or relevant. If we compared two explanation generating methods for the same model, ideally an evaluation metric would rate that explanation higher which actually tells what the model is doing. However, the comparison metric might fail to capture this subtlety and would instead choose the explanation that makes the model look better. An explanation evaluation metric that overcomes these challenges would be an interesting research direction.

An example of a reasonable explanation evaluation metric that uses crowd-sourcing is one that can evaluate if explanation allows the user to trust the machine's answer correctly or not on problems where the human is not an expert. The fine-grained recognition task such as the Caltech-UCSD bird species identification dataset (Welinder et al., 2010) is an example of such a problem.

Chapter 8
Conclusion

Time and again, ensembling methods have proven to be a powerful tool for important applications in the domain of Natural Language Processing and Computer Vision. Ensembles of deep learning models have produced state-of-the-art performance on problems that are shaping the future of how humans interact with AI systems. The tremendous success of these systems has raised questions about trust between humans and AI systems. This has led to a new direction of research on modern explainable AI systems.

In this thesis, we considered the research problem of ensembling multiple machine learning systems applied to problems with structured outputs. We proposed Stacking with Auxiliary Features (SWAF), a novel approach to ensembling multiple diverse system outputs, and demonstrated its success on a wide variety of challenging tasks. These tasks had data in multiple modalities. Knowledge Base Population (KBP) used a text corpus as data, Object detection used images and Visual Question Answering (VQA) included data in both modalities. We also proposed novel methods for ensembling and evaluating visual explanations of VQA models.

We introduced two types of auxiliary features: (i) *instance features* and (ii) *provenance features*. The instance features enable the stacking meta-classifier to learn what component systems to rely on for what instance types. The provenance features enable the stacker to learn that an output is reliable not merely if multiple systems agree on it but if they also agree "where" they got the output. The auxiliary features enable the system to learn to appropriately use provenance and instance information to aid the optimal integration of multiple systems.

For the KBP task of relation extraction, the entity type and relation type served as the instance features. For entity linking, the entity types served as instance features. For the object detection task, the object categories were the instance features. The provenance feature for both KBP tasks was a measure of overlap between systems' provenance. For the object detection, the bounding boxes around object instances served as provenance features.

We also proposed a novel variant of SWAF for ensembling component systems both with training data (supervised systems) and without training data (unsupervised systems). We demonstrated this new variant of SWAF on the KBP tasks of relation extraction and entity linking. We obtained promising results, achieving a new state-of-the-art on both tasks and beating our own prior performance using only systems with training data. We found that adding the unsupervised ensemble increased recall substantially.

We proposed four different categories of auxiliary features to ensemble VQA systems, three of which can be inferred from an image-question pair. For the fourth category of features, we introduced the novel idea of using explanations to improve ensembling. We demonstrated how visual explanations for VQA (represented as localization maps) could be used to aid stacking with auxiliary features. This approach effectively utilized information on the degree to which systems agree on the explanation of their answers. We showed that the combination of all of these categories of auxiliary features, including explanation, gave the best performance.

Visual explanations, which highlight the parts of an image that support a vision system's conclusion, can help us understand the decisions made by Visual Question Answering (VQA) systems. This in turn aids error analysis, increases transparency and helps build trust with human users. Towards this end, we proposed two novel methods for ensembling the explanation maps of multiple systems to produce improved ensemble explanations. Crowd-sourced human evaluations indicated that our ensemble visual explanation significantly qualitatively outperformed each of the component systems' visual explanations.

Research on explainable AI systems is incomplete without a proper evaluation metric to measure their effectiveness and usefulness. In this thesis, we proposed two such metrics that measure explanation quality without the need for a ground-truth human-generated explanation. Our evaluation metrics rely on crowd-sourced human judgments on simple tasks involving comparing visual explanations or making decisions from partially revealed images. We demonstrated our metrics by evaluating explanation maps generated by our ensemble system as well as three component VQA models.

The work done in this thesis served to advance the state-of-the-art on several machine learning applications in computational linguistics and vision. We are optimistic that our research will motivate the development of improved AI systems that do not just learn and make decisions but also explain their choices. We are hopeful of a bright future for AI where humans and intelligent systems interact seamlessly.

References

Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*, 2016.

David W. Aha, Trevor Darrell, Michael Pazzani, Darryn Reid, Claude Sammut, and Peter Stone (Eds.). Explainable Artificial Intelligence (XAI) Workshop at IJCAI, 2017.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*, 2016.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2016)*, pages 39–48, 2016.

Gabor Angeli, Arun Chaganty, Angel Chang, Kevin Reschke, Julie Tibshirani, Jean Y Wu, Osbert Bastani, Keith Siilats, and Christopher D Manning. Stanford's 2013 KBP system. In *TAC2013*, 2013.

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Chaganty, Jason Bolton, Melvin Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D. Manning. Bootstrapped self training for knowledge base population. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*, 2015.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72, 2005.

Michele Banko, Oren Etzioni, and Turing Center. The tradeoffs between open and traditional relation extraction. In *ACL08*, volume 8, pages 28–36, 2008.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proccedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3319–3327, 2017.

Thomas Berg and Peter N Belhumeur. How do you tell a blackbird from a crow? In *Proceedings of ICCV2013*, 2013.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, 2008.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. What do different evaluation metrics tell us about saliency models? *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI2018)*, 2018.

Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. ABC-CNN: An attention based convolutional neural network for Visual Question Answering. *arXiv preprint arXiv:1511.05960*, 2015.

Xiao Cheng, Bingling Chen, Rajhans Samdani, Kai-Wei Chang, Zhiye Fei, Mark Sammons, John Wieting, Subhro Roy, Chizheng Wang, and Dan Roth. Illinois cognitive computation group UI-CCG TAC 2013 entity linking and slot filler validation systems. In *Proceedings of the Sixth Text Analysis Conference (TAC2013)*, 2013.

Franois Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, volume 1, pages 886–893. IEEE, 2005.

Abhishek Das, Harsh Agrawal, C Lawrence Zitnick, Devi Parikh, and Dhruv Batra. Human Attention in Visual Question Answering: Do Humans and Deep Networks Look at the Same Regions? *arXiv preprint arXiv:1606.03556*, 2016.

Abhishek Das, Harsh Agrawal, Larry Zitnick, Devi Parikh, and Dhruv Batra. Human attention in visual question answering: Do humans and deep networks look at the same regions? *Computer Vision and Image Understanding*, 163:90–100, 2017.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. Visual Dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

John DeNero, Shankar Kumar, Ciprian Chelba, and Franz Och. Model combination for machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 975–983. Association for Computational Linguistics, 2010.

T. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2000.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International conference on Knowledge Discovery and Data mining*, pages 601–610. ACM, 2014.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

Asif Ekbal and Sriparna Saha. Stacked ensemble coupled with feature selection for biomedical entity extraction. *Knowledge-Based Systems*, 46:22–32, 2013.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. Overview of linguistic resources for the TAC KBP 2015 evaluations: Methodologies and results. In *TAC 2015*, 2015.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Nicolas Fauceglia, Yiu-Chang Lin, Xuezhe Ma, and Eduard Hovy. CMU System for Entity Discovery and Linking at TAC-KBP 2015. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*, 2015.

Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

Tim Finin, Dawn Lawrie, Paul McNamee, James Mayfield, Douglas Oard, Nanyun Peng, Ning Gao, Yiu-Chang Lin, Josh MacLin, and Tim Dowd. HLTCOE participation in TAC KBP 2015: Cold Start and TEDL. In *Proceedings of the Eighth Text Analysis Conference (TAC 2015)*, 2015.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. ACL, 2003.

Victoria Fossum and Kevin Knight. Combining constituent parsers. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 253–256. Association for Computational Linguistics, 2009.

Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

Lex Fridman, Benedikt Jenik, and Jack Terwilliger. DeepTraffic: Driving Fast through Dense Traffic with Deep Reinforcement Learning. *arXiv preprint arXiv:1801.02805*, 2018.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal Compact Bilinear pooling for Visual Question Answering and Visual Grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP2016)*, 2016.

Jing Gao, Feng Liang, Wei Fan, Yizhou Sun, and Jiawei Han. Graph-based consensus maximization among multiple supervised and unsupervised models. In *NIPS2009*, pages 585–593, 2009.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2016)*, pages 317–326, 2016.

Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR2015)*, pages 437–446, 2015.

Ross Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV2015)*, 2015.

Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision making and a "right to explanation". *AI Magazine*, 38(3), 2017.

Yash Goyal, Akrit Mohapatra, Devi Parikh, and Dhruv Batra. Towards Transparent AI Systems: Interpreting Visual Question Answering Models. *arXiv preprint arXiv:1608.08974*, 2016.

David Gunning. Explainable Artificial Intelligence (XAI), DARPA Broad Agency Announcement. 2016.

Zhengyan He, Shujie Liu, Yang Song, Mu Li, Ming Zhou, and Houfeng Wang. Efficient collective entity linking with stacking. In *EMNLP2013*, pages 426–435, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR2016)*, 2016.

Benjamin Heinzerling, Alex Judea, and Michael Strube. HITS at TAC KBP 2015: Entity Discovery and Linking,and Event Nugget Detection. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*, 2015.

John C. Henderson and Eric Brill. Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP99)*, pages 187–194, College Park, MD, 1999.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating Visual Explanations. *arXiv preprint arXiv:1603.08507*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Fei Huang and Kishore Papineni. Hierarchical system combination for machine translation. In *EMNLP-CoNLL*, pages 277–286, 2007.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Dinesh Jayaraman and Kristen Grauman. Zero-shot recognition with unreliable attributes. In *Advances in Neural Information Processing Systems*, pages 3464–3472, 2014.

Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1148–1158. Association for Computational Linguistics, 2011.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. Overview of TAC-KBP2015 Tri-lingual Entity Discovery and Linking. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*, 2015.

H. Ji, J. Nothman, and H. Trang Dang. Overview of TAC-KBP2016 Tri-lingual EDL and its impact on end-to-end Cold-Start KBP. In *Proceedings of TAC*, 2016.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Edward Johns, Oisin Mac Aodha, and Gabriel J Brostow. Becoming the expert-interactive multi-class machine teaching. In *Proceedings of CVPR2015*, 2015.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE, 2017.

Kushal Kafle and Christopher Kanan. Answer-type prediction for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2016)*, June 2016.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR2014)*, pages 1725–1732, 2014.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems (NIPS2015)*, pages 3294–3302, 2015.

Bryan Kisiel, Bill McDowell, Matt Gardner, Ndapandula Nakashole, Emmanouil A. Platanios, Abulhair Saparov, Shashank Srivastava, Derry Wijaya, and Tom Mitchell. CMUML system for KBP 2015 Cold Start Slot Filling. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*, 2015.

Marcin Korytkowski, Leszek Rutkowski, and Rafał Scherer. Fast image classification by boosting fuzzy classifiers. *Information Sciences*, 327:175–182, 2016.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision (IJCV2017)*, 123(1):32–73, 2017.

Shankar Kumar and William Byrne. Minimum bayes-risk decoding for statistical machine translation. Technical report, DTIC Document, 2004.

H Chad Lane, Mark G Core, Michael Van Lent, Steve Solomon, and Dave Gomboc. Explainable artificial intelligence for training and tutoring. Technical report, 2005.

Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV2014)*, pages 740–755. Springer, 2014.

Ryan Lowe, Nissan Pow, Iulian V Serban, and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 285, 2015.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems (NIPS2016)*, pages 289–297, 2016.

Xiaoqiang Luo. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 25–32, 2005.

Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1682–1690. Curran Associates, Inc., 2014.

Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Ensemble of exemplar-svms for object detection and beyond. In *2011 International Conference on Computer Vision*, pages 89–96. IEEE, 2011.

André FT Martins, Dipanjan Das, Noah A Smith, and Eric P Xing. Stacking dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 157–166. Association for Computational Linguistics, 2008.

David McClosky, Sebastian Riedel, Mihai Surdeanu, Andrew McCallum, and Christopher D Manning. Combining joint models for biomedical event extraction. *BMC Bioinformatics*, 2012.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. pages 1003–1011. Association for Computational Linguistics (ACL2009), 2009.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2016)*, pages 30–38, 2016.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. Attentive explanations: Justifying decisions and pointing to the evidence. *arXiv preprint arXiv:1612.04757*, 2016.

Ted Pedersen. A Simple Approach to Building Ensembles of Naive Bayesian Classifiers for Word Sense Disambiguation. In *Proceedings of the Third Conference on Natural language learning (NAACL2000)*, pages 63–69, 2000.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP2014)*, pages 1532–1543, 2014.

Lorenzo Peppoloni, Massimo Satler, Emanuel Luchetti, Carlo Alberto Avizzano, and Paolo Tripicchio. Stacked generalization for scene analysis and object recognition. In *Intelligent Engineering Systems (INES), 2014 18th International Conference on*, pages 215–220. IEEE, 2014.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Peter J. Bartlett, Bernhard Schölkopf, Dale Schuurmans, and Alex J. Smola, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

Nazneen Fatema Rajani and Raymond J. Mooney. Combining Supervised and Unsupervised Ensembles for Knowledge Base Population. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*, 2016.

Nazneen Fatema Rajani and Raymond J. Mooney. Ensembling visual explanations for vqa. In *Proceedings of the NIPS 2017 workshop on Visually-Grounded Interaction and Language (ViGIL)*, December 2017.

Nazneen Fatema Rajani and Raymond J. Mooney. Stacking With Auxiliary Features. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI2017)*, Melbourne, Australia, August 2017.

Nazneen Fatema Rajani and Raymond J. Mooney. Stacking With Auxiliary Features for Visual Question Answering. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

Nazneen Fatema Rajani, Vidhoon Viswanathan, Yinon Bentor, and Raymond J. Mooney. Stacked Ensembles of Information Extractors for Knowledge-Base Population. In *Association for Computational Linguistics (ACL2015)*, pages 177–187, Beijing, China, July 2015.

Nazneen Fatema Rajani, Mihaela A. Bornea, and Ken Barker. Stacking With Auxiliary Features for Entity Linking in the Medical Domain. In *Proceedings of the ACL workshop on BioNLP 2017*, pages 39–47, 2017.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Vasili Ramanishka, Abir Das, Jianming Zhang, and Kate Saenko. Top-down visual saliency guided by captions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR2017)*, pages 3135–3144, 2017.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS2015)*, 2015.

Lev Reyzin and Robert E Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the 23rd international conference on Machine learning*, pages 753–760. ACM, 2006.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2016)*, 2016.

Sebastian Riedel, David McClosky, Mihai Surdeanu, Andrew McCallum, and Christopher D Manning. Model combination for event extraction in bionlp 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 51–55. ACL2011, 2011.

Miguel Rodriguez, Sean Goldberg, and Daisy Zhe Wang. University of Florida DSR lab system for KBP slot filler validation 2015. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*, 2015.

Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. In *Proceedings of IJCAI2017*, 2017.

Benjamin Roth and Dietrich Klakow. Cross-language retrieval using link-based language models. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 773–774. ACM, 2010.

Benjamin Roth, Tassilo Barth, Michael Wiegand, et al. Effective slot filling based on shallow distant supervision methods. *Proceedings of the Seventh Text Analysis Conference (TAC2013)*, 2013.

Benjamin Roth, Nicholas Monath, David Belanger, Emma Strubell, Patrick Verga, and Andrew McCallum. Building knowledge bases with universal schema: Cold start and slot-filling approaches. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

MS Ryoo, Chia-Chih Chen, JK Aggarwal, and Amit Roy-Chowdhury. An overview of contest on semantic description of human activities (sdha) 2010. In *Recognizing Patterns in Signals, Speech, Images and Videos*, pages 270–285. Springer, 2010.

Kenji Sagae and Alon Lavie. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132. Association for Computational Linguistics, 2006.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

Mark Sammons, Yangqiu Song, Ruichen Wang, Gourab Kundu, et al. Overview of UI-CCG systems for event argument extraction, entity discovery and linking, and slot filler validation. *Proceedings of the Seventh Text Analysis Conference (TAC2014)*, 2014.

Mark Sammons, Haoruo Peng, Yangqiu Song, Shyam Upadhyay, Chen-Tse Tsai, Pavankumar Reddy, Subhro Roy, and Dan Roth. Illinois CCG TAC 2015 Event

Nugget, Entity Discovery and Linking, and Slot Filler Validation Systems. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*, 2015.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *The IEEE International Conference on Computer Vision (ICCV2017)*, Oct 2017.

Georgios Sigletos, Georgios Paliouras, Constantine D Spyropoulos, and Michalis Hatzopoulos. Combining information extraction systems using voting and stacked generalization. *The Journal of Machine Learning Research*, 6:1751–1782, 2005.

Avirup Sil, Georgiana Dinu, and Radu Florian. The IBM systems for trilingual entity discovery and linking at TAC 2015. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*, 2015.

A. Sil, G. Dinu, and R. Florian. The IBM systems for trilingual entity discovery and linking at TAC 2016. In *Proceedings of TAC*, 2016.

Joseph Sill, Gábor Takács, Lester Mackey, and David Lin. Feature-weighted linear stacking. *arXiv preprint arXiv:0911.0460*, 2009.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. In *Proceedings of ICLR2015*.

Stephen Soderland, Natalie Hawkins, Gene L. Kim, and Daniel S. Weld. University of Washington system for 2015 KBP cold start slot filling. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*, 2015.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 217–223, 2017.

Mihai Surdeanu and Heng Ji. Overview of the English slot filling track at the TAC2014 knowledge base population evaluation. In *Proceedings of the Seventh Text Analysis Conference (TAC2014)*, 2014.

Mihai Surdeanu. Overview of the TAC2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *Proceedings of the Sixth Text Analysis Conference (TAC2013)*, 2013.

Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *arXiv preprint arXiv:1708.02711*, 2017.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*, 2016.

Jasper RR Uijlings, Koen EA Van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International Journal of Computer Vision (IJCV)*, 104(2):154–171, 2013.

R. Vilalta and Y. Drissi. A Perspective View and Survey of Meta-learning. *Artificial Intelligence Review*, 2002.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

I-Jeng Wang, Edwina Liu, Cash Costello, and Christine Piatko. JHUAPL TAC-KBP2013 slot filler validation system. In *TAC2013*, 2013.

P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

Matthew Whitehead and Larry Yaeger. Sentiment mining using ensemble classification models. In Tarek Sobh, editor, *Innovations and Advances in Computer Sciences and Engineering*. SPRINGER, Berlin, 2010.

David H. Wolpert. Stacked Generalization. *Neural Networks*, 5:241–259, 1992.

Huijuan Xu and Kate Saenko. Ask, Attend and Answer: Exploring question-guided spatial attention for visual question answering. In *Proceedings of ECCV2016*, 2016.

Dongdong Yang, Senzhang Wang, and Zhoujun Li. Ensemble neural relation extraction with adaptive boosting. *arXiv preprint arXiv:1801.09334*, 2018.

Xiaoxin Yin, Jiawei Han, and Philip S Yu. Truth discovery with multiple conflicting information providers on the web. *Knowledge and Data Engineering, IEEE Transactions on*, 20(6):796–808, 2008.

Dian Yu, Hongzhao Huang, Taylor Cassidy, Heng Ji, Chi Wang, Shi Zhi, Jiawei Han, Clare Voss, and Malik Magdon-Ismail. The wisdom of minority: Unsupervised slot filling validation based on multi-dimensional truth-finding. In *Proc. The 25th International Conference on Computational Linguistics (COLING2014)*, 2014.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.

Y. Zhang, A. Chaganty, A. Paranjape, D. Chen, J. Bolton, P. Qi, and C. Manning. Stanford at TAC KBP 2016: Sealing Pipeline Leaks and Understanding Chinese. In *Proceedings of TAC*, 2016.

Xiangzeng Zhou, Lei Xie, Peng Zhang, and Yanning Zhang. An ensemble of deep neural networks for object tracking. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 843–847. IEEE, 2014.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene CNNs. In *Proceedings of the International Conference on Learning Representations (ICLR2015)*, 2015.

Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167*, 2015.

Vita

Nazneen Fatema Rajani was born and raised in Mumbai, India. She obtained her Bachelor's degree from BITS - Pilani, Goa in 2011. She then moved to the United States in 2012 to join the University of Texas at Austin to pursue her graduate studies. She obtained her Masters in Computer Science in 2014 and continued working on her doctoral studies, where she has been working in the areas of Natural Language Processing, Computer Vision and at the intersection of both these areas.

Permanent Address: me@nazneenrajani.com

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1]LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.