CROWDSOURCING ANNOTATION FOR MACHINE LEARNING IN NATURAL LANGUAGE PROCESSING TASKS

by

Omar F. Zaidan

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

April, 2012

Abstract

Human annotators are critical for creating the necessary datasets to train statistical learners, but annotation cost and limited access to qualified annotators forms a data bottleneck. In recent years, researchers have investigated overcoming this obstacle using *crowdsourcing*, which is the delegation of a particular task to a large group of untrained individuals rather than a select trained few.

This thesis is concerned with crowdsourcing annotation across a variety of natural language processing tasks. The tasks reflect a spectrum of annotation complexity, from simple labeling to translating entire sentences. The presented work involves new types of annotators, new types of tasks, new types of data, and new types of algorithms that can handle such data.

The first part of the thesis deals with two text classification tasks. The first is the identification of dialectal Arabic sentences. We use crowdsourcing to create a large annotated dataset of Arabic sentences, which is used to train and evaluate language models for each Arabic variety. We also introduce a new type of annotations we call *annotator rationales*, which complement traditional class labels. We collect rationales for dialect identification and for a sentiment analysis task on movie reviews. In both tasks, adding rationales yields significant accuracy improvements.

In the second part, we examine how crowdsourcing can be beneficial to machine translation (MT). We start with the evaluation of MT systems, and show the potential of crowdsourcing to edit MT output. We also present a new MT evaluation metric, RYPT, that is based on human judgment, and well-suited for a crowdsourced setting. Finally, we demonstrate that crowdsourcing can be used to collect translations. We

discuss a set of features that help distinguish well-formed translations from those that are not, and show that crowdsourcing yields high-quality translations at a fraction of the cost of hiring professionals.

Thesis Committee:

Chris Callison-Burch
Associate Research Professor
Department of Computer Science
Johns Hopkins University

David Yarowsky
Professor
Department of Computer Science
Johns Hopkins University

Christine D. Piatko
Principal Research Scientist
Johns Hopkins University Applied Physics Laboratory

Acknowledgments

My awesome advisor told me that a majority of people reading a Ph.D. thesis "will only ever read the Introduction and Conclusion." If it were up to me, I'd want them to read **this** part too, because those mentioned below are why this thesis exists.

First and foremost, I thank my family. Mom and Dad, thank you for the HUGE amounts of love and support. Ahmad, you have no idea how much I have learned from you. You are a freaking genius. Mona, thank you for showing me that two people who might seem different are actually so similar. Also, thank you for the brilliant music. And thank you guys for bringing Gergana and Na'el into our lives. Maya, Hanin, Shams, and Adam make me extremely thankful to be an uncle.

And then there's Khaled, one of the smartest and most amazing guys I know. If my 'mentorship' had anything to do with that, then I'm incredibly proud. Thank you for the memories.[1] You were once my adorable baby brother (*seeboo!*), but became a person for whom I have much respect, in whom I place my trust, and with whom I want to do **everything**. I only wish I could spend even more of my time with you.

If there's one person who suffered with me through my Ph.D., that would be my Dima. Boo, thank you for toughing it out with me. I can't wait to meet our kids and tell them our story. Thank you most importantly for showing me the other way of doing things. I love you, and we are going to do great things together. I promise.

My thesis advisor, Chris Callison-Burch, played a big part in shaping and promoting my research. He helped with too many things to even try to recount. I'm also immensely proud of our work, and cherish the fact that I have a CCB number of 1. And on a personal level, he is the most affable, supportive, and just plain fun person.

[1] I forgive you for the LEGO. You forgive me for the unfinished airport, yes?

Chris, I miss hanging out with you already. I wish you and Dawn all the best.

Christian Scheideler was my first advisor at JHU, but by my 2nd year, had accepted a position at TU München. Christian helped me survive a tough first semester, and I also say *Danke vielmals* for the offer to accompany him to Munich. Jason Eisner then played a pivotal role advising me, and our collaboration resulted in several papers and got my foot in CLSP's door. Jason, thank you for being an amazing teacher, and for taking a chance on a grad student whose only exposure to NLP was your course.

David Yarowsky and Christine Piatko rounded out my thesis committee, and provided many suggestions to improve the quality of this document.[2] I also want to thank my GBO committee: Chris, Jason, Christine, Sanjeev Khudanpur, and Mark Dredze. I was ten times more nervous about the GBO than my defense(!), but you guys turned it into an enjoyable conversation about my work.

CS and CLSP staff greatly reduced the amount of stress I endured. At CS, Debbie and Cathy have helped with scheduling and (a lot of) paperwork. At CLSP, Monique, Desirée, and Justin have helped me with payroll, more paperwork, and IT support. Much thanks goes to Laura Graham, who I'm convinced basically knows *everything*, and Steve Rifkin, who is always super responsive and super friendly.

I've had fantastic educators as an undergrad at SLU. Thank you Mike Sheard, Jeff Greathouse, and the folks at MCSS (Maegan Bos, Jim DeFranza, Ed Harcourt, Brian Ladd, Patti Frazer Lock). In my formative years, Ms. Lamia Shloudi taught me how to think mathematically, and Mr. Salah Rammal taught me BASIC and made me realize, hey, I can make a computer do useful stuff. Ms. Fatima Dabbas and Mr. Omar Tayeh launched me on a great career path. Thank you all. Also, thank you Ms. Rula Kamal, for being, simply, the best.

Whether they know it or not, I learned a lot from: Eftekhar Al-Farkh (RIP), Laila Bustami, Rami Bustami (RIP), Sam Carliles, Rance Davis, Nizar Habash, John Jaunzems, Fred Jelinek (RIP), Zhifei Li, John Makhoul, Ben Mitchell, and Noah Smith. I thank you!

I have been fortunate to have made really awesome friends at JHU/Baltimore:

[2]Naturally, all remaining errors are either mine, or the doing of a very sneaky troll.

Abby, Alex “Sasha” Klementiev, Charley, Hassan, Jay, Mahmoud, Megan, Mike L., Osama, Sammy, and Siddharth A. Thank you for the great and generally hilarious times. To my pre-JHU dear friends Alba, Alejandro, Hamza, Mais, and Mohammad A.S.: thank you for pretending to be so interested in what I do. To Abdul, Ammar, Amr, Bashar I./S./T., Lara, and Lindsay: one of my greatest regrets is not being able to stay in touch with you as well as I wanted. I’m sorry for that.

Last but not least, Juri Ganitkevitch is all kinds of awesome. He is a bauce, my friend, my bro, and, yes, my dawg. YOU NEED TO BEFRIEND HIM IMMEDIATELY.

I’m also thankful for the chance to have met many other people whose time at JHU overlapped with mine: Adam G., Adam L., Allen A.W., Anni, Anoop, Ariya, Arnab, Ashley, Balakrishnan, Ben M., Ben V.D., Binit, Brian, ByungGyu, Carolina, Chris W., Chungwei, Con, Courtney, Damianos, Dan, David S., Delip, Ehsan, Elliott, Eric, Habibullah, Haolang, Hari, Jason S., John B., John K., Jonny, Kailash, Kapil, Keith, Kuriakose, Lane, Markus, Matt G., Matt P., Micha, Michael A., Michael P., Mike C., Ming, Nick, Nikesh, Nishant, Noamaan, Omar A., Puyang, Ramendra, Raphael, Razvan, Roy, Rui, Safi, Samuel, Scott, Siddharth S., Sivaram, Sridhar, Sriram, Sushant, Svitlana, Thomas, Usman, Ves “Vest” Stoyanov, Vijay, Wei, Wes, Wren, Yahnatan, Yi, Zach P., and Zach S. Chris Dyer counts too. ◡̈

During my time at JHU, I was supported by several teaching assistanships from the CS department, as well as DARPA’s GALE Project, the WSE-APL Partnership Fund, and Raytheon BBN Technologies. Further funds to support my work also came through EuroMatrix, NSF, Microsoft, and Google.

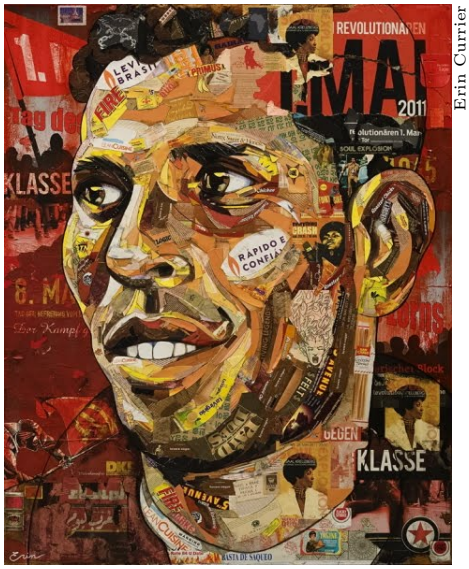
Thank you JHU for inviting many great speakers. Thank you Microsoft and Google for developing great technology. Thank you Erin Currier for the portrait on the next page.[3] Thank you Zaidans for a 17/12 that will go down in history. And thank you Chipotle, Billy Joel, and Jon Stewart for helping me maintain my sanity.

[3]URL: http://erincurrierfineart.com/mBouazizi.html

This thesis is dedicated to

Mohammed Bouazizi

(29 March 1984 - 4 January 2011)



Erin Currier

إذا الشعب يوما أراد الحياة فلا بد أن يستجيب القدر

ولا بد لليل أن ينجلي ولا بد للقيد أن ينكسر

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Natural language processing (NLP) applications are becoming ubiquitous, in the form of programs that process human speech, engines that find information in large swaths of online data, and apps that translate between languages. Much of the progress on difficult NLP problems has been due to empirical methods, and statistical learning has become a standard approach for most tasks. The cornerstone of statistical learning is the availability of a large and representative training set. By detecting the hidden patterns within the training set, a statistical learner is able to generalize to future instances of the problem, including instances not encountered during training. Hence, the conventional wisdom is that *more data are better data* (Church and Mercer, 1993).

In this thesis, we are concerned with training sets that require obtaining judgments from human annotators. We propose *delegating annotation tasks to a large group of untrained annotators*, rather than to a select few trained annotators. We show that ***crowdsourcing***, as it has come to be known, presents a unique opportunity to obtain massive amounts of data in a relatively short period of time, and we show that untrained workers can produce annotations that are high enough in quality to train statistical NLP models.

The main contribution of this thesis is to illustrate how large quantities (and entirely new types) of training data can be collected via crowdsourcing, for a variety of NLP tasks. We use crowdsourcing to create training datasets for tasks for

which no training data had previously existed, we introduce new annotation tasks and novel algorithms that use the resulting data, and we show that complex tasks can be delegated to non-professionals and completed at a fraction of the cost of hiring professionals. Throughout the thesis, we present approaches for detecting low-quality data, allowing us to exercise effective quality control over the collected data.

1.1 Major Themes and Overview

We will see how crowdsourcing annotation tasks can help statistical learning in NLP, by aiding either the training, tuning, or evaluation of statistical learners. The tasks discussed in this thesis reflect a spectrum of annotation complexity, from simple label selection (from a finite set of class labels), through selecting textual segments, editing sentences so they are more fluent and meaningful, and finally translating entire sentences. Despite this variety, a number of important themes emerge throughout the thesis, and come into play across the different annotation tasks. Those themes reflect the inherent difficulties of crowdsourcing (which we must deal with properly), and the inherent difficulties of traditional annotation (which we aim to overcome with the aid of crowdsourcing).

1.1.1 Quality Control

The very nature of crowdsourcing – a parallelized effort with low barriers to entry – is also why it comes with its own set of challenges. When annotations are provided by anonymous untrained workers, in a setting that allows limited direct feedback about the quality of their work, collected data can be of questionable quality. In this thesis, we devise strategies that help us practice effective quality control. Those methods are helpful in detecting spammy annotators, who are only interested in earning the financial reward. As they make no effort to perform the task faithfully, they are responsible for much of data quality degradation. We are also able to detect a more subtle form of undesired behavior, which occurs when a worker is willing but unable to perform the task properly, due to difficulties they face when performing the

Src:	باراک اوبامہ امریکہ ایران کے ساتھ نئی حکمت عملی اپنائے گا
Ref1:	Barack Obama: America Will Adopt New Strategy with Iran
Ref2:	America to Adopt New Strategy for Iran: Barack Obama
Ref3:	Barack Obama: America Will Adopt a New Iran Strategy
Turk1:	Barak Obam will do a new policy with Iran.
Turk2:	Barack Obama: America will use a new policy towards Iran.
Turk3:	Barak Obama and America weave new evil strategies against Iran.

Figure 1.1: In Chapter 6, we investigate crowdsourcing an Urdu-to-English translation task. This Figure shows several translations for an Urdu headline, produced by professional and non-professional translators.

annotation task (difficulties that might be inherent to the task itself).

For example, Chapter 6 gives an overview of our effort to crowdsource an Urdu-to-English translation task. The participants in that task were non-professional translators, who were furthermore non-native speakers of English. As a result, their submissions are characteristic of speakers of English as a second language (Figure 1.1).

We discuss a set of strategies that help us minimize the amount of poor translations. These quality control measures include simple things like rendering the source text into images, to prevent workers from using MT systems. They also include crowdsourcing an entire quality evaluation task, to rank the alternative translations that were collected for every foreign sentence. We also developed a statistical model that allows us to score each translation and each translator. Our model computes a set of features that enable us to distinguish well-formed translations from those that are not, and effective translators from those who are not.

We quantitatively show that our approach allows us to select translations of high quality. Figure 1.2 shows that **unfiltered** crowdsourced translations would have a BLEU score of 28.1 measured against professional translators. Once we employ our

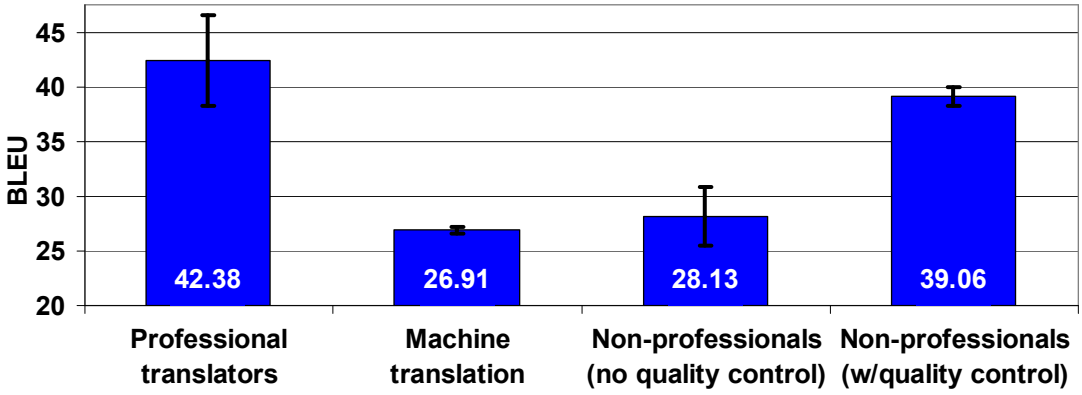


Figure 1.2: The effect of quality control on crowdsourced translation quality. The BLEU score of crowdsourced translations (right-most column) is within the error bars of the BLEU score of professional translations (left-most column), but only if proper quality control is exercised. If crowdsourced translations are not filtered first, they are not much better than machine translation output.

quality control measures, we can achieve a BLEU score of 39.1, which falls within the range of professional translators, whose BLEU scores range from 36.3 to 44.4.

1.1.2 Novel Annotation Schemes (and Novel Models Benefiting from Them)

Another major theme of the thesis is the creation and collection of *new types of data*, and proposing *new models* that take advantage of such data. This theme is tightly connected to crowdsourcing, since new types of data could be difficult to collect otherwise. High overhead costs are usually associated with creating even small datasets, and most research is constrained to existing datasets because of the prohibitively high cost of creating new ones. Crowdsourcing frees us from this constraint This freedom allows us to solve new tasks and to develop new annotation schemes. New types of data that go beyond simple class labels allow us to train new styles of models. We also show how the power of the crowd can be used to collect annotations that would be infeasible to obtain in a typical setting, if a very large volume of judgments is required.

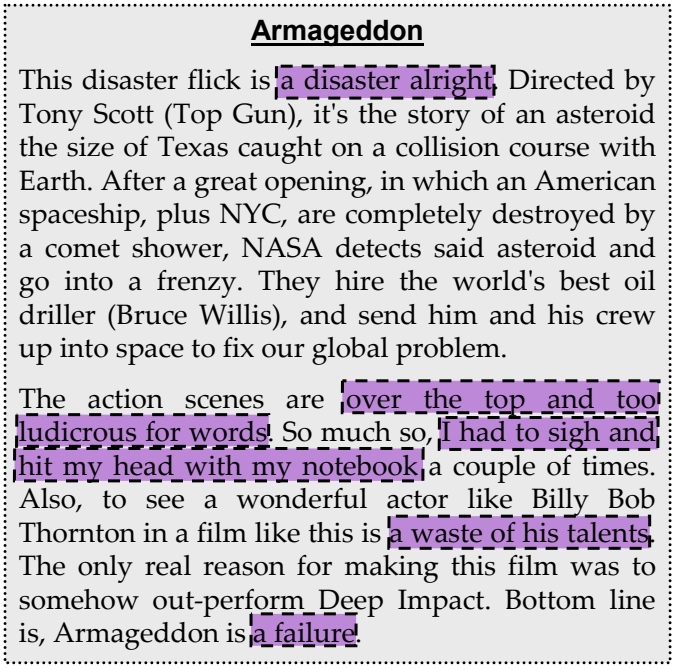


Figure 1.3: In Chapter 4, we introduce *annotator rationales*, a new type of data that aids text classification. This is a movie review that has had textual segments of it highlighted as rationales supporting a negative class label.

In Chapter 4, we propose a new learning framework aided by a new type of annotations that we call *annotator rationales*. Traditionally, annotators had been asked to provide only **what** the correct label is. We also propose that they tell us **why** they chose a particular class label, providing additional hints to the learner to generalize better to unseen data. For example, Figure 1.3 shows a movie review classified as a negative example, along with the annotator-provided rationales that support this class label.

Rationales provide a new source of information for the learner. The novelty of this data type allows us to develop novel methods to incorporate rationales into the training procedure. We propose two such methods. The first method augments a support vector machine by enforcing a separation between the original training examples and their corresponding *contrast* examples, in which rationales are removed (Figure 1.4). The second method directly models annotator's behavior using a conditional random

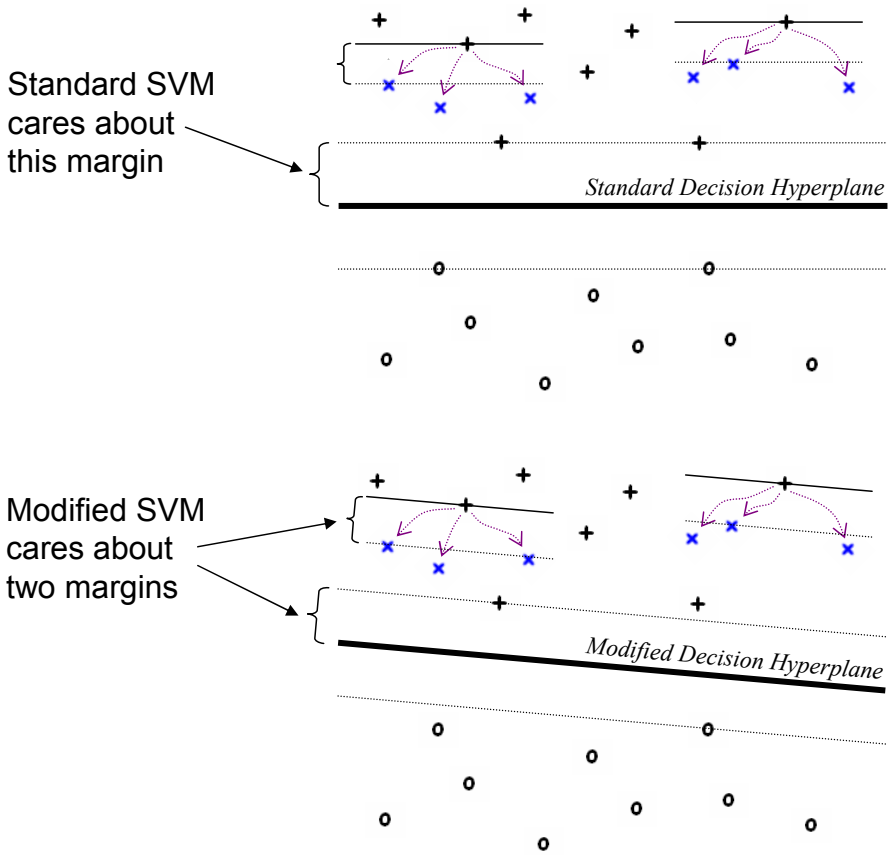


Figure 1.4: An illustration comparing the standard SVM to the modified SVM of Chapter 4. The standard SVM attempts to maximize the separation between positive (+) and negative (o) examples. The modified SVM deviates from the maximum-margin hyperplane if that increases the separation between original examples and the corresponding 'contrast' examples (the blue xs), which have rationales masked out.

field, and incorporates it into the training objective of a log-linear classifier. In the task of distinguishing positive movie reviews from negative ones, both methods yield significant improvements over two strong baselines that use only class labels, increasing accuracy rates from around 89% to over 92%. We also show how rationale data can be collected, via crowdsourcing, for Arabic dialect identification, also leading to improved accuracy rates.

Later in Chapter 5, we show how the power of the crowd can be harnessed to collect annotations that would be infeasible to obtain otherwise. We present a new

manual evaluation metric, RYPT, that requires annotators to make judgments on the acceptability of individual constituents in the output. A large number of those phrase-level judgments are needed to compute RYPT scores, making the metric particularly well-suited for a crowdsourced setting. Crowdsourcing allows the fast, distributed collection of those judgments, taking advantage of the **crowd** in crowdsourcing.

We demonstrate that it is feasible to collect a large enough amount of labels to optimize machine translation system parameters to *manual* human judgments. We also demonstrate that RYPT is an acceptable proxy for human judgment. When taking the highest-scoring translation according to RYPT, and comparing it to the highest-scoring translation according to the standard automatic metric BLEU, we find that human judges prefer the RYPT selection 45% of the time, compared to only 29% for the BLEU selection.

1.1.3 Reducing Annotation Cost

Each example in a training set must be annotated with its output (e.g. a class label in document classification, a translation in machine translation), which is provided by a human annotator. While supervised machine learning systems are capable of achieving high accuracy rates, it is usually labor-intensive and expensive to construct the (large) needed number of training examples. Since human annotation is a costly, as well as time-consuming, process, annotation cost is a bottleneck for many NLP applications. Previous research has attempted to reduce the annotation cost by resorting to active learning (e.g. Lewis and Gale (1994)), semi-supervised learning (e.g. Chapelle et al. (2006)), or adaptation from a different domain (e.g. Daumé III and Marcu (2006)). In this thesis, we use crowdsourcing to gather large amounts of data to overcome this bottleneck, as crowdsourcing enables us to do so at a cost that is considerably less than hiring professional annotators.

For example, in Chapter 3 we discuss crowdsourcing a simple labeling task to identify dialectal content in Arabic sentences. The task asked annotators to read Arabic sentences and indicate the level of dialectal content in each sentence, and the particular type of Arabic dialect it is in. While it is not possible to quantify the cost

of this task in a non-crowdsourced setting (as it had never been done before), no such comparison need be made to make it clear our crowdsourced approach is incredibly cost-effective: at a total annotation cost of about $3,000, we collected 330,930 labels – a rate of *less than 1 cent per label*.

In the most complex annotation task discussed in this thesis, crowdsourcing enables us to recreate an Urdu-to-English dataset by delegating the translation task to non-professional translators. Over a period of about a month, we managed to collect over 7,000 translations for under $800. In another, much larger annotation effort to translate dialectal Arabic into English, we crowdsourced the translation of 1.5M Arabic words – enough to train an MT system – at a rate of about $0.03/word, a full order of magnitude less than the cost of hiring a professional translator.

1.1.4 Access to Linguistically Qualified Annotators

In some cases, it is very difficult to obtain annotations of a certain type, as they may require certain domain knowledge. In NLP it is often hard to find annotators with proficiency in certain languages. Crowdsourcing addresses this limiting factor by giving researchers easy access to a worldwide workforce with diverse linguistic skills. This is particularly critical when dealing with widely-spoken languages that are nevertheless considered "rare" languages in terms of currently available datasets. Our tasks for dialect identification and translation involved low-resource languages: dialectal Arabic (the varieties of which are spoken in Middle Eastern countries), and the Urdu language (spoken in India and Pakistan).

Chapter 3 introduces a classification task for which no training data had previously existed, which is the identification of dialectal Arabic sentences and distinguishing them from sentences written in Modern Standard Arabic (MSA, the standardized variant of the Arabic language). The different Arabic varieties differ from each other in non-trivial ways, which necessitates the creation of specialized datasets for each of them. However, due to the prevalence of MSA in written form, almost all Arabic

datasets have predominantly MSA content. Furthermore, due to the fact that the Arabic variants share much of the same vocabulary and use the same character set, it is not an easy task to automatically separate dialectal Arabic content from MSA content, making the creation of dialectal datasets challenging.

We use crowdsourcing to create a large dataset of Arabic sentences, each annotated with a class label reflecting which variety of Arabic it is. This dataset is used to train and evaluate language models for each Arabic variety, and we show that such models are effective for automatic dialect identification. Our models achieve accuracy rates that exceed 85%, significantly outperforming baselines that rely on MSA-only data, which do no better than 70%. The collected data is also used to automatically compile a list of the most dialectal terms, by allowing us to order word types by their *dialectness factor*, which captures how much more likely the word is to appear in a dialectal context. Without our data, creating such an ordering would be extremely challenging, if not impossible.

In this task, crowdsourcing is demonstrated to be an effective method to create a large training set for a task that would have been difficult to approach previously, due to lack of training data. This novel effort relied primarily on native speakers located in Middle Eastern countries (Figure 1.5), illustrating the world-wide reach of crowdsourcing. Similarly, our effort to translate Urdu sentences relied almost exclusively on translators located in India and Pakistan, where Urdu is most widely spoken.

1.2 Main Contributions

The main outcomes of the thesis form a spectrum of theoretical and practical contributions along the four major themes discussed above:

- We implement a wide variety of strategies to exercise **effective quality control**. Our strategies range from simple steps like rendering text into images, to a more complex machine-learning-based method for evaluating workers and their submissions. We also tackle the problem of quality control in the absence

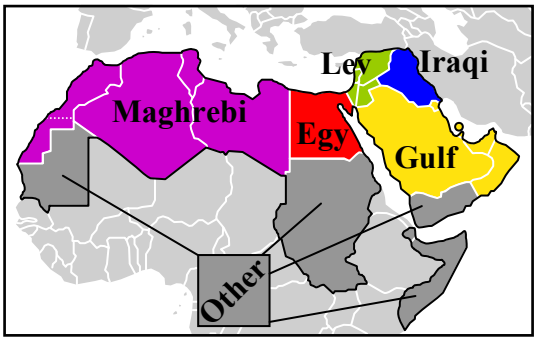


Figure 1.5: The data collection effort of Chapter 3 relied primarily on workers located in Middle Eastern countries, where various dialectal Arabic varieties are spoken. Had crowdsourcing not been available, it would have been very difficult to carry out such an effort.

of gold-standard data.

- We introduce **several new types of annotation tasks** and new models that go with them. We introduce *annotator rationales*, and design two models that incorporate rationales in their training. We introduce a new MT metric, RYPT, based on human judgment of MT quality, and propose strategies for the efficient collection of the judgments.

- We illustrate that a wide variety of NLP annotation tasks can be performed at a **low annotation cost**. The cost of crowdsourcing annotation is invariably a fraction of the cost of hiring and training professional annotators, and the data is often of comparable quality when proper quality control is performed.

- We create **new datasets that had previously not been feasible to create**, due to the access to an international work force. We show that large annotation efforts can yield valuable data resources even for "rare" languages such as Urdu and dialectal Arabic, be they simple sentence-level annotations, or English translations for the purposes of creating a parallel corpus.

In addition, we created several datasets, released to the public whenever possible:

- **The Arabic Online Commentary Dataset**: a 52M-word monolingual corpus rich in dialectal Arabic content, compiled by harvesting reader commentary on articles of three Arabic newspapers.
- **The Dialectal Arabic Dataset**: a set of 108K Arabic sentences from the Arabic Online Commentary Dataset, each annotated for the level and type of dialect in it.
- **The Movie Review Sentiment Polarity Dataset, Enriched with Annotator Rationales**: a version of Pang and Lee's dataset of 2,000 movie reviews, each enriched with annotator rationales supporting the gold-standard label.
- **Translations for the NIST 2009 Urdu–English Evaluation Set**: a recreated version of the NIST dataset, with four crowdsourced English translations for each Urdu sentence (in addition to ten edited versions of those translations, provided by US-based Turkers).
- **Dialectal Arabic-English Parallel Dataset**: a parallel dataset of 180k sentence pairs (1.5M Arabic words; 2.3M English words). The data is not currently public, but will soon be released through the LDC to help facilitate the DARPA BOLT program.

1.3 Thesis Outline

The next Chapter (Chapter 2) introduces the reader to Amazon's Mechanical Turk service, the world's leading crowdsourcing venue. We provide a summary of prior work within computational linguistics that has used the service, as well as a review of prior work on quality control. We also give general guidelines for the design of crowdsourced annotation tasks.

The rest of the thesis is organized as follows:

- Chapter 3 starts with the task of identifying dialectal Arabic sentences and distinguishing them from standard Arabic sentences. We use crowdsourcing

to create a large dataset of Arabic sentences, each annotated with a class label reflecting which variety of Arabic it is. This dataset is used to train and evaluate language models for each Arabic variety, and we show that such models are effective for automatic dialect identification. In this instance, crowdsourcing is demonstrated to be an effective method to create a large training set for a task that was difficult to approach previously (due to lack of training data).

- In Chapter 4 we propose collecting enriched annotations called *annotator rationales*, to complement traditional class labels, and aid learning system parameters that generalize better to unseen data. We demonstrate that crowdsourcing is an effective method to collect a unique type of annotations, and we propose theoretical methods that incorporate such data. We collected rationales and applied our methods to a sentiment analysis task (distinguishing positive movie reviews from negative ones), as well as the dialect identification task, showing significant accuracy improvements in both tasks.

- Chapter 5 deals with the evaluation of MT systems. We show the potential of crowdsourcing to edit MT output, helping recreate HTER scores, which in an evaluation metric measuring the amount of post-editing required to fix MT output. We then present another method for manual evaluation of MT output, which relies on a new evaluation metric. This new metric, RYPT, is based on human judgment of output quality, and is particularly well-suited for a crowdsourced setting. We present methods that enable tuning MT system parameters using this metric, even though it is based on manual evaluation.

- In Chapter 6 we demonstrate that crowdsourcing can be helpful in collecting translations and creating parallel datasets. We recreate an Urdu-to-English evaluation set by crowdsourcing the translation task to non-professional translators, whose submissions are characteristic of speakers of English as a second language. Nonetheless, we discuss a set of features that can help distinguish well-formed translations from those that are not, and effective translators from those who are not. We show that crowdsourcing translation yields results of

near-professional quality at a fraction of the cost of hiring professionals. We also present an effort to collect translations of dialectal Arabic content. The resulting parallel corpus is large enough to train MT systems that outperform systems trained on up to 100 times as much MSA-only data.

1.4 Publications Resulting from this Thesis

Much of the core material (Chapters 3–6) is based on publications at major conferences in the field of computational linguistics. Most of those papers were joint work with Chris Callison-Burch. Other co-authors are called out below where applicable.

Chapter 3 has been submitted as an article to Computational Linguistics, and some of the experiments follow a setup from the AOC paper:

- Zaidan and Callison-Burch (2012). Arabic dialect identification. Computational Linguistics.
- Zaidan and Callison-Burch (2011a). The Arabic Online Commentary Dataset: An annotated dataset of informal Arabic with high dialectal content. In the proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies.

Chapter 4 is based on these two papers:

- Zaidan, Eisner, and Piatko (2007). Using "annotator rationales" to improve machine learning for text categorization. In the proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics.
- Zaidan and Eisner (2008). Modeling annotators: A generative approach to learning from annotator rationales. In the proceedings of the Conference on Empirical Methods on Natural Language Processing.

Chapter 5 expands a 2010 short paper on crowdsourcing editing, and a long 2009 paper on RYPT:

- Zaidan and Callison-Burch (2010). Predicting human-targeted translation edit rate via untrained human annotators. In the proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics.

- Zaidan and Callison-Burch (2009). Feasibility of human-in-the-loop minimum error rate training. In the proceedings of the Conference on Empirical Methods on Natural Language Processing.

Chapter 6 is based on a 2011 paper on crowdsourcing translation. Results in Section 6.5 are based on a 2012 paper on which I am a co-author. My contribution was to help establish the pipeline and methodology for collecting crowdsourced translations. The relevant publications are:

- Zaidan and Callison-Burch (2011b). Crowdsourcing translation: Professional quality from non-professionals. In the proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies.

- Zbib, Malchiodi, Devlin, Stallard, Matsoukas, Schwartz, Makhoul, Zaidan, and Callison-Burch (2012). Machine translation of Arabic dialects. In the proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

Chapter 2

Crowdsourcing and Amazon's Mechanical Turk

Crowdsourcing is the delegation of a particular task to a large group of individuals rather than a single person. The term, a combination of the words "crowd" and "outsourcing", was first used by Jeff Howe of *Wired* magazine (Howe, 2006), to refer to companies taking a function once performed by employees or hired professionals, and outsourcing it to a large group of individuals, particularly when facilitated by online tools and venues.[1]

In this Chapter, we present Amazon's Mechanical Turk service (MTurk). MTurk is the world's leading crowdsourcing venue, and all our crowdsourced data is gathered though it. We will define relevant terminology, and provide a literature review of NLP-oriented crowdsourcing publications, as well as research on exercising effective quality control over noisy submissions. We also discuss the main advantages of using the service, and provide a rundown of recommended practices that can ensure that

[1]There is some overlap between crowdsourcing and each of *social computing* and *human computation*, but the terms are not always synonymous. Social computing includes a strong social component, and its purpose is not usually to perform a computation or an automated task (Dryer et al., 1999; Parameswaran and Whinston, 2007). Human computation is more closely related to crowdsourcing, where human processing power replaces that of computers' (usually in tasks that computers are not able to solve yet), whereas crowdsourcing replaces human workers in a conventional setting with many individuals of the crowd (von Ahn, 2005; Law and von Ahn, 2011). See Quinn and Bederson (2011) for a more detailed survey of those and other related terms.

data quality and data throughput are maximized. Those advantages and practices will resurface again and again throughout the thesis, as we see how they are applied and observed in a variety of annotation tasks that range in complexity and required linguistic skills.

2.1 Amazon's Mechanical Turk Service

To collect crowdsourced translations, we use Amazon's Mechanical Turk (MTurk), an online marketplace designed to pay people small sums of money to complete *Human Intelligence Tasks* (or HITs), the smallest unit of work on MTurk. Tasks on MTurk range from labeling images to moderating blog comments to providing feedback on relevance of results for search queries. According to the MTurk HIT count, there are several hundred thousand HITs available at any given time (Figure 2.1). MTurk was launched publicly in November 2005, as one of Amazon's suite of Web Services. The original "Mechanical Turk" was a late 18th-century hoax, which appeared to be a chess-playing automaton but was in fact controlled by a (human) chess player hiding inside it. This name is used for Amazon's service because it facilitates hiring humans to perform tasks that machines are not very good at; Amazon's tagline for MTurk is "artificial artificial intelligence."

Anyone with an Amazon account can either submit HITs to be completed, or work on HITs that were submitted by others. Workers are referred to as "Turkers", and those submitting HITs to be completed are referred to as "Requesters." A Requester specifies the reward to be paid for each completed item, sometimes as low as $0.01. Turkers are free to select whichever HITs interest them, and to bypass HITs they find uninteresting or which they deem pay too little.

The advantages of MTurk include:

- **A large and low-cost labor force.** As of 2007, Amazon reported there are more than 100,000 Turkers on MTurk (Pontin, 2007). While the figure is likely to include Turkers who are not very active in the system, MTurk is by far the world's largest and most recognizable crowdsourcing venue.

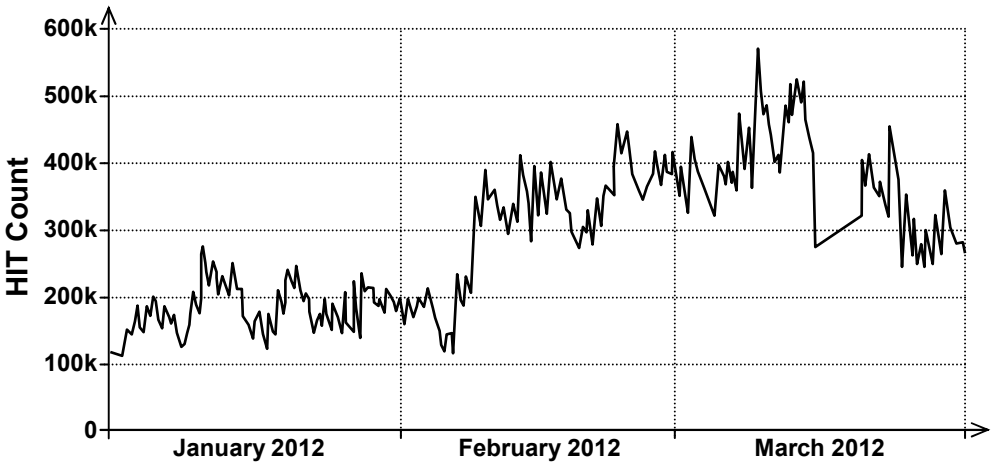


Figure 2.1: MTurk's HIT count over the first three months of 2012. The chart is based on data from www.mturk-tracker.com, which tracks, among other things, the HIT count reported by Amazon.

- **Little overhead for hiring workers.** A Requester need not incur costs to advertise their tasks to attract workers, nor do they need to get involved in designing the infrastructure that allows tracking and monitoring workers' progress. Furthermore, Requesters are not obligated to pay Turkers until after submitted work is reviewed.

- **An easy micropayment system.** All Turkers on MTurk have the option of being paid in the form of Amazon credit, which can be used to purchase merchandise from Amazon.com. Turkers in the United States (and, more recently, in India) have the option of transferring their earnings directly into their bank accounts in the form of local currencies.

- **Short turnaround time.** As tasks get completed *in parallel* by a large number of individuals, it is possible to achieve data throughput at a much higher rate than via traditional annotation efforts.

- **Access to foreign markets.** The low entry barrier for Turkers puts foreign markets on the table as an option, giving us direct access to workers in such markets, and hence much improved odds of finding native speakers of languages

that are not widely spoken in the US.

One possible downside is that Amazon does not provide any information about Turkers. (Each Turker is identifiable only through an anonymous ID.[2]) In particular, no information is available (nor collected) about a Turker's educational background, skills, or even native language(s). This makes it difficult to determine if a Turker is qualified to complete a particular task, though it is possible for Requesters to explicitly ask annotators to volunteer such information.[3]

2.1.1 The Demographics of Mechanical Turk

Ipeirotis (2010a) conducted a survey in February 2010, paying Turkers on MTurk $0.10 for participating and providing information about their background and their experience on MTurk. The survey found that US-based individuals formed a plurality of Turkers, at 47% of participants, followed by Indian-based Turkers, at 34% of participants. Turkers were found to be, in general, younger and more educated than the general public. An interesting outcome of the answers is that the profile of US-based Turkers differed from the profile of Indian-based Turkers: US Turkers were more likely to be female (65% vs. 30%) and older (41% over 35 vs. 13%).

Turkers indicated that they completed tasks on MTurk for a variety of reasons, such as to pass the time, for fun, or out of interest in the tasks themselves. The major motivation, though, is that Turkers consider MTurk to be a source of income. Some Turkers, often located in India, consider MTurk to be a *primary* source of income. That said, most Turkers spend under eight hours per week on MTurk, and do not end up earning enough to consider MTurk a consistent and reliable source of income (Rose, 2009).

There is a lot of variety on the Requester side of MTurk as well, as can be seen by examining the tasks posted to the marketplace. Tasks ask Turkers to collect

[2]On the other hand, this anonymization makes MTurk an attractive option for e.g. university researchers, who are usually required to ensure that their subjects remain anonymous in order to obtain approval from institutional review boards.

[3]As long as such information could not be used to identify the Turker, such as their name or e-mail address.

contact information for businesses, perform image labeling to aid object recognition, participate in surveys and questionnaires, and perform transcription of audio clips. Unfortunately, there are many spammy Requesters as well (Ipeirotis, 2010b). They post tasks rewarding positive voting on a particular Facebook page or YouTube video, fake account creation, and fake ad-click generation. All such HITs are against MTurk's terms of service.[4]

2.2 Literature Review

The research community has shown strong interest in MTurk as an option for data collection, and a plethora of papers were published in the last few years detailing how annotations were collected to aid learning in a variety of NLP tasks. In this Section, we give an overview of prior work that used crowdsourcing for NLP-related annotation efforts. We then give a review of prior work for quality control on MTurk.

2.2.1 Annotation for NLP Tasks on MTurk

Snow et al. (2008) were the first to use MTurk to obtain data for several NLP tasks, such as textual entailment and word sense disambiguation. Hsueh et al. (2009) asked Turkers to provide sentiment classifications for political blogs, and defined several measures to measure annotation quality. Mihalcea and Strapparava (2009) had Turkers compose short paragraphs describing their opinions on topics such as abortion and the death penalty. Chang et al. (2009) carried out human evaluations on MTurk to measure the quality of automatically inferred topics. Their tasks captured aspects of topic modelling that existing metrics cannot express adequately, and they found that certain models could be judged to be better (by human annotators) under their evaluation design, but not under an automatic metric such as predictive perplexity. Kittur et al. (2008) asked Turkers to evaluate Wikipedia articles along several dimensions, such as objectivity and writing quality.

[4]More examples of violating HITs can be found here: https://www.mturk.com/mturk/help?helpPage=policies .

In the domain of speech recognition, Marge et al. (2010) solicited speech transcriptions and showed that professional quality can be achieved by reconciling non-professional transcriptions from multiple Turkers. Novotney and Callison-Burch (2010) collected transcriptions of conversational speech and achieved professional performance by simply collecting more data, rather than attempting to focus on improving data quality. McGraw et al. (2009) focused on the task of individual word recognition, and asked Turkers to listen to short audio clips of individual words, and then choose a word from a short list of candidates.

Callison-Burch (2009) ventured beyond simple labeling tasks, and showed that Turkers could accomplish more complex tasks like providing translations and composing questions for reading comprehension tests. Kaisser and Lowe (2008) created a corpus of question-answer sentence pairs, by presenting Turkers with questions and a text, and asking them to find the answers from that text. Also in the QA domain, Mrozinski et al. (2008) used MTurk to create a corpus of *Why*-questions and corresponding answers. Rosenthal et al. (2010) crowdsourced a task for PP attachment in sentences taken from the Wall Street Journal, and Jha et al. (2010) investigated the same task but for genres other than news. Nakov (2008) created a corpus of interpretations of noun-noun compounds (e.g. "tear gas" and "apple cake") to aid in understanding the semantics of such compounds. Rashtchian et al. (2010) ask Turkers to annotate images by writing descriptive sentences of them (e.g. "Two men playing cards at a table" and "The two men are in an intense card game"). There are several other papers that crowdsource image labeling (e.g. Sorokin and Forsyth (2008); Deng et al. (2009)), though the annotations are usually non-linguistic in nature.

Gillick and Liu (2010) showed that evaluation of summaries is difficult to achieve on MTurk, as Turkers' judgments were not useful in replicating system rankings given by expert judges. On the other hand, Munro et al. (2010) gave a summary of several projects that are using crowdsourcing technologies in novel ways to conduct psycholinguistic studies. They find that crowdsourced results are indistinguishable from those of more controlled laboratory experiments.

Several of the above papers appeared in a NAACL 2010 workshop devoted ex-

clusively to creating data resources for NLP tasks via MTurk (Callison-Burch and Dredze, 2010). Two other workshops concerning crowdsourcing, focusing on translation in particular, were held at the University of Maryland (Bederson and Resnik, 2010), and at the 2010 meeting of the Association for Machine Translation in the Americas (Désilets, 2010). The goal was to facilitate discussion among a group of individuals with various backgrounds, computational and professional, regarding crowdsourcing efforts, future directions, and the influence of crowdsourcing on the providers and users of translation services.

2.2.2 Quality Control on MTurk

Majority voting is the simplest form of quality control on MTurk. Snow et al. (2008), the first to use MTurk for NLP annotation, relied on majority voting, though they also proposed a component for annotator bias correction. Their results showed that a few non-expert labels usually suffice. Sheng et al. (2008) also relied on majority voting to improve data quality, as they applied it to a number of datasets from the UCI Repository (Frank and Asuncion, 2010).

Dawid and Skene (1979) developed an EM-based algorithm for evaluating labelers by inferring their error rates in the absence of gold-standard labels. Smyth et al. (1994) take a similar approach, and apply it in an image labeling task. One disadvantage of both works is that they assume all annotators label the same set of examples. Raykar et al. (2010) took a more Bayesian approach to solving the same problem, proposing a framework that also gives an estimate of the hidden labels. Jin and Ghahramani (2003) investigated the scenario where each training example is annotated with multiple labels, but only one of which is correct, and presented a discriminative method to tackle the problem.

Dekel and Shamir (2009a,b) also dealt with noisy labels but placed more emphasis on a crowdsourced setting than the work above. In crowdsourcing, the number of annotators scales up with the number of labeled examples, resulting in only few labels for most annotators, and making it difficult to accurately assess annotation quality for an annotator. They presented two data-cleaning algorithms that rely on aggregating

all annotators' data in order to identify the ones not doing the task properly. One advantage of their framework is that it does not assume that a training example has multiple redundant judgments, nor that annotators'

Whitehill et al. (2009) proposed a probabilistic model to filter labels from non-experts, in the context of an image labeling task. Their system generatively models image difficulty, as well as noisy, even adversarial, annotators. They apply their method to simulated labels rather than real-life labels. Donmez et al. (2009) not only estimated the quality of annotators' work but also learned which training examples are best to annotate next, taking an active learning approach to the problem. Ipeirotis et al. (2010) distinguished between annotator bias and annotator error, and presented an algorithm that is able to quantify each separately. This allowed them to correct for annotator bias, which makes even biased annotators helpful.

Rashtchian et al. (2010) crowdsourced an image annotation task, asking annotators to compose descriptive sentences of them. As an example of free-form text entry, quality control is more difficult to perform adequately than in simple labeling tasks. They found that pre-screening Turkers via a qualification test (of grammar and spelling skills) was a very effective method. Kittur et al. (2008) employed an interesting quality control mechanism by giving Turkers the *illusion* that their answers will be checked by a human reviewer. They included questions that could in theory be verifiable, but are actually not verified. This was sufficient to substantially improve the quality of obtained data and greatly reduced spammy submissions.

2.3 MTurk Best Practices

A Requester should take care to design the annotation interface so as to maximize the amount of data collected, and at the same time ensure that the task is being performed properly and effectively. It is essential that Requesters try out their HITs themselves, and it is recommended to ask a non-NLP person to perform it as well. Requesters should also take measures to ensure that the collected data is of high quality, and that spammy behavior is easy to detect. We review here some common

accepted practices when designing annotation interfaces.

Short, simple instructions. The MTurk marketplace contains thousands of tasks, and Requesters are, at least to some degree, in competition with each other for Turkers' time. For that reason, the task's instructions should be short and simple, and should highlight only the most important concepts needed to perform the task. Ideally, the instructions would be centered around actual examples that illustrate different instances of the problem. If possible and when appropriate, visual illustrations are highly recommended, as they communicate information effectively, and also attract the attention of Turkers.

User-friendly annotation interfaces. Many of the tasks on MTurk could become boring and tiresome for the annotator. Often, this is inherent to the annotation task itself, which ceases to be interesting after completing a number of HITs. Requesters should attempt to mitigate at least the mechanical and physical strain on the annotator, by designing user-friendly annotation interfaces. For example, an interface for textual input (e.g. translation) that is easy to navigate using the tab key is superior to an interface that requires a combination of keyboard and mouse use to move between items of the HIT and submit it.

Embedded control items. It is highly recommended to allocate a small subset of the data to be annotated, and have it annotated by a professional annotator (or the Requester themselves). This set of examples and their gold-standard labels can be used to effectively evaluate Turkers as follows. A small portion (10%–20%) of the data to be annotated on MTurk should be sampled from this set of professionally-annotated data, and embedded randomly within the items in any given HIT. Control items should be indistinguishable from other items to the Turker. Since they are known to the Requester, and since the correct answers are known beforehand, this makes direct evaluation of a Turker's quality of work much easier.

Collecting redundant answers. MTurk gives a Requester the option to have a HIT performed by more than one Turker. Therefore, it is possible to collect multiple labels for the same item and then take a majority vote over them, allowing us to be more confident of the correctness of the label. This is particularly effective if the task

involves picking one of a small set of labels, and 3 to 5 redundant labels are usually adequate. Also, collecting redundant judgments would easily provide sufficient data to allow Requesters to measure and report inter-annotator agreement.

MTurk's *Qualifications*. MTurk has a number of built-in mechanisms for quality control. A Requester can specify a number of *Qualifications* that a Worker must meet in order to be allowed to work on the Requester's HITs. For example, Requesters can specify that a Worker have a minimum approval rating, say 90%, on all previously submitted HITs. Requesters can also specify that a Worker must have completed a minimum number of HITs.[5] Requesters can also design their own Qualification Tests, which assign scores to Workers based on how well they answer a number of multiple-choice questions supplied by the Requester. Another useful Qualification is the Country Qualification, which allows the Requester to require that Workers be present in a particular country (or be outside a particular country). One limitation of MTurk's Qualifications is that a Requester cannot specify a logical disjunction of them – a Worker is qualified for a HIT only if they meet *all* the specified Qualifications.[6]

Rendering text as images. Using image versions of textual input is recommended, as that ensures a consistent view across users' machines, and allows us to optimize for an easily readable font and width chosen to aid reading. More importantly, this would guard against cheating in cases where annotators would be able to copy the text and paste it as input to an automated system, such as a machine translation system. If generating images is difficult, it is possible to make the text un-selectable using a simple HTML function.[7]

Use of native (non-English) language. When the target pool of Turkers are

[5]This could be useful for excluding newly registered Workers, who start out with an "approval rating" of 100%.

[6]For example, a Requester can specify that a Worker must be located outside **all** of {India, Pakistan, Russia}, by requiring the Worker must meet the three Country Qualifications `country≠India`, `country≠Pakistan`, and `country≠Russia`. However, there is no easy way to specify that a Worker be present in **any one** of {USA, UK, Canada}. Such a Qualification would be very useful for HITs that require native speakers of English. In such cases, a Requester could create three separate batches, one for each country, but that is not ideal for several reasons.

[7]The text would still appear in the HTML source code. Using images instead of text completely eliminates access to sentences in textual form.

not native speakers of English, it is recommended to use terminology in that language throughout the interface. This would provide the Turker with confirmation that their understanding of the task is correct. This also plays a role in attracting those Turkers, since they would likely be more interested in the task if it contains content in their native language.

Non-monetary motivation. The wording and presentation of the task could provide motivation for the Turker besides the monetary reward, such as explaining the Requester's own research and motivation for creating the task, and how the collected data will be used. In such cases, a Turker could feel they are part of a larger group effort that seeks to improve current technology, which could make the task more enjoyable and bearable.

Embedding code for the Geolocation plugin. Since no information exists about Turkers besides an anonymous ID, it is recommended to use the Geolocation plugin[8], which extracts information about the annotator, such as their location (city and country), IP address, and browser language. Such information is indicative of a Turker's native language, and could therefore help identify those who are not qualified to perform a task that requires fluency in a particular language.

Interaction with Turkers. Many Turkers are eager to provide comments and feedback to Requesters regarding the tasks they upload to MTurk. It is therefore advisable to open a channel of communication with the Turkers, and respond to their questions and suggestions. In the same vein, it is critical that Requesters review submitted work promptly, as that builds trust with the Turkers and assures Turkers that their efforts will not go unrewarded.

2.4 Typical Turker Behavior and MTurk Sampling

It is worth pointing out that, although crowdsourcing delegates a task to a large number of people, in reality much of the work is performed by a (relatively) small set

[8] http://www.geoplugin.com/webservices/javascript

of Turkers. As an example, take the data collected for the Arabic dialect identification task of Chapter 3. Figure 2.2 illustrates that most of the data came from a few prolific Turkers. For example, the three most active Turkers contribute slightly more than half the data, and about 75% of the data is provided by the top seven Turkers.

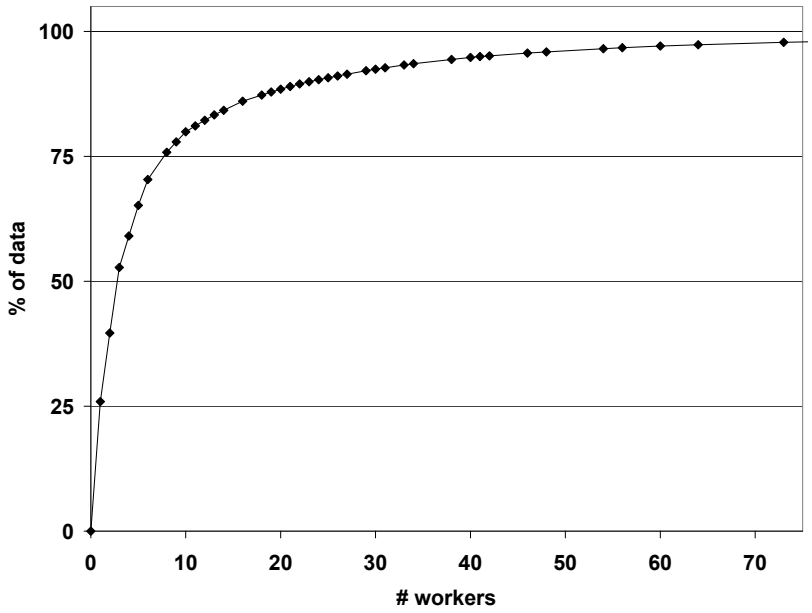


Figure 2.2: The percentage of the approved data contributed by the x most prolific Turkers.

Another aspect of MTurk worth pointing out is the order and sampling of HIT's from a batch of HIT's. We will again consider the data collected for the dialect ID task. The size of the batch uploaded for that task was quite large, and consisted of over 10,000 HITs. Each of those HITs was requested to be completed by *three distinct annotators*. Therefore, each HIT had three assignments associated with it, to collect redundant judgments for each one.

In what order were these HITs presented to Turkers? Examining the order in which HITs were selected to be completed by Turkers gives an insight into the sampling process of MTurk. Rather than selecting HITs uniformly from the entire pool of HITs, the strategy is geared toward getting HITs performed to completion rather than getting a new HIT answered. This is indicated by the fact that most submitted

assignments at any point in time come from HITs that were performed to completion (Figure 2.3).

The selection order clarifies how the sampling is geared to achieving that (Figure 2.4), in that it indicates that sampling is uniform but *within a certain window* of the set of HITs at any point in time. For instance, this window seems to be the first 600 HITs or so initially, slowly moving higher as more and more HITs are performed to completion. By following this strategy, MTurk ensures that Requesters would have complete data for a relatively small number of HITs, rather than have incomplete data for a relatively large number of HITs.

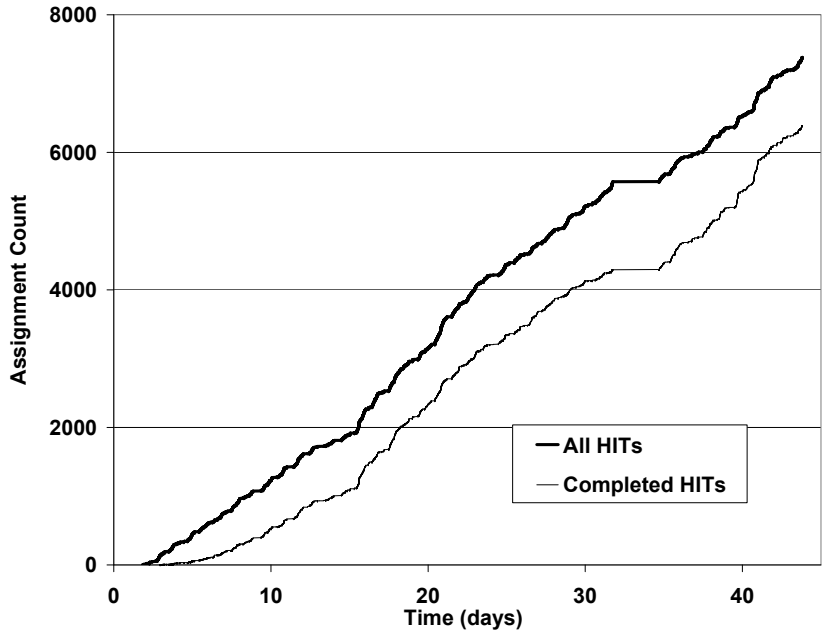


Figure 2.3: The number of assignments completed from all HITs (top curve), and the portion of assignments completed from completed HITs only (bottom curve), during the first 50 days of a data collection effort for Arabic dialect identification (Chapter 3). Three assignments were requested for each HIT, and therefore, if sampling were uniform over the full set (14k HITs), the bottom curve would be much lower.

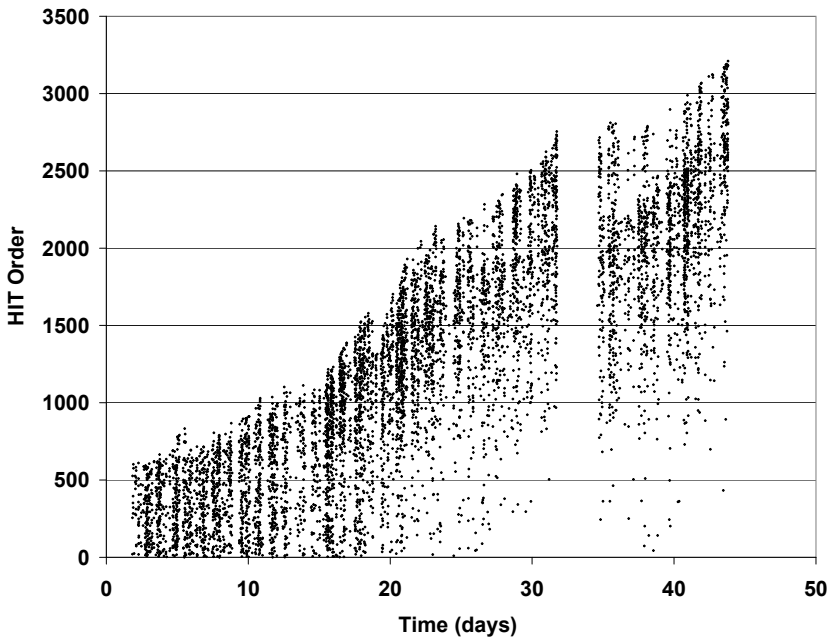


Figure 2.4: The sampling order of the HITs during the same data collection effort of Figure 2.3, where the *order* of a HIT is the order in which it appeared in the MTurk input file when the task was created. The gap around days 32–34 is due to the task being inactive after expiring, before this inactivity was realized (and the task extended) on day 34.

2.5 Conclusion

In this Chapter, we introduced crowdsourcing and Amazon's Mechanical Turk service (MTurk), the world's leading crowdsourcing venue. We also provided an overview of prior work related to crowdsourcing, focusing on NLP-oriented work, as well as more general research on quality control. We included a set of guidelines that reflect our experience in crowdsourcing the various annotation tasks of this thesis, which could help other researchers ensure a high throughput of high-quality data. Finally, we made note of some typical characteristics of data collection efforts, and discussed MTurk's sampling patterns for large batches.

Crowdsourcing comes with its own set of advantages and challenges. In this Chapter, we provided an overview of the advantages (in 2.1), as well as a set of best practices to overcome the challenges (in 2.3). Throughout this thesis, we will come across several 'instantiations' of those advantages and challenges, as we crowdsource a variety of annotation tasks, from simple labeling, to complete sentence composition. In the next Chapter, we crowdsource a labeling task to identify dialectal content in Arabic sentences. This effort is a perfect illustration of every single one of our recommended best practices: we use simple instructions with Arabic terminology and a user-friendly interface, we embed non-dialectal sentences as control items, we use Geolocation information to exclude many spammers, and we motivate Turkers by telling them about our research and interacting with them. The task also illustrates MTurk's main advantages: it gave us immediate access to an international workforce that includes many native speakers of Arabic, and allowed us to build the first dataset of its kind quickly and affordably.

Chapter 3

Arabic Dialect Identification

The written form of the Arabic language, *Modern Standard Arabic* (MSA), differs in a non-trivial manner from the various spoken regional dialects of Arabic – the true "native languages" of Arabic speakers. Those dialects, in turn, differ quite a bit from each other. However, due to MSA's prevalence in written form, almost all Arabic datasets have predominantly MSA content.

In this Chapter, we discuss the creation of an Arabic dataset with dialect annotations. We present the *Arabic Online Commentary Dataset*, a 52M-word monolingual dataset rich in dialectal Arabic content, and describe our annotation effort to identify the dialect level (and dialect itself) in each sentence of the dataset. Given this new annotated dataset, we investigate the problem of Arabic dialect identification: given the word sequence forming an Arabic sentence, determine the variety of Arabic in which it is written.[1]

The data discussed in this Chapter consists of annotations that require a particular linguistic skill, namely the ability to understand dialectal varieties of Arabic. Native speakers of Arabic, whether dialectal or MSA, are difficult to find outside of the Middle East. Therefore, this Chapter showcases crowdsourcing's added benefit by giving researchers access to annotators from across the world, and highlights the possibility of gathering large amounts of data without face-to-face interaction with

[1]This Chapter is mostly recent work, but it has been submitted as an article to Computational Linguistics (Zaidan and Callison-Burch, 2012). Some of the experimental results in this Chapter are based on Zaidan and Callison-Burch (2011a)

annotators. Our annotation setup also demonstrates how and why user interfaces and annotation setups should be easy to use and should, at least to some degree, make the annotation task fun to perform.

The collected labels are also quite interesting in that they shed light on somewhat unanticipated patterns and biases in annotators' behavior. Also, since this is the first effort of its kind, this Chapter raises interesting questions about how we can perform quality control in the absence of any gold-standard data.

The Chapter starts with an introduction to the various Arabic dialects, and what makes Arabic dialect identification a difficult problem. We then discuss the dialect annotation and crowdsourcing the task, examining annotator behavior and several types of observed human annotator biases. The newly created dataset is used to train and evaluate statistical models for dialect identification, a task that we explore in detail.

3.1 Background: The MSA/Dialect Distinction in Arabic

The Arabic language, with an official status in over 20 countries and spoken by more than 250 million people, is a loose term that refers to many existing varieties. Arabic is characterized by an interesting linguistic dichotomy: the written form of the language, *Modern Standard Arabic* (MSA), differs in a non-trivial fashion from the various *spoken varieties* of Arabic, each of which is a regional dialect (or a *lahjah*, *lit.* dialect; also *darjah*, *lit.* common). MSA is the only variety that is standardized, regulated, and taught in schools, necessitated because of its use in written communication in formal venues.[2] The regional dialects, used primarily for day-to-day dealings and spoken communication, are not taught formally in schools, and remain somewhat absent from traditional, and certainly official, written communication.

[2]The term "MSA" is used primarily by linguists and in educational settings. For example, constitutions of countries where Arabic is an official language simply refer to "The Arabic Language," the reference to the standard form of Arabic being implicit.

These dialects do not have an explicit set of grammar rules, but there is certainly a concept of *grammatical* and *ungrammatical*. Furthermore, even though they are 'spoken' varieties, it is certainly possible to produce dialectal Arabic *text*, by phonetically spelling out words using the same letters used in MSA. Thanks to phonetic spelling, and in spite of the lack of a "correct" spelling for many dialectal words, there is usually little ambiguity and a high rate of agreement regarding how a word would be spelled. Therefore, from a scientific point of view, the dialects can be considered separate languages in their own right, much like North Germanic languages (Norwegian/Swedish/Danish) and West Slavic languages (Czech/Slovak/Polish).[3]

There is a reasonable level of mutual intelligibility across the dialects, but the extent to which a particular individual is able to understand other dialects depends heavily on that person's own dialect and their exposure to Arab culture and literature from outside of their own country. For example, the typical Arabic speaker has little trouble understanding the Egyptian dialect, thanks in no small part to Egypt's history in movie-making and television show production, and their popularity across the Arab world. On the other hand, the Moroccan dialect, especially in its spoken form, is quite difficult to understand by a Levantine speaker.

A certain amount of code-switching occurs between MSA and dialectal Arabic, though most of it occurs at sentence boundaries. When it does occur intra-sententially, one common pattern is at the beginning of a noun phrase, i.e. choosing to use the MSA version of a term when speaking dialectally. (MSA-to-dialect code-switching could occur as well.) It should be noted that many pronouns and prepositions in Arabic are attached to the corresponding nouns. When code switching occurs, it covers those attached pronouns/prepositions as well.

[3]Note that such a view is not widely accepted by Arabic speakers, who hold MSA in high regard. They consider dialects, including their own, to be simply imperfect, corrupted versions of MSA, rather than separate languages. A notable exception might be the Egyptian dialect, where a nationalistic movement gave rise to such phenomena as the Egyptian Wikipedia, where articles are written exclusively in Egyptian, and little, if any, MSA exists.

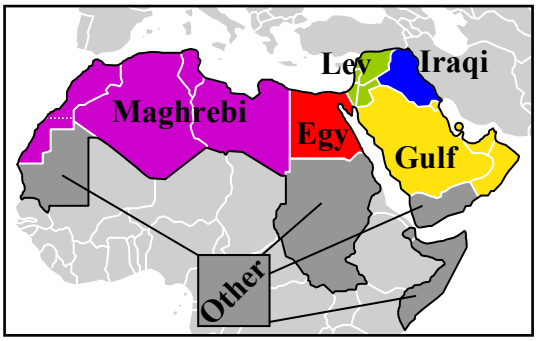


Figure 3.1: One possible breakdown of spoken Arabic into dialect groups: Maghrebi, Egyptian, Levantine, Gulf, and Iraqi. Habash (2010) gives a breakdown along mostly the same lines. We used this map as an illustration for annotators in our dialect classification annotation task, with Arabic names for the dialects instead of English.

3.1.1 The Dialectal Varieties of Arabic

One possible breakdown of regional dialects into main groups is as follows (see Figure 3.1):

- **Egyptian**: the most widely understood dialect, due to a thriving Egyptian television and movie industry, and Egypt's highly influential role in the region for much of 20th century.
- **Levantine**: a set of dialects that differ somewhat in pronunciation and intonation, but are largely equivalent in written form; closely related to Aramaic.
- **Gulf**: arguably the closest of the regional dialect to MSA, particularly in verb conjugation.
- **Iraqi**: sometimes considered to be one of the Gulf dialects, though it has distinctive features of its own in the use of prepositions and verb conjugation. The differences are more pronounced in spoken form.
- **Maghrebi**: heavily influenced by the French and Berber languages, and could be unintelligible by speakers of other dialects, especially in spoken form.

We emphasize here that there are no sharp boundaries between dialect groups as might be understood from Figure 3.1. Rather, the transition is smooth and fluid, and our map merely shows the main dialect groups. For example, there are differences between the Egyptian spoken in Cairo and the Egyptian spoken in Sinai (which is a variety of Egyptian closer to Gulf than the Cairene variety).[4]

There are a large number of linguistic differences between MSA and the regional dialects. Some of those differences are on the level of short vowels, which are omitted in Arabic text, and therefore would not appear in written form. That said, many differences do manifest themselves textually:

- MSA's morphology is relatively richer than dialects. For instance, MSA has a *dual* form in addition to the singular and plural forms, whereas the dialects mostly lack the dual form. Also, MSA has two plural forms, one masculine and one feminine, whereas the dialects often make no such gendered distinction.[5]

- Dialects lack grammatical case, while MSA has a complex case system. In MSA, most cases are expressed with diacritics that are rarely explicitly written, with the accusative case being a notable exception, as it is expressed using a suffix ($+A$) in addition to a diacritic (e.g. on objects and adverbs).

- MSA has certain function words with no direct equivalents in the dialects, such as *lqd* for introducing declarative sentences and *hl* for yes/no questions.[6]

- There are lexical choices differences in the vocabulary itself, especially for non-nouns. Table 3.1 gives several examples.

- Differences in verb conjugation, even when the triliteral root is preserved. See the lower part of Table 3.1 for some conjugations of the root *$-r-b* (to drink).

[4]People from other regions of Egypt may choose to revert to the more 'standard' Cairene Egyptian to ensure they are understood by as many people as possible.

[5]Dialects may preserve the dual form for nouns, but mostly lack it in verb conjugation and pronouns, using plural forms instead. The same is true for the gendered plural forms, which exist for many nouns (e.g. 'teachers' is either *mElmyn* (male) or *mElmAt* (female)), but not frequently used otherwise.

[6]We use the Buckwalter transliteration scheme to represent Arabic orthography, which maps each Arabic letter to a single, distinct ASCII character. See Appendix A for the character mapping.

English	MSA	EGY	LEV	GLF
Book	*ktAb*	*ktAb*	*ktAb*	*ktAb*
Year	*snp*	*snp*	*snp*	*snp*
Money	*nqwd*	*flws*	*mSAry*	*flws*
Come on!	*hyA!*	*ylA!*	*ylA!*	*ylA!*
I want	*Aryd*	*EAyz*	*bdy*	*AbgY*
Now	*Al/n*	*dlwqt*	*hlq*	*AlHyn*
When?	*mtY?*	*AmtY?*	*AymtY?*	*mtY?*
What?	*mA*A?*	*Ayh?*	*Ay$?*	*w$?*
I drink	*A$rb*	*b$rb*	*b$rb*	*A$rb*
He drinks	*y$rb*	*by$rb*	*b$rb*	*y$rb*
We drink	*n$rb*	*bn$rb*	*bn$rb*	*n$rb*

Table 3.1: A few examples illustrating the differences across MSA and three Arabic dialects: Egyptian, Levantine, and Gulf. (The Buckwalter transliteration scheme is used – see Appendix A.) Even when a word is spelled the same across two or more varieties, the pronunciation might differ due to differences in short vowels (which are not spelled out).

The above list, and Table 3.1, deal with differences that are expressed at the individual-word level. It is important to note that Arabic varieties differ markedly in style and sentence composition as well. For instance, all varieties of Arabic, MSA and otherwise, allow both SVO and VSO word orders, but MSA has a higher incidence of VSO sentences than dialects do.

3.1.2 Existing Arabic Data Sources

Despite the fact that speakers are usually less comfortable communicating in MSA than in their own dialect, MSA content significantly dominates dialectal content, as MSA is the variant of choice for formal and official communication. Relatively little printed material exists in local dialects, such as folkloric literature and some modern poetry, but the vast majority of published Arabic is in MSA. As a result, MSA's dominance is also apparent in datasets available for linguistic research. The problem is somewhat mitigated in the speech domain, since dialectal data exists in the form of phone conversations and television program recordings, but, in general, dialectal Arabic datasets are hard to come by.

The abundance of MSA data has greatly aided research on computational methods applied to Arabic, but only the MSA variant of it. A state-of-the-art Arabic-to-English machine translation system performs quite well when translating MSA source sentences, but often produces incomprehensible output when the input is dialectal. For example, most words of the dialectal sentence shown in Figure 3.2 are transliterated, whereas an equivalent MSA sentence is handled quite well. The high transliteration rate is somewhat alarming, as the first two words of the dialectal sentence are relatively frequent: *AymtY* means 'when' and *rH* corresponds to the modal 'will'. Granted, it is conceivable that processing dialectal content is more difficult than MSA, but the main problem is the lack of dialectal training data.[7]

This is an important point to take into consideration, since the dialects differ to a large enough extent to warrant treating them as more or less different languages.

[7]It can in fact be argued that, since MSA is the variant with the more complex sentence structure and richer morphology, *it* is the more 'difficult' variant to process and translate.

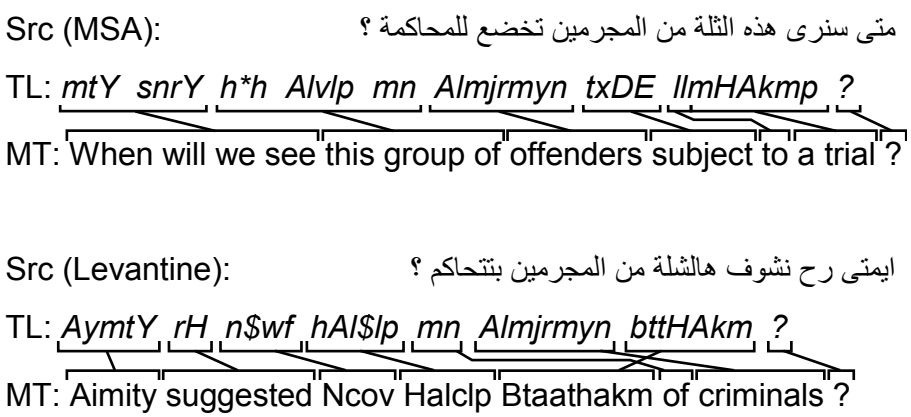


Figure 3.2: Two roughly equivalent Arabic sentences, one in MSA and one in Levantine Arabic, translated by the same MT system (Google Translate) into English. An acceptable translation would be *When will we see this group of criminals undergo trial* (or *tried*)*?*. The MSA variant is handled well, while the dialectal variant is mostly transliterated.

Attempting to translate *dialectal* Arabic using an MT system trained on **MSA** data is similar to using a **Spanish**-to-English system to translate a *Portuguese* sentence. In fact, doing just that (Figure 3.3) yields very similar behavior to that seen in the Arabic mismatch example.

This example illustrates the need for dialectal data, to train MT systems to handle dialectal content properly. A similar scenario would arise with many other NLP tasks, such as parsing or speech recognition, where dialectal content would be needed in large quantities for adequate training. A robust dialect identifier could sift through immense volumes of Arabic text, and separate out dialectal content from MSA content.

3.1.3 The Arabic Online Commentary Dataset

One domain of written communication in which MSA and dialectal Arabic are both commonly used is the online domain, since it is more individual-driven and less institutionalized than other venues. This makes a dialect much more likely to be the user's language of choice, and dialectal Arabic has a strong presence in blogs,

Spanish-English System:

Src: **Quando** veremos **esse** grupo de criminosos **serem julgados** ?

MT: **Quando esse** group of criminals see **Serem julgados** ?

Portuguese-English System:

Src: Quando veremos esse grupo de criminosos serem julgados ?

MT: When will we see this group of criminals to be judged ?

Figure 3.3: The output of a Spanish-to-English system when given a Portuguese sentence as input, compared to the output of a Portuguese-to-English system, which performs well. The behavior is very similar to that in Figure 3.2, in particular the pervasive transliteration of words when there is a language mismatch.

forums, chat rooms, and user/reader commentary. Therefore, online data is a valuable resource of dialectal Arabic text, and harvesting this data is a viable option for computational linguists for purposes of creating large datasets to be used in statistical learning.

In that spirit, the *Arabic Online Commentary* Dataset (AOC) (Zaidan and Callison-Burch, 2011a) is a 52M-word monolingual dataset created by harvesting reader commentary from the online versions of three Arabic newspapers (Table 3.2). The data is characterized by the prevalence of dialectal Arabic alongside MSA. The most common dialects correspond to the respective countries of publications: ***Al-Ghad*** is published in Jordan (primary dialect: Levantine), ***Al-Riyadh*** is published in Saudi Arabia (Gulf), and ***Al-Youm Al-Sabe'*** is published in Egypt (Egyptian).[8]

While a significant portion of the AOC's content is dialectal, there is still a very large portion of it that is in MSA. (Later analysis in 3.3.2.1 shows dialectal content is roughly 40%.) In order to take full advantage of the AOC (and other Arabic datasets with at least some dialectal content), it is desirable to separate dialectal content from non-dialectal content automatically. The task of dialect identification (and its automation) is the focus for the remainder of this Chapter.

[8]URLs: www.alghad.com, www.alriyadh.com, and www.youm7.com .

News Source	***Al-Ghad***	***Al-Riyadh***	***Al-Youm Al-Sabe'***	**ALL**
Country of publication	Jordan	Saudi Arabia	Egypt	
Primary dialect	Levantine	Gulf	Egyptian	
# articles	6.30k	34.2k	45.7k	86.1k
# comments	26.6k	805k	565k	1.4M
# sentences	63.3k	1,686k	1,384k	3.1M
# words	1.24M	18.8M	32.1M	52.1M
comments/article	4.23	23.56	12.37	16.21
sentences/comment	2.38	2.09	2.45	2.24
words/sentence	19.51	11.14	23.22	16.65

Table 3.2: A summary of the different components of the AOC dataset. Overall, 1.4M comments were harvested from 86.1k articles, corresponding to 52.1M words.

3.2 Arabic Dialect Identification

The discussion of the varieties of Arabic and the differences between them gives rise to the task of automatic *dialect identification* (DID). In its simplest form, the task is to build a learner that can, given an Arabic sentence S, determine whether or not S contains dialectal content. Another form of the task would be determine in *which* dialect S was written, which requires identification at a more fine-grained level.

In many ways, DID is equivalent to **language** identification, applied to a group of closely related languages that share a common character set. Given the parallels between DID and language identification, and the fact that the latter is a largely solved problem, is DID also easily solved with standard statistical methods? And what are some applications that would benefit from Arabic DID?

3.2.1 The Difficulty of Arabic DID

Despite the differences illustrated in the previous section, in which we justify treating the different dialects as separate languages, it is not a trivial matter to **automatically** distinguish and separate the dialects from each other. Since all Arabic varieties use the same character set, and furthermore much of the vocabulary is shared among different varieties, identifying dialect in a sentence is not simply a matter of, say, compiling a dialectal dictionary and detecting whether or not a given sentence contains dialectal words.

This word-level source ambiguity is caused by several factors:

- A dialectal sentence might consist entirely of words that are used across all Arabic varieties, including MSA. Each of the sentences in Figure 3.4 consists of words that are used both in MSA and dialectally, and an MSA-based dictionary would not (and should not) recognize those words as OOV. Nevertheless, the sentences are heavily dialectal.

- Some words are used across the varieties with *different* functions. For example, *Tyb* is used dialectally as an interjection, but is an adjective in MSA. (This is similar to the English usage of *okay*.)

- Primarily due to the omission of short vowels, a dialectal word might have the same spelling as an MSA word with an entirely different meaning, forming pairs of false friends. This includes strongly dialectal words such as *dwl* and *nby*: *dwl* is either Egyptian for *these* (pronounced *dowl*) or the MSA for *countries* (pronounced *duwal*); *nby* is either the Gulf for *we want* (pronounced *nibi*) or the MSA for *prophet* (pronounced *nabi*).

At this point, it might be somewhat mysterious to a non-Arabic speaker what, exactly, the difference is between MSA and local dialects. In particular, what makes certain sentences, such as those of Figure 3.4, dialectal, even when none of the individual words are? The answer lies in the **structure** of those sentences and the particular **word order** within them, rather than the individual words themselves.

AR (dialectal): معقول ينجح مهرجان الاردن السنة ؟

TL: *mEqwl ynjH mhrjAn AlArdn Alsnp ?*

Gloss: possible succeed festival Jordan the-year ?

EN: Is it possible that the Jordan Festival will succeed this year ?

AR (dialectal): يسلم قلمك يا استاذة حنان ، فرقعة اعلامية وخلاص

TL: *yslm qlmk yA AstA*p HnAn , frqEp AElAmyp wxlaS*

Gloss: be-safe pen-your oh teacher Hanan , explosion media and-done

EN: Bless your pen Mrs. Hanan , this is no more than media noise

AR (dialectal): الرجال افعال ، لو بكلام كان حكمت العالم بكلامي

TL: *AlrjAl AfEAl , lw bklAm kAn Hkmt AlEAlm bklAmy*

Gloss: the-men actions , if with-talk was ruled-I the-world with-talk-my

EN: Men are actions , if it were a matter of words I would have ruled the world with my words.

Figure 3.4: Three heavily dialectal sentences that do not contain individually dialectal words, taken from three different newspapers. A word-based OOV-detection approach would fail to classify these sentences as being dialectal, since all these words could appear in an MSA corpus.

Figure 3.4 shows MSA sentences that express the same meaning as the dialectal sentences from Figure 3.4. As one could see, the two versions of any given sentence could share much of the vocabulary, but in ways that are noticeably different to an Arabic speaker. Furthermore, the differences would be starker still if the MSA sentences were composed from scratch, rather than by modifying the dialectal sentences, since the tone might differ substantially when composing sentences in MSA.

AR (dialectal): معقول ينجح مهرجان الاردن السنة ؟

TL: *mEqwl ynjH mhrjAn AlArdn Alsnp ?*

Gloss: possible succeed festival Jordan the-year ?

AR (MSA): هل من الممكن أن ينجح مهرجان الأردن هذه السنة ؟

TL: *hl mn Almmkn >n ynjH mhrjAn Al>rdn h*h Alsnp ?*

Gloss: is? of the-possible that succeed festival Jordan this the-year ?

EN: Is it possible that the Jordan Festival will succeed this year ?

AR (dialectal): يسلم قلمك يا استاذة حنان ، فرقعة اعلامية وخلاص

TL: *yslm qlmk yA AstA*p HnAn , frqEp AElAmyp wxlaS*

Gloss: be-safe pen-your oh teacher Hanan , explosion media and-done

AR (MSA): سلم قلمك يا أستاذة حنان ، هذه مجرد ضجة إعلامية

TL: *slm qlmk yA >stA*p HnAn , h*h mjrd Djp <ElAmyp*

Gloss: was-safe pen-your oh teacher Hanan , this only noise media

EN: Bless your pen Mrs. Hanan , this is no more than media noise

AR (dialectal): الرجال افعال ، لو بكلام كان حكمت العالم بكلامي

TL: *AlrjAl AfEAl , lw bklAm kAn Hkmt AlEAlm bklAmy*

Gloss: the-men actions , if with-talk was ruled-I the-world with-talk-my

AR (MSA): الرجال بالأفعال ، لو بالكلام لحكمت العالم بكلامي

TL: *AlrjAl bAl>fEal , lw bAlklAm lHkmt AlEAlm bklAmy*

Gloss: the-men with-the-actions , if with-the-talk would-ruled-I the-world with-talk-my

EN: Men are actions , if it were a matter of words I would have ruled the world with my words.

Figure 3.5: The dialectal sentences of Figure 3.4, with MSA equivalents.

3.2.2 Applications of Dialect Identification

Being able to perform automatic DID is interesting from a purely linguistic and experimental point of view. In addition, automatic DID has several useful applications:

- Distinguishing dialectal data from non-dialectal data would aid in creating a large monolingual dialectal dataset, exactly as we would hope to do with the AOC dataset. Such a dataset would aid many NLP systems that deal with dialectal content, for instance to train a language model for an Arabic dialect speech recognition system (Novotney et al., 2011). Identifying dialectal content can also aid in creating parallel datasets for machine translation, with a dialectal source side.

- A user might be interested in content of a specific dialect, or, conversely, in strictly non-dialectal content. This would be particularly relevant in fine-tuning and personalizing search engine results, and could allow for better user-targeted advertising. In the same vein, being able to recognize dialectal content in user-generated text could aid in characterizing communicants and their biographic attributes (Garera and Yarowsky, 2009).

- In the context of an application such as machine translation, identifying dialectal content could be quite helpful. Most MT systems, when faced with OOV words, either discard the words or make an effort to transliterate them. If a segment is identified as being dialectal first, the MT system might instead attempt to find equivalent MSA words, which are presumably easier to process correctly (e.g. as in Salloum and Habash (2011) and, to some degree, Habash (2008)). Even for non-OOV words, identifying dialectal content before translating could be critical, to resolve the false-friends ambiguity of the kind mentioned in 3.2.1.

3.3 Crowdsourcing Arabic Dialect Annotation

In this section, we discuss crowdsourcing Arabic dialect annotation. We discuss how we built a dataset of Arabic sentences, each of which is labeled with whether or not it contains dialectal content. The labels include additional details about the **level** of dialectal content, if it exists, and of which **type** of dialect it is. The sentences themselves are sampled from the AOC Dataset, and we observe that about 40% of sentences contain dialectal content, with that percentage varying between 37% and 48%, depending on the news source.

We first present the annotation interface and discuss an effective way for quality control that can detect spamming behavior. We then examine the collected data itself, analyzing annotator behavior, measuring agreement among annotators, and identifying interesting biases exhibited by the annotators. In Section 3.4, we use the collected data to train and evaluate statistical models for several dialect identification tasks.

3.3.1 Annotation Interface

The annotation interface displayed a group of Arabic sentences, randomly selected from the AOC. For each sentence, the annotator was instructed to examine the sentence and make two judgments about its dialectal content: the **level** of dialectal content, and its **type**, if any. The instructions were kept short and simple:

> This task is for Arabic speakers who understand the different local Arabic dialects, and can distinguish them from *Fusha*[9] Arabic.
>
> Below, you will see several Arabic sentences. For each sentence:
>
> 1. Tell us how much dialect is in the sentence, and then
> 2. Tell us which Arabic dialect the writer intends.

[9] *Fusha* is the Arabic word for MSA, pronounced *foss-ha*.

The instructions were accompanied by the map of Figure 3.1, to visually illustrate the dialect breakdown. Figure 3.6 shows the annotator interface populated with some actual examples, with labeling in progress. We also collected self-reported information such as native Arabic dialect and number of years speaking Arabic, and the interface had built-in functionality to detect each annotator's geographic location based on their IP address.

Of the 3.1M sentences in the AOC, we selected a 'small' subset of about 110,000 sentences to be annotated for dialect.[10] The sentences were randomly grouped into sets of 10 sentences each, and when Workers performed our task, they were shown the 10 sentences of a randomly selected set, on a single HTML page. As a result, each screen contained a mix of sentences across the three newspapers presented in random order. As control items, each screen had two additional sentences that were taken from the *article bodies*. Such sentences are almost always in MSA Arabic, and so their expected label is MSA. Any worker who frequently mislabeled the control sentences with a non-MSA label was considered a spammer, and their work was rejected.

Hence, each screen had twelve sentences in total. We offered a reward of $0.05 per screen (later raised to $0.10), and had each set redundantly completed by three distinct Workers. The data collection lasted about 4.5 months, during which 33,093 HIT Assignments were completed, corresponding to 330,930 collected labels (excluding control items). The total cost on annotation was $3,050.52 ($2,773.20 for rewards, and $277.32 for Amazon's commission).

3.3.2 Annotator Behavior

With the aid of the embedded control segments (taken from article bodies) and expected dialect label distribution, it was possible to spot spamming behavior and reject it. Table 3.3 shows three examples of workers whose work was rejected on this basis, having clearly demonstrated they are unable or unwilling to perform the task

[10] There are far fewer sentences available from *Al-Ghad* than the other two sources (fourth line of Table 3.2). We have taken this imbalance into account and heavily oversampled *Al-Ghad* sentences when choosing sentences to be labeled.

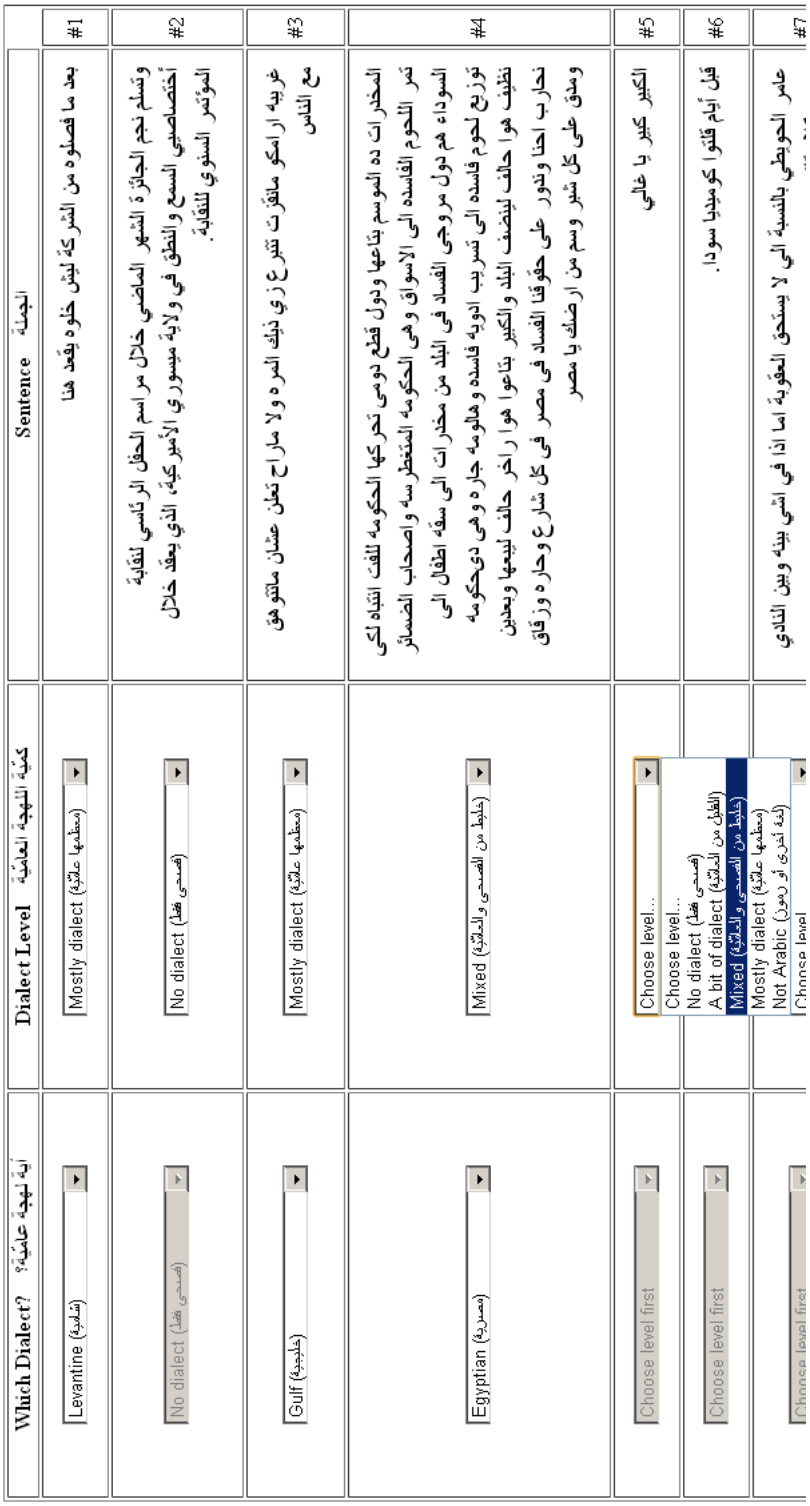


Figure 3.6: The interface for the dialect identification task. This example, and the full interface, can be viewed at the URL http://bit.ly/eUtiO3 .

	A29V7OGM2C6205	A3SZLM2NK8NUOG	A8EF1I6CO7TCU	Typical
`MSA` in control items	0%	14%	33%	>90%
`LEV` in *Al-Ghad*	0%	0%	15%	25%
`GLF` in *Al-Riyadh*	8%	0%	14%	20%
`EGY` in *Al-Youm Al-Sabe'*	5%	0%	27%	33%
Other dialects	56%	0%	28%	<1%
Incomplete answers	13%	6%	1%	<2%
Worker location	Romania	Philippines	Jordan	Middle East
Claimed native dialect	Gulf	"Other"	Unanswered	(Various)

Table 3.3: Some statistics over the labels provided by three spammers. Compared to the typical worker (right-most column), all workers perform terribly on the MSA control items, and also usually fail to recognize dialectal content in commentary sentences. Other red flags, such as geographic location and 'identifying' unrepresented dialects, are further proof of the spammy behavior.

faithfully. 11.4% of the assignments were rejected on this basis. In the approved assignments, the embedded MSA control sentence was annotated with the MSA label 94.4% of the time. In the remainder of this Chapter, we analyze only data from the approved assignments.

We note here that we only rejected assignments where the annotator's behavior was **clearly** problematic, opting to *approve* assignments from workers mentioned later in 3.3.2.3, who exhibit systematic biases in their labels. While these annotators' behavior is non-ideal, we cannot assume that they are not working faithfully, and therefore rejecting their work might not be fully justified. Furthermore, such behavior might be quite common, and it is worth investigating these biases to benefit future research.

3.3.2.1 Label Distribution

Overall, 454 annotators participated in the task, 138 of whom completed at least 10 HITs. Upon examination of the provided labels for the commentary sentences, 40.7% of them indicate some level of dialect, while 57.1% indicate no dialectal content (Figure 3.7(a)). Note that 2.14% of the labels identify a sentence as being non-Arabic, non-textual, or were left unanswered. The label breakdown is a strong confirmation of our initial motivation, which is that a large portion of reader commentary contains dialectal content.[11]

Figure 3.7 also illustrates the following:

- The most common dialectal label within a given news source matches the dialect of the country of publication. This is not surprising, since the readership for any newspaper is likely to mostly consist of the local population of that country.
- The three news sources vary in the prevalence of dialectal content. The Egyptian newspaper has a markedly larger percentage of dialectal content (46.6% of labels) compared to the Saudi newspaper (40.1%) and the Jordanian newspaper (36.8%).
- A nontrivial amount of labels (5-8%) indicate `General` dialectal content. The `General` label was meant to indicate a sentence that is dialectal but lacks a strong indication of a *particular* dialect. While many of the provided `General` labels presumably reflect the annotator's intent to express this fact, there is evidence that some annotators used this category in cases where `Not sure` might have been more appropriate (see 3.3.2.3).
- Non-Arabic content, while infrequent, is not a rare occurrence in the Jordanian and Egyptian newspapers, at around 3%. The percentage is much lower in the Saudi newspaper, at 0.8%. This might reflect the deeper penetration of the

[11] Later analysis in 3.3.2.3 shows that a non-trivial portion of the labels were provided by MSA-biased annotators, indicating that dialectal content could be even more prevalent than what is initially suggested by the MSA/dialect label breakdown.

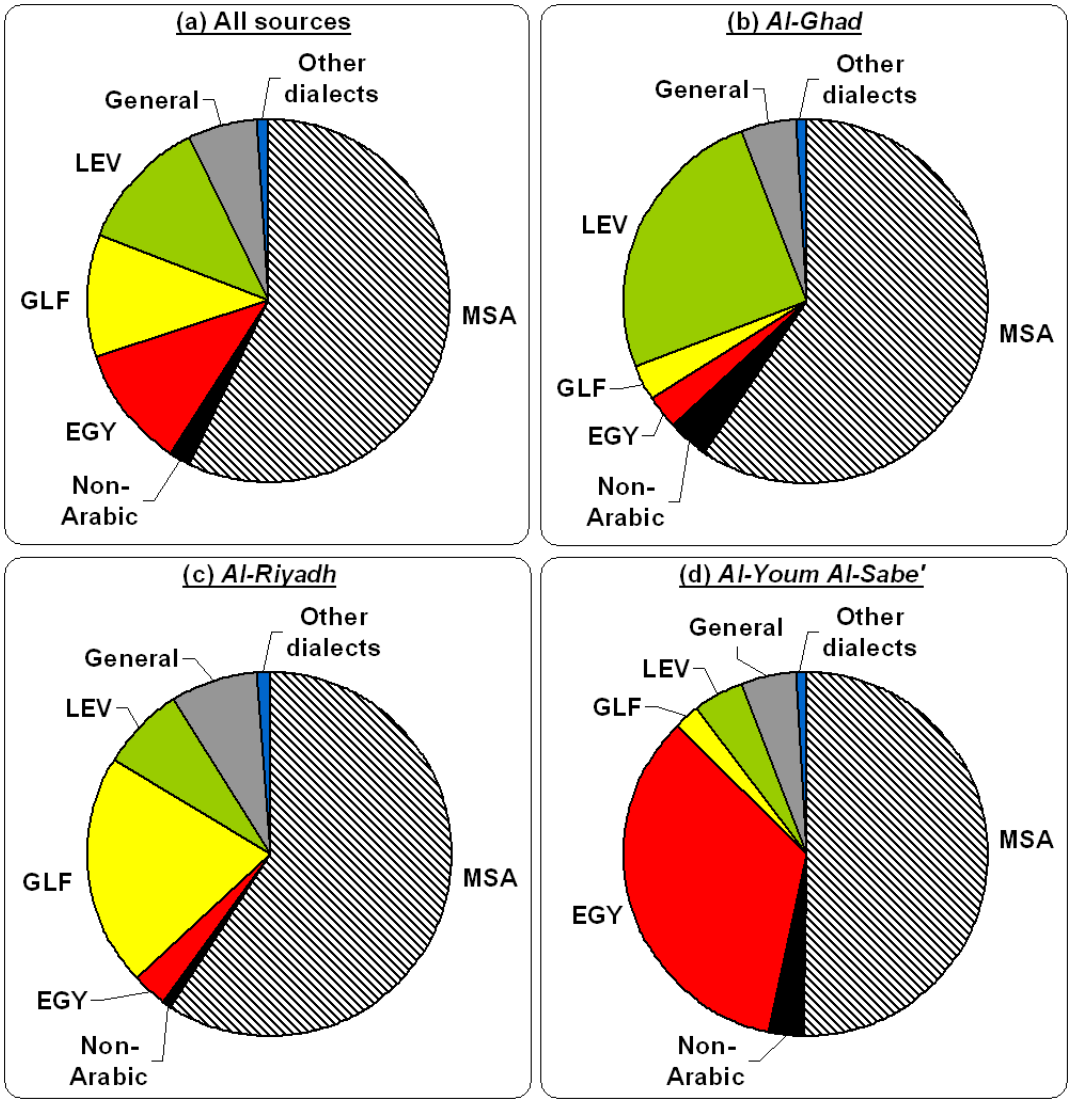


Figure 3.7: The distribution of labels provided by the workers for the dialect identification task, over all three news sources (a) and over each individual news source (b–d). *Al-Ghad* is published in Jordan, *Al-Riyadh* in Saudi Arabia, and *Al-Youm Al-Sabe'* in Egypt. Their local readerships are reflected in the higher proportion of corresponding dialects.

English language (and English-only keyboards) in Jordan and Egypt compared to Saudi Arabia.

We can associate a label with each segment based on the majority vote over the three provided labels for that segment. If a sentence has at least two annotators choosing a dialectal label, we label it as `Dialect`. If it has at least two annotators choosing the MSA label, we label it as `MSA`.[12] In the remainder of the Chapter, we will report classification accuracy rates that assume the presence of gold-standard class labels. Unless otherwise noted, this majority-vote label set is used as the gold-standard in such experiments.

3.3.2.2 Annotator Agreement and Performance

The annotators exhibit a decent level of agreement with regard to whether a segment is dialectal or not, with full agreement (i.e. across all three annotators) on 72.2% of the segments regarding this binary `Dialect`/`MSA` decision. (The probability of three annotators agreeing randomly on a binary decision is 25%.) The full-agreement percentage decreases to 56.2% when expanding the classification from a binary decision to a fine-grained scale that includes individual dialect labels as well. This is still quite a reasonable result, since the criterion is somewhat strict: it does not include a segment labeled, say, {`Levantine`, `Levantine`, `General`}, though there is good reason to consider that annotators are in 'agreement' in such a case.

So how good are humans at the classification task? We examine their classification accuracy, dialect recall, and MSA recall. We define dialect (MSA) recall to be the number of sentences labeled as being dialectal (MSA), over the total number of sentences that have dialectal (MSA) labels based on the majority vote. Overall, human annotators have a classification accuracy of 90.3%, with dialect recall at 89.0%, and MSA recall at 91.5%. Those recall rates do vary across annotators, as shown in Figure 3.8, causing some accuracy rates to drop as low as 80% or 75%. Of the annotators performing at least 5 HITs, 89.4% have accuracy rates >= 80%.

[12] A very small percentage of sentences (2%) do not have either kind of agreement. Upon inspection, these were typically found to be sentences that are in English, e-mail addresses, romanized Arabic, or simply random symbols.

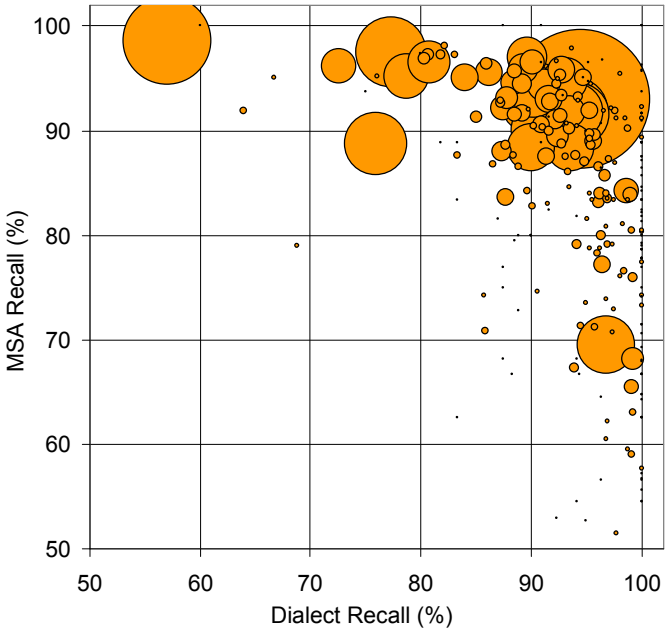


Figure 3.8: A bubble chart showing workers' MSA and dialect recall. Each data point (or 'bubble') in the graph represents one annotator, with the bubble size corresponding to the number of Assignments completed by that annotator.

Most annotators have both high MSA recall and high dialect recall, with about 70% of them achieving at least 80% in both MSA and dialect recall. Combined with the general agreement rate measure, this is indicative that the task is well-defined – it is unlikely that many people would agree on something that is incorrect.

3.3.2.3 Annotator Bias Types

Examining the submitted labels of individual workers reveals interesting annotation patterns, and indicates that annotators are quite diverse in their behavior. An annotator can be observed to have one or more of the following bias types:[13]

- **MSA bias/dialect bias**: Figure 3.8 shows that annotators vary in how willing they are to label a sentence as being dialectal. While most workers (top right)

[13]These biases should be differentiated from *spammy* behavior, which we already can deal with quite effectively, as explained in 3.3.2.

exhibit both high MSA and high dialect recall, other annotators have either a MSA bias (top left) or a dialect bias (bottom right).

- **Dialect-specific bias**: Many annotators over-identify a particular dialect, usually their native one. If we group the annotators by their native dialect and examine their label breakdown (Table 3.4), we find that Egyptian speakers over-identify segments as being Egyptian, Gulf speakers over-identify Gulf, and Levantine speakers over-identify Levantine. This holds for speakers of other dialects as well, as they over-identify other dialects more often than most speakers. Another telling observation is that Iraqi speakers have a bias for the Gulf dialect, which is quite similar to Iraqi, and Maghrebi speakers have a bias for Egyptian, reflecting geographic distribution of the dialects.

- **The `General` bias**: The `General` label is meant to signify sentences that cannot be definitively classified as one dialect over another. This is the case when enough evidence exists that the sentence is not in MSA, but contains no evidence for a specific dialect. In practice, some annotators make very little use of this label, even though many sentences warrant its use, while other annotators make extensive use of this label (see for example Table 3.5). One interesting case is that of annotators whose `General` label seem to mean they are unable to identify the dialect, and a label like `Not sure` might have been more appropriate. Take the case of the Maghrebi worker in Table 3.5, whose `General` bias is much more pronounced in the Jordanian and Saudi newspapers. This is an indication they might have been having difficulty distinguishing Levantine and Gulf from each other, but are familiar with the Egyptian dialect.

	Group size	% EGY	% GLF	% LEV	% GNRL	% Other dialects
All speakers	454	26.1	27.1	28.8	15.4	2.6
Egyptian speakers	121	**38.0**	19.1	25.9	10.9	6.1
Gulf speakers	32	25.6	**29.4**	21.7	21.8	1.4
Levantine speakers	181	21.2	28.4	**35.9**	12.9	1.6
Iraqi speakers	16	23.9	**29.0**	18.9	18.2	10.1
Maghrebi speakers	67	**34.5**	28.0	20.5	12.7	4.3
Other/Unknown	37	27.8	18.8	17.9	**31.4**	4.1

Table 3.4: The specific-dialect label distribution (given that a dialect label was provided), shown for each speaker group.

	All workers	A1M50UV37AMBZ3	A2ZNK1PZOVIECD
% General	6.3	12.0	2.3
% General in *Al-Ghad*	5.2	14.2	3.1
% General in *Al-Riyadh*	7.7	13.1	2.6
% General in *Al-Youm Al-Sabe'*	4.9	7.6	1.0
Native dialect	(Various)	Maghrebi	Egyptian

Table 3.5: Two annotators with a General label bias, one who uses the label liberally, and one who is more conservative. Note that in both cases, there is a noticeably smaller percentage of General labels in the Egyptian newspaper than in the Jordanian and Saudi newspapers.

3.4 Models for Automatic Dialect Identification

From a computational point of view, we can think of dialect identification as language identification, though with finer-grained distinctions that make it more difficult than typical language ID. Even languages that share a common character set can be be distinguished from each other at high accuracy rates using methods as simple as examining character histograms (Cavnar and Trenkle, 1994; Dunning, 1994; Souter et al., 1994), and, as a largely-solved problem, the one challenge becomes whether the language can be identified for very short segments.

Due to the nature and characteristics and high overlap across Arabic dialects, relying on character histograms alone is ineffective (see 3.4.3.1), and more context is needed. We will explore higher-order letter models as well as word models, and determine what factors determine which model is best.

3.4.1 Smoothed N-Gram Models

Given a sentence S to classify into one of k classes $C_1, C_2, ..., C_k$, we will choose the class with the maximum conditional probability:

$$C^* = \underset{C_i}{\operatorname{argmax}}\, P(C_i|S) = \underset{C_i}{\operatorname{argmax}}\, P(S|C_i) \cdot P(C_i) \tag{3.1}$$

Note that the decision process takes into account the prior distribution of the classes, which is estimated from the training set. The training set is also used to train probabilistic models to estimate the probability of S given a particular class. We rely on training n-gram language models to compute such probabilities. In language model scoring, a sentence is typically split into words. We will also consider *letter*-based models, where the sentence is split into sequences of characters. Note that letter-based models would be able to take advantage of clues in the sentence that are not complete words, such as prefixes or suffixes. This would be useful if the amount of training data is very small, or if we expect a large domain shift between training

and testing, in which case content words indicative of MSA or dialect might not still be valuable in the new domain.

3.4.2 Baselines

To properly evaluate classification performance trained on dialectal data, we compare the language-model classifiers to two baselines that do not use the newly collected data. Rather, they use available MSA-only data and attempt to determine how MSA-like a sentence is.

The first baseline is based on the assumption that a dialectal sentence would contain a higher percentage of 'non-MSA' words that cannot be found in a large MSA corpus. To this end, we extracted a vocabulary list from the Arabic Gigaword Corpus, producing a list of 2.9M word types. Each sentence is given a score that equals the OOV percentage, and if this percentage exceeds a certain threshold, the sentence is classified as being dialectal. Another variant of this baselines excludes singletons from the vocabulary list.

The second approach is more fine-grained. We train a language model using MSA-only data, and use it to score a test sentence. Again, if the perplexity exceeds a certain threshold, the sentence is classified as being dialectal. To take advantage of domain knowledge, we train this MSA model on the sentences extracted from the article bodies of the AOC, which corresponds to 43M words of highly-relevant content.

3.4.3 Experimental Results

In this section, we explore using the collected labels to train word- and letter-based DID systems, and show that they outperform other baselines that do not use the annotated data.

3.4.3.1 MSA vs. Dialect Classification

We measure classification accuracy at various training set sizes, using 10-fold cross validation, for several classification tasks. We examine the task both as a general MSA

Model	MSA vs. dialect	*Al-Ghad* MSA vs. dialect	*Al-Riyadh* MSA vs. dialect	*Al-Youm Al-Sabe'* MSA vs. dialect	*Al-Ghad* dialect vs. *Al-Riyadh* dialect vs. *Al-Youm Al-Sabe'* dialect
Majority Class	58.75	62.54	59.99	51.89	46.49
OOV % vs. Gigaword	65.46	65.11	65.30	66.72	N/A
(no singletons)	65.52	65.60	65.13	66.74	N/A
MSA LM-scoring	66.56	67.80	66.78	65.22	N/A
Letter-based, 1-graph	68.10	69.92	68.02	70.35	56.30
Letter-based, 3-graph	83.49	85.11	81.85	85.96	84.86
Letter-based, 5-graph	85.01	85.68	81.41	87.04	**88.70**
Word-based, 1-gram	**85.65**	**87.18**	**83.26**	**87.90**	88.44
Word-based, 2-gram	82.77	84.09	80.63	85.85	88.38
Word-based, 3-gram	82.52	83.69	80.38	85.62	88.37

Table 3.6: Accuracy rates (%) on several classification tasks for various models. Models in the top part of the table do not use the dialect-annotated data, while models in the bottom part do. (For the latter kind of models, the accuracy rates reported are based on a training set size of 90% of the available data.)

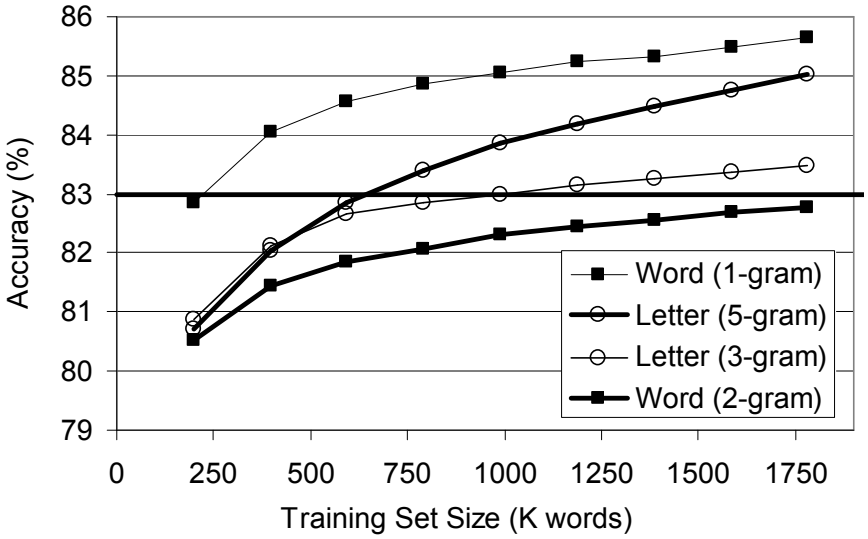


Figure 3.9: Learning curves for the general MSA vs. dialect task, with all three news sources pooled together. Learning curves for the individual news sources can be found in Figure 3.10. The 83% line has no significance, and is provided to ease comparison with Figure 3.10.

vs. dialect task, as well as when restricted within a particular news source. We train unigram, bigram, and trigram (word-based) models, as well as unigraph, trigraph, and 5-graph (letter-based) models. Table 3.6 summarizes the accuracy rates for these models, and compares them to accuracy rates for the baselines that do not use the dialect-annotated data.

Generally, we find that a unigram word model performs best, with a 5-graph model slightly behind. Bigram and trigram word models suffer from the sparseness of the data and lag slightly behind, given the large number of parameters they would need to estimate (and instead resort to smoothing heavily). The 3- and 5-graph letter-based models, with a significantly smaller vocabulary size, do not suffer from this problem, and perform better. This could be a double-edged sword though, especially for the trigraph model, as it means the model is less expressive and converges faster.

Overall though, the experiments show a clear superiority of a supervised method, be it word- or letter-based, over baselines that use existing MSA-only data. Whichever model we choose (with the exception of the unigraph model), the obtained accuracy

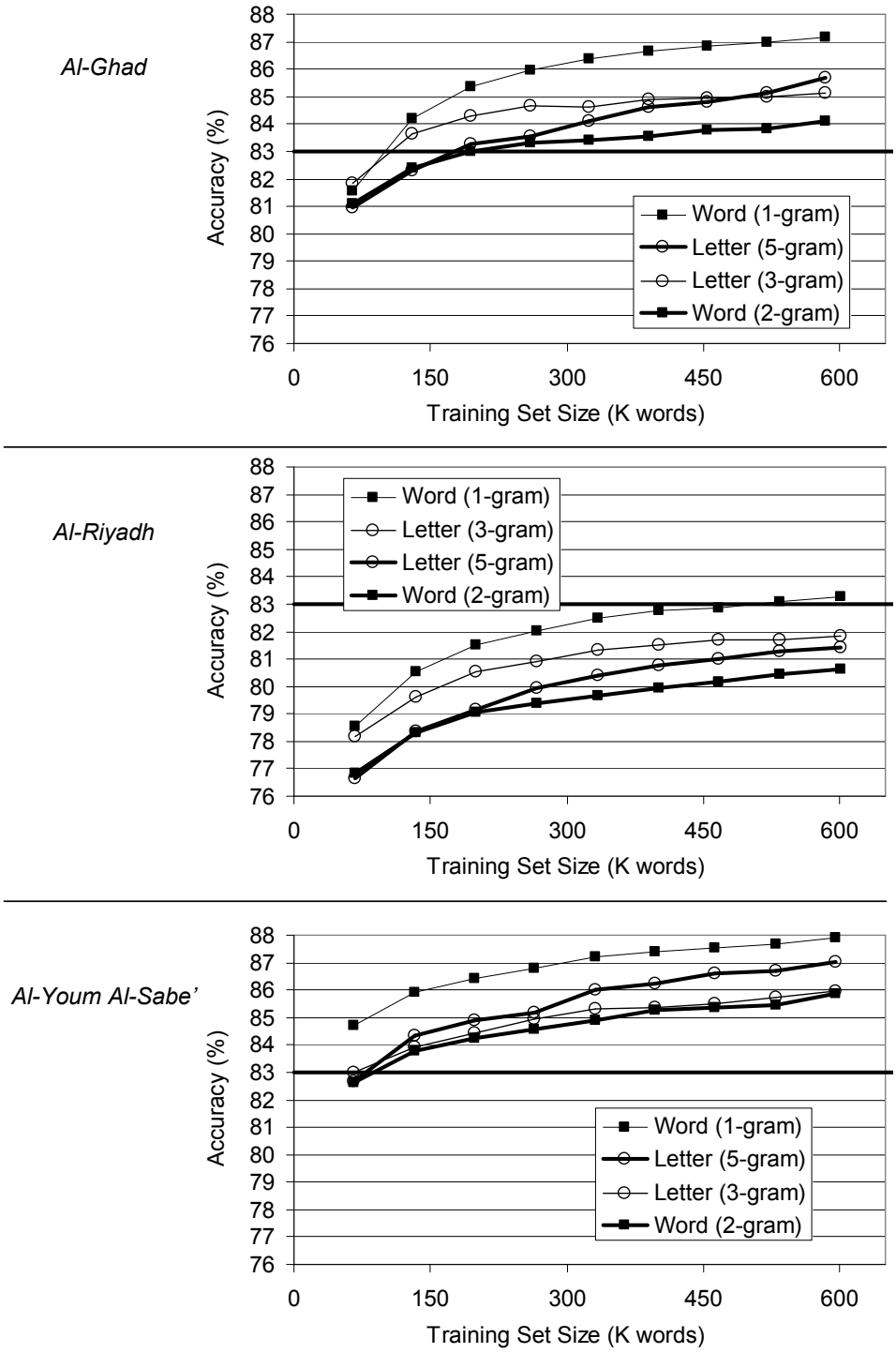


Figure 3.10: Learning curves for the MSA vs. dialect task, for each of the three news sources. The 83% line has no significance, and is provided to ease comparison across the three components, and with Figure 3.9.

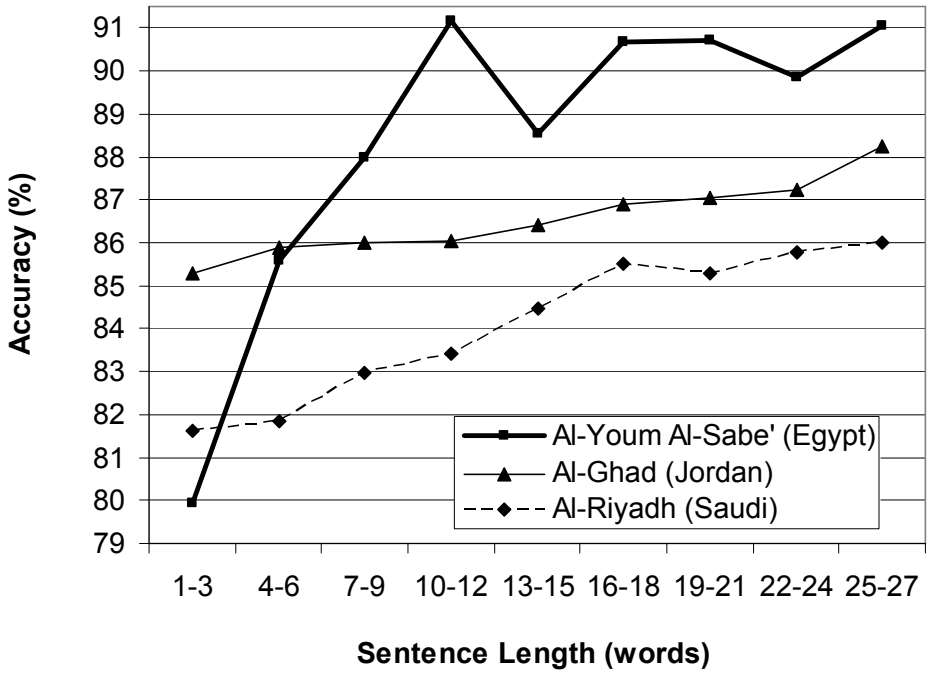


Figure 3.11: Accuracy rate vs. sentence length (for the unigram word model trained on 90% of the data).

rates show a significant dominance over the baselines.

It is worth noting that a classification error becomes less likely to occur as the length of the sentence increases (Figure 3.11). This is not surprising given prior work on the language identification problem (Řehůřek and Kolkus, 2009; Verma et al., 2009), which points out that the only 'interesting' aspect of the problem is performance on short segments. The same is true in the case of dialect identification: a short sentence that contains even a single misleading feature is prone to misclassification, whereas a long sentence is likely to have other features that help identify the correct class label.

One could also observe that distinguishing dialect is a more difficult task in the Saudi newspaper than in the Jordanian, which in turn is harder than in the Egyptian newspaper. This could be evidence that the Gulf dialect is the closest of the dialects to MSA, and Egyptian is the farthest.[14] Note also that this is not due to the fact

[14]This is indeed in agreement with the history of MSA, which descends from an Arabic variety

that the Saudi sentences tend to be significantly shorter – the ease of distinguishing Egyptian holds even at higher sentence lengths, as shown by Figure 3.11.

Examining the letter and word distribution in the corpus provides valuable insight into what features of a sentence are most dialectal. Let $DF(w)$ denote the *dialectness factor* of a word w, defined as:

$$DF(w) \stackrel{\text{def}}{=} \frac{f(w|D)}{f(w|MSA)} = \frac{count_D(w)/count_D(.)}{count_{MSA}(w)/count_{MSA}(.)} \tag{3.2}$$

where $count_D(w)$ (resp. $count_{MSA}(w)$) is the number of times w appeared in the dialectal (resp. MSA) sentences, and $count_D(.)$ is the total number of words in those sentences. Hence, $DF(w)$ is simply a ratio measuring how much more likely w is to appear in a dialectal sentence, than in an MSA sentence. Note that the dialectness factor can be easily computed for letters as well, and can be computed for bigrams/bigraphs, trigrams/trigraphs, etc.

Figure 3.12 lists, for each news source, the word types with the highest and lowest dialectness factor. The most dialectal words tend to be function words, and they also tend to be **strong** indicators of dialect, judging by their very high DF. On the other hand, the MSA word group contains several content words, relating mainly to politics and religion.

One must also take into account the actual frequency of a word, as DF only captures relative frequencies of dialect/MSA, but does not capture how often the word occurs in the first place. Figure 3.13 plots both measures for the words of *Al-Ghad* newspaper. The plot illustrates which words are most important to the classifier: the words that are farthest away from the point of origin, along both dimensions.

As for letter-based features, many of the longer ones (e.g. 5-graph features) are essentially the same words important to the unigram word model. The letter-based models are however able to capture some linguistic phenomena that the word model is unable to: the suffixes $+\$$ (*not* in Levantine) and $+wn$ (plural conjugation in Gulf), and the prefixes $H+$ (*will* in Egyptian), $bt+$ (present tense conjugation in Levantine and Egyptian), and $y+$ (present tense conjugation in Gulf).

that was spoken in the Gulf region.

Al-Ghad				*Al-Riyadh*				*Al-Youm Al-Sabe'*			
w		Gloss	*DF*(*w*)	*w*		Gloss	*DF*(*w*)	*w*		Gloss	*DF*(*w*)
$w	شو	what	139.0	*Ay$*	ايش	what	247.7	*AwY*	اوى	very	116.6
xly	خلي	let	132.8	*w$*	وش	what	105.9	*dy*	دي	this (f.)	108.8
xlS	خلص	enough	117.5	*lyn*	لين	until	92.9	*glT*	غلط	wrong	106.6
AlHky	الحكي	the-talk	115.8	*xl*	خل	let	83.8	*dhwqtY*	دلوقتى	now	75.7
Endw	عندو	he has	95.3	*ly$*	ليش	why	82.4	*E$An*	عشان	so that	70.2
bdy	بدي	I will/want	93.6	*jAy*	جاي	coming	72.2	*mAfy$*	مافيش	none	65.7
AzA	ازا	if	93.6	*ybwn*	بيون	they will/want	69.7	*dY*	دى	this (f.)	64.5
mnyH	منيح	good	93.6	*blA$*	بلاش	lest	65.8	*tAny*	تاني	another/again	62.3
$wy	شوي	little	92.8	*El$An*	علشان	so that	64.5	*dh*	ده	this (m.)	61.4
<nw	انو	that	90.2	*lyh*	ليه	why	48.6	*Antw*	انتو	you (pl.)	59.8
hAy	هاي	this (f.)	80.0	*wbs*	وبس	that's all	46.0	*AntwA*	انتوا	you (pl.)	59.3
bEdyn	بعدين	then	70.2	*yby*	بيي	he will/wants	44.4	*AntA*	انتا	you (s.)	58.8
mw	مو	not	65.5	*$wy*	شوي	little	43.9	*jAY*	جاى	coming	58.8
Ay$	ايش	what	63.8	*mArAH*	ماراح	will not	39.6	*EAwz*	عاوز	I want	57.8
bdw	بدو	he will/wants	60.3	*wyn*	وين	where	38.8	*El$An*	علشان	so that	57.5
		.	.			.	.			.	
		.	.			.	.			.	
		.	.			.	.			.	
Ebr	عبر	through	0.146	*<lyh*	إليه	to him	0.154	*<lY*	إلى	to	0.133
w>n	وأن	and-that	0.145	*w>n*	وأن	and-that	0.149	*AlmsyH*	المسيح	Christ	0.125
Al<slAm	الإسلام	Islam	0.138	*rswl*	رسول	messenger	0.131	*<dArp*	إدارة	management	0.125
tEAlY	تعالى	almighty	0.138	*ns>l*	نسأل	we ask	0.130	*flmA*A*	فلماذا	so-why	0.113
SlY	صلى	blessed	0.127	*fymA*	فيما	whilst	0.127	*$yjA*	شيئا	(any)thing	0.111
AldymqrATyp	الديمقراطية	democratic	0.108	*y>ty*	يأتي	comes	0.127	*AlfADl*	الفاضل	esteemed	0.092
Alljnp	اللجنة	the-committee	0.095	*tEAlY*	تعالى	almighty	0.122	*ljmAl*	لجمال	to-Jamal	0.090
f<n	فإن	(declarative)	0.062	*fmn*	فمن	who	0.117	*Al>stA**	الأستاذ	mister	0.078
AlmfAwDAt	المفاوضات	the-negotiations	0.038	*tlk*	تلك	that (f.)	0.105	*<lyh*	إليه	to-him	0.055
AlmbA$rp	المباشرة	the-direct	0.029	*lqd*	لقد	(declarative)	0.090	*lldktwr*	للدكتور	to-the-doctor	0.051

Figure 3.12: Words with the highest and lowest dialectness factor values.

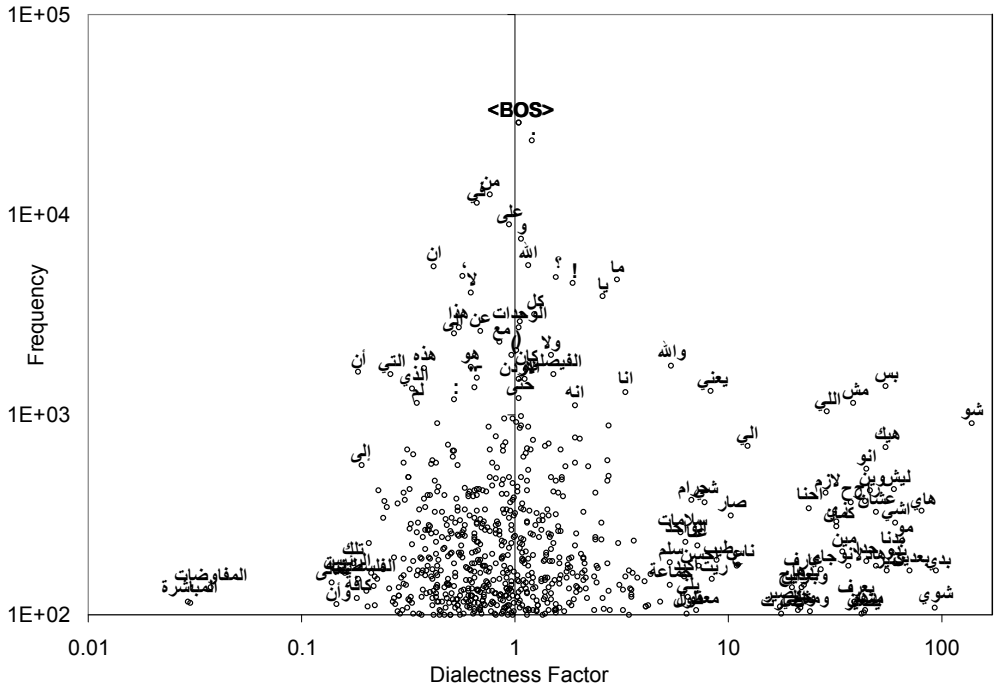


Figure 3.13: A plot of the most common words in the *Al-Ghad* sentences, showing each word's DF and corpus frequency. The right- and left-most words here also appear in Figure 3.12. Not every word from that list appears here though, since some words have counts below 100. For clarity, not all points display the word they represent.

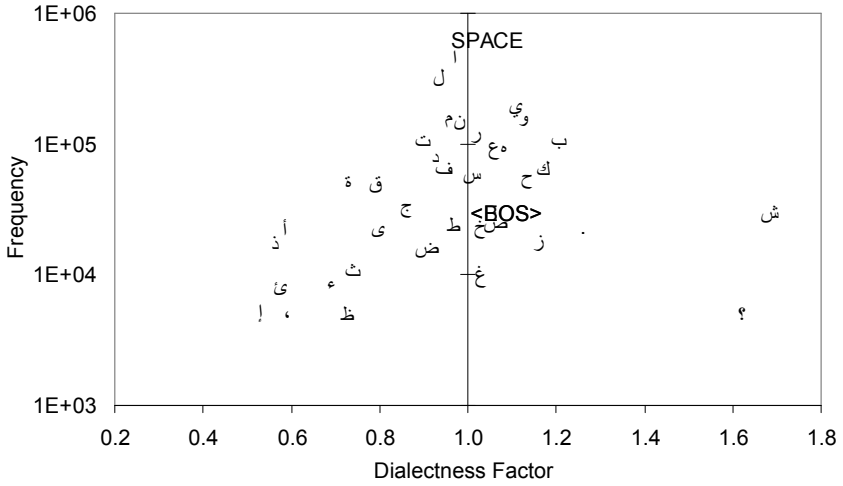


Figure 3.14: A plot of the most common letters in the *Al-Ghad* sentences, showing each letter's DF and corpus frequency.

Figure 3.14 sheds some light on why even the unigraph model outperforms the baselines. It picks up on subtle properties of the MSA writing style that are lacking when using dialect. Namely, there is closer attention to following *hamza* rules (distinguishing A, $<$, and $>$ from each other, rather than mapping them all to A), and better adherence to (properly) using $+p$ instead of $+h$ at the end of many words. There is also a higher tendency to use words containing the letters that are most susceptible to being corrupted when pronounced dialectally: $*$ (usually corrupted as z), Z (corrupted as D), and v (corrupted as t).

3.5 Related Work

The Cross Lingual Arabic Blog Alerts (COLABA) project (Diab et al., 2010) is another large effort to create dialectal Arabic resources (and tools). They too focus on online sources such as blogs and forums, and use information retrieval tasks to measure their ability to properly process dialectal Arabic content. The COLABA project demonstrates the importance of using dialectal content when training and designing tools that deal with dialectal Arabic, and deal quite extensively with resource creation and data harvesting for dialectal Arabic.

Chiang et al. (2006) investigate building a parser for Levantine Arabic, *without* using any significant amount of dialectal data. They utilize an available Levantine-MSA lexicon, but no parses of Levantine sentences. Their work illustrates the difficulty of adapting MSA resources for use in a dialectal domain.

As far as we can tell, no prior dialect identification work exists that is applied to Arabic text. However, Lei and Hansen (2011) and Biadsy et al. (2009) investigate Arabic dialect identification in the *speech* domain. Lei and Hansen build Gaussian mixture models to identify the same three dialects we consider, and are able to achieve an accuracy rate of 71.7% using about 10 hours of speech data for training. Biadsy et al. use a much larger dataset (170 hours of speech data) and take a phone recognition and language modeling approach (of Zissman (1996)). In a four-way classification task (with Iraqi as a fourth dialect), they achieve a 78.5% accuracy rate. It must be

noted that both works use *speech* data, and that dialect identification is done on the *speaker* level, not the sentence level as done here.

3.6 Conclusion

Online reader commentary is a rich source of dialectal Arabic that has not been harvested before. We relied on this resource to create a large dataset of informal Arabic, rich in dialectal content, called the Arabic Online Commentary Dataset. We then crowdsourced a labeling task to identify sentences that have dialectal content, creating a novel dataset where each sentence is annotated with the level and type of dialect in it. We used the collected labels to train and evaluate automatic classifiers for dialect identification, a task that had previously been impossible to tackle with statistical methods due to lack of training data. Our experiments uncovered interesting linguistic aspects about the task and annotators' behavior. Most importantly, our classifiers significantly outperformed baselines that use MSA-only data.

Given the recent political unrest in the Middle East (2011), other rich sources of dialectal Arabic are Twitter posts (e.g. with the `#Egypt` tag) and discussions on various political Facebook groups. Like the reader commentary we harvested, such posts are very likely to contain a high degree of dialectal data, given the topic at hand and the individualistic nature of the posts. Once harvested, dialectal content could be very useful in a number of ways. For example, we revisit the dialect identification task in Chapter 6, as we present a pipeline for creating a dialectal Arabic-English parallel dataset. The dialect identification task is the first phase of the pipeline, and we manage to create a parallel dataset 1.5M words in size by crowdsourcing the translation task. This new parallel corpus is used to train MT systems that dramatically outperform MSA-trained systems when translating dialectal Arabic content.

This Chapter demonstrated that crowdsourcing makes it much easier for researchers to create their own annotated datasets, even for rarer languages. Rather than having to rely on data creation efforts by the LDC, for instance, researchers are able to collect annotations with relatively little overhead or delay – and can control

the style and form of the annotations as well. In addition to allowing us to collect data for new tasks, crowdsourcing also allows us to collect new *types* of data. In the next Chapter, we present *annotator rationales*, a new type of annotation that allows us to train more sophisticated models. In the context of the dialect identification task, we crowdsourced a task where Turkers are asked to highlight portions (i.e. *rationales*) of Arabic sentences to justify a dialectal label. The rationales are then incorporated into the training of statistical learners, yielding significant improvements in classification accuracy over corresponding baselines.

Chapter 4

Annotator Rationales for Text Classification

In the previous Chapter, we considered a task, Arabic dialect identification, that we cast as a classic NLP problem (language identification), and solved using a classic supervised learning approach (n-gram language models). The types of annotations we collected were not particularly novel in their type, and the novelty lied in the domain itself and data gathering effort. In this Chapter, we take advantage of crowdsourcing not to create a standard type of annotations for a new task, but to create a whole new **type** of annotations.

Supervised learning algorithms rely on class labels to tease out the important features (and their relative importance). We propose a new framework that can aid supervised learning algorithms, by enriching the traditional annotation procedure to obtain *annotator rationales*. We ask annotators to provide not only **what** the correct answer is, but also **why**. These rationales provide additional hints to the learner that enable it to better generalize to new examples and better learn the true parameters of the underlying model. We review two methods to incorporate such enriched annotations into existing classifiers.[1]

The first method modifies the training objective of support vector machines (Cortes and Vapnik, 1995), by incorporating the intuition that the learner should be less con-

[1]This Chapter is based on Zaidan et al. (2007) and Zaidan and Eisner (2008).

fident of an example's true label if that example's rationales are masked out. Hence, the resulting classifier is one that has a large margin around the decision hyperplane, as in the standard approach, but also one that attempts to maximize the separation between training examples and their corresponding 'contrast' examples, in which the rationales are masked out.

The second method modifies the training objective of a log-linear model, by tying the classifier's parameters to a second model that directly models the rationale annotation process. The intuition is that the rationales are based, in part, on the parameters of the classification model, just like the true class labels are based on those parameters. Therefore, just like class labels can be used to infer the classifier's parameters, so can the rationales.

We present experimental results on a sentiment classification task, that of distinguishing positive movie reviews from negative ones. We then show how rationale data can be collected, via crowdsourcing, for the Arabic dialect identification task of the previous Chapter. In both tasks, we see that using rationales yields significant accuracy improvements over both baselines. A cost analysis shows that the benefit of added rationales justifies the associated cost.

4.1 Annotator Rationales

Annotators play an important role in creating many of the training datasets used in supervised statistical learning, by providing the correct class labels for training examples. This is particularly relevant in NLP given the inherent difficulty of language-related applications. Each training label from by an annotator provides a small piece of evidence that can guide the learner as it attempts to capture the annotator's knowledge and the decision-making process. The hope is that, given a large enough set of labels that cover a diverse enough set of input instances, a learning algorithm will be able to deduce sufficient information and get a clear understanding of the inherent annotator knowledge.

However, a lot of thought goes into the process of making a classification decision

on the annotator's part, and yet they typically only give us a small piece of evidence: their final verdict. It is a testament to the power of learning algorithms that they can learn so well given only a limited view of the annotator's thinking. In short, an annotator usually tells us *what* the right answer is, but has no direct way of communicating *why* it is the right answer, or *how* they reached their decision.

We propose that annotators be given the chance to communicate exactly that 'why' information, by enriching the annotation task so that they provide coarse hints about the reason they chose a particular class label, in addition to providing the label itself. Specifically, we propose that an annotator who is categorizing documents also *highlight relevant portions of the example*, i.e. substrings of the text being classified, that influenced their judgment. We call such clues *rationales*, and they need not correspond to machine learning features. For example, in the task of classifying movie reviews into positive (favorable) or negative (unfavorable) documents, an annotator could highlight the following boldfaced segments as positive rationales:

- **You will enjoy the hell out of** American Pie.
- Fortunately, they **managed to do it in an interesting and funny way**.
- He is **one of the most exciting martial artists on the big screen**, continuing to perform his own stunts and **dazzling audiences** with his flashy kicks and punches.
- The romance was **enchanting**.

and the following segments as negative rationales:

- A woman in peril. A confrontation. An explosion. The end. **Yawn. Yawn. Yawn.**
- When a film makes watching Eddie Murphy **a tedious experience, you know something is terribly wrong**.
- The movie is **so badly put together** that even the most casual viewer may notice the **miserable pacing and stray plot threads**.

Armageddon

This disaster flick is a disaster alright. Directed by Tony Scott (Top Gun), it's the story of an asteroid the size of Texas caught on a collision course with Earth. After a great opening, in which an American spaceship, plus NYC, are completely destroyed by a comet shower, NASA detects said asteroid and go into a frenzy. They hire the world's best oil driller (Bruce Willis), and send him and his crew up into space to fix our global problem.

The action scenes are over the top and too ludicrous for words. So much so, I had to sigh and hit my head with my notebook a couple of times. Also, to see a wonderful actor like Billy Bob Thornton in a film like this is a waste of his talents. The only real reason for making this film was to somehow out-perform Deep Impact. Bottom line is, Armageddon is a failure.

Figure 4.1: An example review, for the 1998 movie *Armageddon*. It is an example of a negative review, as it expresses the reviewer's unfavorable opinion of the movie.

- **Don't go see** this movie.

4.1.1 Why Would Rationales Help?

Why should we expect rationales to be useful? Aren't learning algorithms already quite good at learning from class labels alone? Consider the text in Figure 4.1. This is a movie review expressing an unfavorable opinion, and it is quite easy to see that the review is negative, and there is in fact little doubt that the author disliked the movie quite a bit. But the classification task is not as easy as it might first seem. Consider instead the same task, but in some language other than English: Figure 4.2 shows the same negative review, but translated into Arabic. Presumably, a non-Arabic speaker would not be much more skillful at classifying Arabic documents after only being told that this is a negative review.

This really is the situation from the point of view of the machine learner, and we

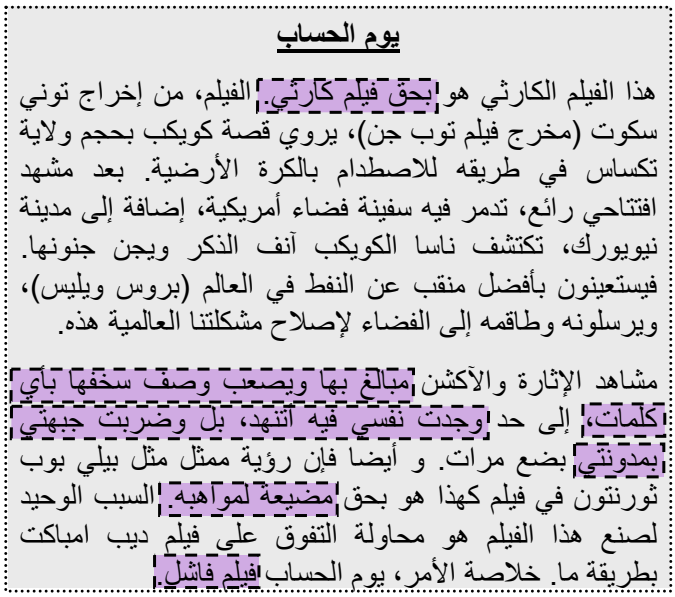


Figure 4.2: The example review from Figure 4.1, translated into Arabic, and annotated with rationales that support a negative class label. Corresponding English rationales are shown in Figure 4.3.

should not expect any machine learner to gain much benefit from a single class label. On the other hand, if certain segments were highlighted as support for the negative classification (e.g. Figure 4.3), a human learner would gain more insight into what makes a document negative. By the same token, a machine learner would also benefit from such information if it were somehow integrated into its learning, and gain more insight into the decision process of the human annotator.

We must keep in mind that we not only want to capture an annotator's knowledge, but we also want to do so effectively. Therefore, one immediate concern about collecting rationales is the additional time required for annotators to highlight them. We note here that most of the extra time needed to collect rationales reflects the need to physically *highlight* them, not necessarily to *find* them. If the annotator is already engaged in determining the correct class label for a document, then they are presumably already identifying the rationales mentally, though not explicitly marking them. Therefore, there is a class label-rationale synergy that allows the annotator to provide the enriched annotation without too much extra time. The synergy can and

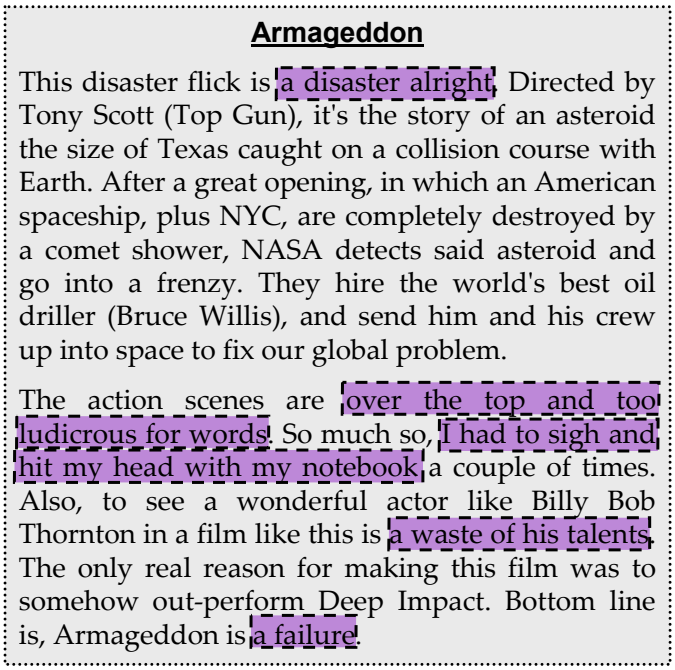


Figure 4.3: The example review from Figure 4.1, annotated with rationales that support a negative class label (as in Figure 4.2).

should be taken full advantage of by designing an efficient and easy-to-use annotation interface, which allows annotators to indicate the rationales easily and quickly.

4.1.2 The Movie Review Polarity Dataset

The snippets above were taken from movie reviews in Pang and Lee's Movie Review Polarity Dataset (Pang and Lee, 2004).[2] The dataset consists of 1,000 positive and 1,000 negative movie reviews obtained from the Internet Movie Database (IMDb) review archive, all written before 2002 by a total of 312 authors, with a cap of 20 reviews per author per category. Pang and Lee have divided the 2,000 documents into 10 folds, each consisting of 100 positive reviews and 100 negative reviews. These gold-standard classifications were derived from the number of "stars" assigned by each review's author. That said, most reviews contain a mix of praise, criticism, and factual description – it is possible for a mostly critical review to give a positive overall recommendation, or vice versa.

Documents in the first nine folds F_0–F_8 of the dataset (1,800 documents) were annotated with rationales that supported the gold-standard classifications by an annotator A0. The last fold, F_9, is used as a test fold in all of our experiments.[3]

A histogram of rationale counts is shown in Figure 4.4. As mentioned above, the rationale annotations were just textual substrings, and the annotator did not need knowledge of the classifier features. Thus, our rationale dataset is useful for other researchers who wish to investigate how rationales could be useful, and they need not be restricted to the same feature set (or learning methods) that we consider here.

We use A0's data to determine the effectiveness of the two methods below (in 4.2 and 4.3), and how they compare to two baseline classifiers. We would also like to know if the method would work well for data by annotators other than A0. To this end, we gathered additional rationale annotations from more annotators. We

[2]Polarity dataset version 2.0.

[3]The test fold was not annotated for rationales, since rationales are not needed at test time. In retrospect, it would have been beneficial if we had collected rationales for F_9's documents as well, since that would have made it possible to conduct e.g. cross-validation experiments, rather than fix the test set.

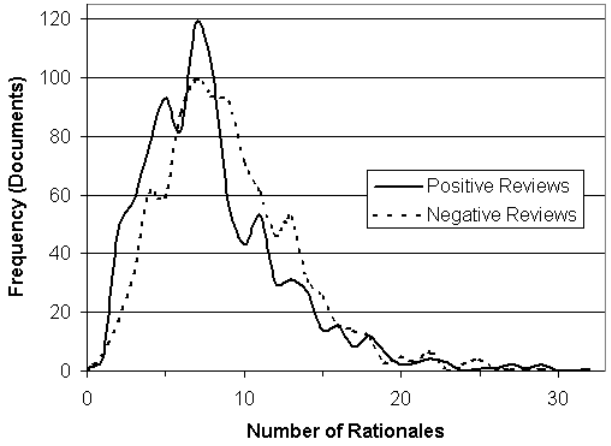


Figure 4.4: Histograms of rationale counts per document (A0's annotations). The overall mean, median, and mode are 8.55, 8, and 7, respectively.

randomly selected 100 reviews and collected class and rationale annotation data from each of six annotators A1–A6. We report results using data from A1–A3, since we used the data from A4–A6 as development data in the early stages of our work.

4.1.3 Data Collection

The annotation involves **boldfacing** the rationale phrases using an HTML editor. The annotators were given guidelines that instructed them to read reviews and justify *why* a review is positive or negative by highlighting rationales for the document's class. The instructions read, in part:

> Each review was intended to give either a positive or a negative overall recommendation. You will be asked to justify why a review is positive or negative. To justify why a review is positive, highlight the most important words and phrases that would tell someone to see the movie. To justify why a review is negative, highlight words and phrases that would tell someone not to see the movie. These words and phrases are called **rationales**.
>
> You can highlight the rationales as you notice them, which should result in several rationales per review. Do your best to mark enough rationales to provide convincing support for the class of interest.

> You do not need to go out of your way to mark everything. You are probably doing too much work if you find yourself going back to a paragraph to look for even more rationales in it. Furthermore, it is perfectly acceptable to skim through sections that you feel would not contain many rationales, such as a reviewer's plot summary, even if that might cause you to miss a rationale here and there.

The last two paragraphs were intended to provide some guidance on how *many* rationales to annotate, encouraging annotators to do their best to mark enough rationales to provide convincing support, but emphasizing that they need not go out of their way to mark everything.

4.1.4 Notation and Features

Each example in the training set will consist of the document itself, its annotated class, and its rationale markup. We denote those components with x, y, and r, respectively. At test time, we will have to predict the y label of a document x, but without the aid of rationales markup for the test example.

We use the same set of binary features as in previous work on this dataset (Pang et al., 2002; Pang and Lee, 2004). Specifically, let $V = \{v_1, ..., v_{17744}\}$ be the set of unstemmed word or punctuation types with count ≥ 4 in the full 2,000-document corpus. Thus, each document is reduced to a 0-1 vector with 17,744 dimensions. The models investigated below are specified by a corresponding weight vector of the same size, with positive weights favoring class label $y = +1$ and discouraging $y = -1$, while negative weights do the opposite. (This weight vector is called $\vec{w}$ in Section 4.2 and called $\vec{\theta}$ in Section 4.3. Both play exactly the same role, but are given different names to agree with the common notation for these different models.)

4.2 Integrating Rationales into a Discriminative Model

One popular approach for text categorization is to use a discriminative model such as a Support Vector Machine (SVM) (e.g. Dumais (1998); Joachims (1998)). An ordinary soft-margin SVM chooses $\vec{w}$ and $\vec{\xi}$ to minimize

$$\frac{1}{2}\|\vec{w}\|^2 + C(\sum_i \xi_i) \tag{4.1}$$

subject to the constraints

$$(\forall i) \quad \vec{w} \cdot \overrightarrow{x_i} \cdot y_i \geq 1 - \xi_i \tag{4.2}$$

$$(\forall i) \quad \xi_i \geq 0 \tag{4.3}$$

where $\overrightarrow{x_i}$ is a training example, $y_i \in \{-1, +1\}$ is its desired classification, and ξ_i is a slack variable that allows training example $\overrightarrow{x_i}$ to miss satisfying the margin constraint if necessary. The parameter $C > 0$ controls the cost of taking such slack, and should generally be lower for noisier or less linearly separable datasets.

4.2.1 Contrast Examples

We propose that SVM training can incorporate annotator rationales as follows. From the rationale annotations on a positive example $\overrightarrow{x_i}$, we will construct one or more "not-quite-as-positive" *contrast examples* $\overrightarrow{v_{ij}}$. In our text categorization experiments below, each contrast document $\overrightarrow{v_{ij}}$ was obtained by starting with the original and "masking out" one or all of the several rationale substrings that the annotator had highlighted (r_{ij}). The intuition is that the *correct* model should be less sure of a positive classification on the contrast example $\overrightarrow{v_{ij}}$ than on the original example $\vec{x_i}$, because $\overrightarrow{v_{ij}}$ lacks evidence that the annotator found significant.

We can translate this intuition into additional constraints on the correct model, i.e. on the weight vector $\vec{w}$. In addition to the usual SVM constraint on positive examples that $\vec{w} \cdot \overrightarrow{x_i} \geq 1$, we also want (for each j) that $\vec{w} \cdot \vec{x}_i - \vec{w} \cdot \overrightarrow{v_{ij}} \geq \mu$, where $\mu \geq 0$ controls the size of the desired margin between original and contrast examples.

We add the *contrast constraints*

$$(\forall i, j) \;\; \vec{w} \cdot (\overrightarrow{x_i} - \overrightarrow{v_{ij}}) \cdot y_i \geq \mu(1 - \xi_{ij}), \tag{4.4}$$

where $\overrightarrow{v_{ij}}$ is one of the contrast examples constructed from example $\overrightarrow{x_i}$, and $\xi_{ij} \geq 0$ is an associated slack variable. Just as these extra constraints have their own margin μ, their slack variables have their own cost, so the objective function of Equation 4.1 becomes

$$\frac{1}{2}\|\vec{w}\|^2 + C(\sum_i \xi_i) + C_{contrast}(\sum_{i,j} \xi_{ij}) \tag{4.5}$$

The parameter $C_{contrast} \geq 0$ determines the importance of satisfying the contrast constraints. It should generally be less than C if the contrasts are noisier than the training examples.

In practice, it is possible to solve this optimization using a standard soft-margin SVM learner without modification to its implementation. Dividing Equation 4.4 through by μ, it becomes

$$(\forall i, j) \;\; \vec{w} \cdot \overrightarrow{x_{ij}} \cdot y_i \geq 1 - \xi_{ij}, \tag{4.6}$$

where $\overrightarrow{x_{ij}} \stackrel{\text{def}}{=} \frac{\overrightarrow{x_i} - \overrightarrow{v_{ij}}}{\mu}$. Since Equation 4.6 takes the same form as Equation 4.2, we simply add the pairs $(\overrightarrow{x_{ij}}, y_i)$ to the training set as *pseudoexamples*, weighted by $C_{contrast}$ rather than C so that the learner will use the objective function of Equation 4.5. In our experiments, we optimize μ, C, and $C_{contrast}$ on held-out data (see 4.4.2).

To allow for a biased hyperplane (i.e. that does not necessarily pass through the point of origin), we use the standard trick of adding a zeroth feature, with a value of 1 for all training examples. That is, the standard constraints become $\vec{w} \cdot (1, \overrightarrow{x_i}) \cdot y_i \geq 1 - \xi_i$ (adding w_0 as a bias term). One subtlety is that this feature should be set to 0 for pseudoexamples, not 1: $\vec{w} \cdot \frac{(1, \vec{x}_i) - (1, \overrightarrow{v_{ij}})}{\mu} \cdot y_i = \vec{w} \cdot (0, \overrightarrow{x_{ij}}) \cdot y_i \geq 1 - \xi_{ij}$.

Figure 4.5 illustrates what our modification means in practice. Whereas the standard SVM only cares about maximizing the margin of the decision hyperplane, the modified SVM also pays attention to the margin separating original examples from their contrast examples. This means that the modified SVM is willing to adjust the decision hyperplane a bit, deviating from the maximum margin and sacrificing a bit

of performance on the training set, in hopes that the adjusted margin can better generalize to unseen examples.

4.3 Integrating Rationales into a Generative Model

We also consider integrating rationales into the training of a generative model. Consider a standard log-linear model, the training of which involves choosing model parameters[4] $\vec{\theta}$ that maximize the following objective function:

$$\prod_{i=1}^{n} p_\theta(y_i \mid x_i) \cdot p_{\text{prior}}(\vec{\theta}) \tag{4.7}$$

This is a standard training method where parameters are chosen that maximize the conditional likelihood of the class labels. The second term of the training criterion imposes a prior on the model parameters, functioning as a regularizer. A reasonable prior is the standard diagonal Gaussian prior (with some tunable variance σ_θ^2), biasing weights in $\vec{\theta}$ towards zero.

We define the classifier p_θ to be a standard log-linear model:

$$p_\theta(y \mid x) \stackrel{\text{def}}{=} \frac{\exp(\vec{\theta} \cdot \vec{f}(x, y))}{Z_\theta(x)} \stackrel{\text{def}}{=} \frac{u(x, y)}{Z_\theta(x)} \tag{4.8}$$

where $\vec{f}(\cdot)$ extracts a feature vector from a classified document, $\vec{\theta}$ are the corresponding weights of those features, and $Z_\theta(x) \stackrel{\text{def}}{=} \sum_y u(x, y)$ is a normalizer.

4.3.1 Modeling Rationales Explicitly

We would like to explicitly *model* rationale annotation as a noisy process that reflects, imperfectly and incompletely, the annotator's internal decision procedure. We incorporate into the training objective of Equation 4.7 an additional term that

[4]The log-linear model's parameters $\vec{\theta}$ play exactly the same role as the SVM's $\vec{w}$.

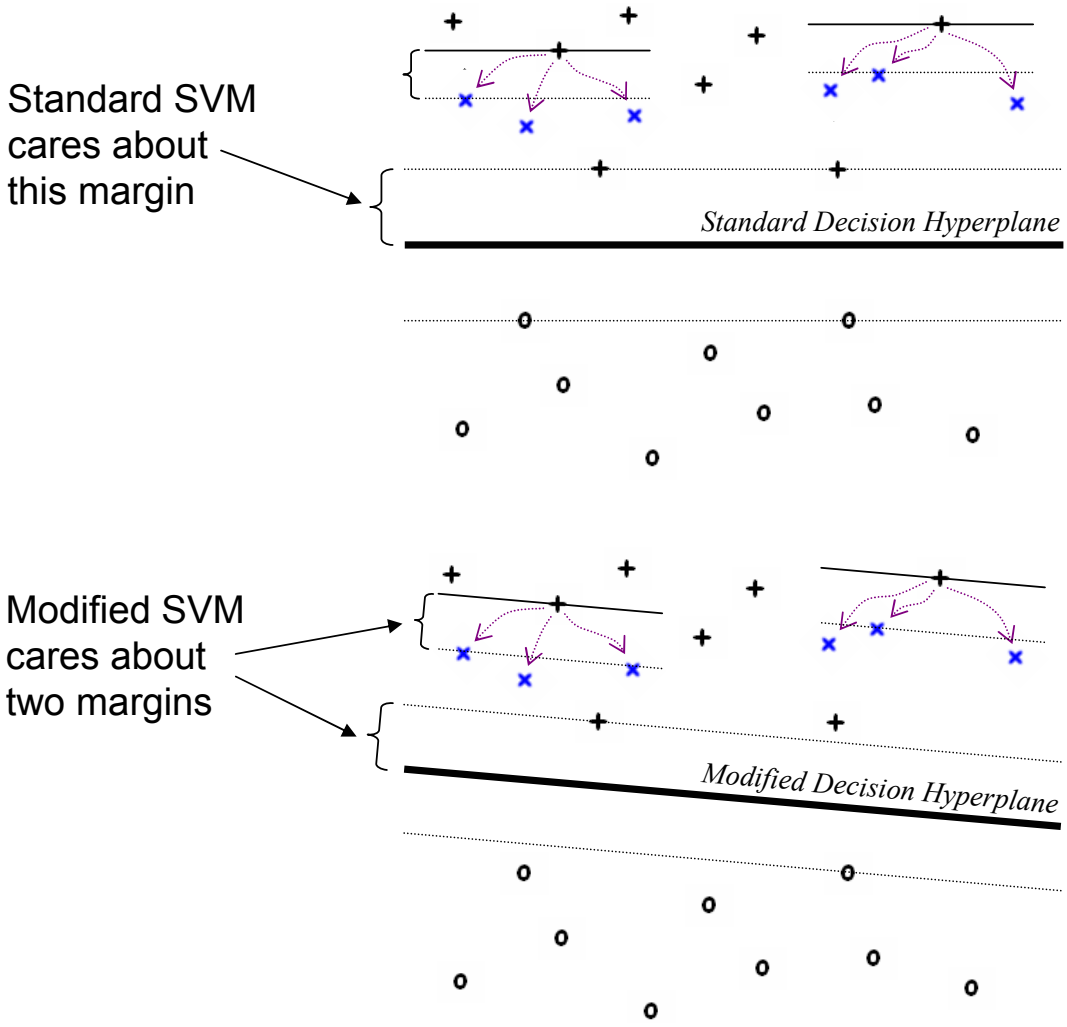


Figure 4.5: An illustration comparing the standard SVM to the modified SVM. The standard SVM attempts to maximize the separation between positive (+) and negative (o) examples. The modified SVM deviates from the maximum-margin hyperplane if that increases the separation between original examples and the corresponding contrast examples (the blue xs).

attempts to *model the rationales as well*:

$$\prod_{i=1}^{n} p_\theta(y_i \mid x_i) \cdot p_\phi(r_i \mid x_i, y_i, \vec{\theta}) \cdot p_{\text{prior}}(\vec{\theta}) \cdot p_{\text{prior}}(\vec{\phi}) \tag{4.9}$$

Here we are trying to model **all** the annotations, both y_i and r_i. The first factor models y_i using an ordinary probabilistic classifier p_θ, while the novel second factor models r_i using some model p_ϕ of how annotators generate the rationale annotations.

We have not yet defined this second model p_ϕ – we do so in the next subsection. The crucial point is that it depends on $\vec{\theta}$ (since r_i is supposed to reflect the relation between x_i and y_i that is modeled by $\vec{\theta}$). As a result, the machine learner has an incentive to modify $\vec{\theta}$ in a way that increases the second factor, even if this somewhat decreases the first factor on training data. That is, a given guess of $\vec{\theta}$ might make Equation 4.7 large, yet accord badly with the annotator's rationales. In that case, the second term of Equation 4.9 will exert pressure on $\vec{\theta}$ to change to something that conforms more closely to the rationales. If the annotator is correct, such a $\vec{\theta}$ will generalize better beyond the training data.[5] The optimization technique for choosing $\vec{\theta}$ and $\vec{\phi}$ is discussed in 4.4.3.

4.3.2 p_ϕ as a Conditional Random Field

The interesting part of our model is $p_\phi(r \mid x, y, \vec{\theta})$, which models the rationale annotation process. *The rationales r reflect $\vec{\theta}$, but in noisy ways.*

The rationales collected in this task are textual segments of a document to be classified. The document itself is a word token sequence $\vec{x} = x_1, ..., x_M$. We encode its rationales as a corresponding tag sequence $\vec{r} = r_1, ..., r_M$, as illustrated in Figure 4.6. Here $r_m \in \{\texttt{I}, \texttt{O}\}$ according to whether the token x_m is **in** a rationale or **outside** all rationales. x_1 and x_M are special boundary symbols, tagged with `O`.

[5] Interestingly, even examples where the annotation y_i is wrong or unhelpful can provide useful information about $\vec{\theta}$ via the pair (y_i, r_i). Two annotators marking the same movie review might disagree on whether it is overall a positive or negative review—but the second factor still allows learning positive features from the first annotator's positive rationales, and negative features from the second annotator's negative rationales.

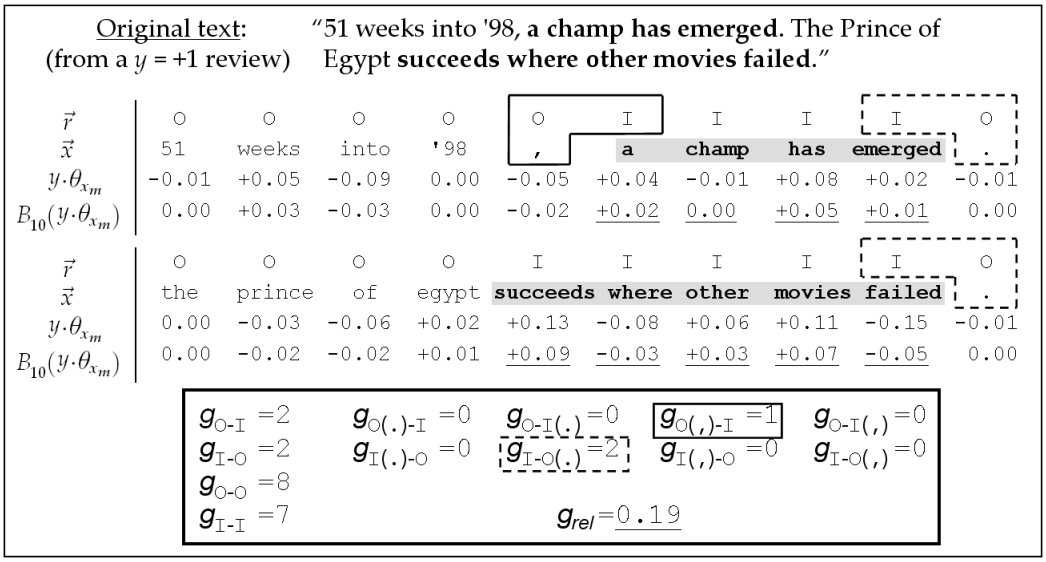


Figure 4.6: Modeling rationales as sequence annotation. The annotator highlighted two textual segments as rationales for a positive class. Highlighted words in $\vec{x}$ are tagged `I` in $\vec{r}$, and other words are tagged `O`. The figure also shows some ϕ-features. For instance, $g_{O(,)\text{-}I}$ is a count of `O-I` transitions that occur with a comma as the left word. Notice also that g_{rel} is the sum of the underlined values.

We model the full tag sequence $\vec{r}$ at once using a conditional random field (Lafferty et al., 2001). A CRF is just another conditional log-linear model:

$$p_\phi(r \mid x, y, \vec{\theta}) \stackrel{\text{def}}{=} \frac{\exp(\vec{\phi} \cdot \vec{g}(r, x, y, \vec{\theta}))}{Z_\phi(x, y, \vec{\theta})} \stackrel{\text{def}}{=} \frac{u(r, x, y, \vec{\theta})}{Z_\phi(x, y, \vec{\theta})} \tag{4.10}$$

where $\vec{g}(\cdot)$ extracts a feature vector, $\vec{\phi}$ are the corresponding weights of those features, and $Z_\phi(x, y, \vec{\theta}) \stackrel{\text{def}}{=} \sum_r u(r, x, y, \vec{\theta})$ is a normalizer.

As usual for linear-chain CRFs, $\vec{g}(\cdot)$ extracts two kinds of features: first-order emission features that relate r_m to (x_m, y, θ), and second-order transition features that relate r_m to r_{m-1} (although some of these also look at x). The emission features account for the relationship between a word's label and its polarity: Does r faithfully signal the features of θ that strongly support classifying x as y? The transition features are independent of $\vec{\theta}$, and they account for the relationship between a word's label and the labels of the surrounding words: How frequent and how long are typical rationales?

Put more simply, the former says r_m is `I` in a positive document because x_m has a highly positive θ. The latter says r_m is `I` simply because it is next to another `I`. Thanks to the transition features, we do not need to posit high θ features to explain every word in a rationale. The highlights on these words are "explained away" as having been made only because this annotator prefers not to end a rationale in mid-phrase, or prefers to sweep up several nearby words with a single long rationale rather than many short ones.

4.3.2.1 p_ϕ's Emission Features

Given $(\vec{x}, y, \vec{\theta})$, let us say that a unigram $w \in \vec{x}$ is **relevant**, **irrelevant**, or **anti-relevant** if $y \cdot \theta_w$ is respectively $\gg 0$, ≈ 0, or $\ll 0$. That is, w is relevant if its presence in x strongly supports the annotated class y, and anti-relevant if its presence strongly supports the opposite class $-y$. We would like to learn the extent ϕ_{rel} to which annotators try to *include* relevant unigrams in their rationales, and the (usually lesser) extent ϕ_{antirel} to which they try to *exclude* anti-relevant unigrams. This will help us infer $\vec{\theta}$ from the rationales.

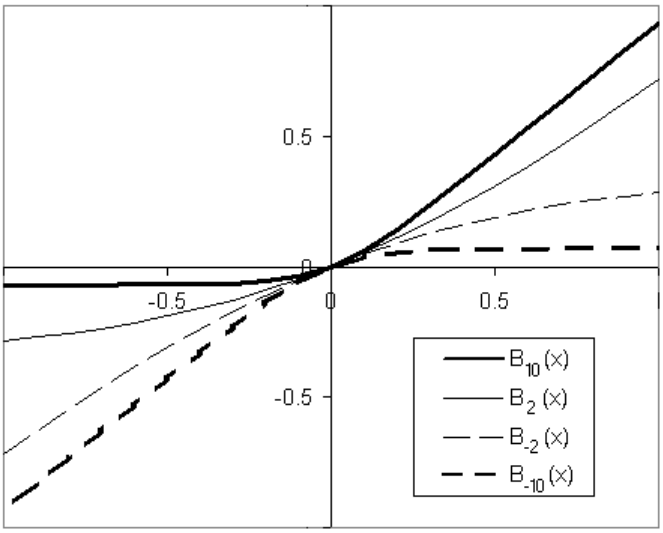


Figure 4.7: The function family B_s of Equation 4.13, shown for $s \in \{10, 2, -2, -10\}$. Note how for $s > 0$, B_s is close to zero for negative values of x, which is suitable for g_{rel}. For $s < 0$, B_s is close to zero for positive values of x, which is suitable for g_{antirel}.

The details are as follows. ϕ_{rel} and ϕ_{antirel} are the weights of two emission features extracted by $\vec{g}$:

$$g_{\text{rel}}(\vec{x}, y, \vec{r}, \vec{\theta}) \stackrel{\text{def}}{=} \sum_{m=1}^{M} I(r_m = \texttt{I}) \cdot B_{10}(y \cdot \theta_{x_m}) \tag{4.11}$$

$$g_{\text{antirel}}(\vec{x}, y, \vec{r}, \vec{\theta}) \stackrel{\text{def}}{=} \sum_{m=1}^{M} I(r_m = \texttt{I}) \cdot B_{-10}(y \cdot \theta_{x_m}) \tag{4.12}$$

where $I(\cdot)$ denotes the indicator function, returning 1 if its argument is true, and 0 otherwise. Relevance and negated anti-relevance are respectively measured by the differentiable nonlinear functions B_{10} and B_{-10}, which are defined by

$$B_s(a) = (\log(1 + \exp(a \cdot s)) - \log(2))/s \tag{4.13}$$

and graphed in Figure 4.7. Sample values of B_{10} and g_{rel} are shown in Figure 4.6.

How does this work? The g_{rel} feature is a sum over all unigrams in the document $\vec{x}$. It does not fire strongly on the irrelevant or anti-relevant unigrams, since B_{10} is close to zero there. But it fires positively on relevant unigrams w if they are tagged

with `I`, and the strength of such firing increases approximately linearly with θ_w. Since the weight $\phi_{\text{rel}} > 0$ in practice, this means that raising a relevant unigram's θ_w (if $y = +1$) will proportionately raise its log-odds of being tagged with `I`. Symmetrically, since $\phi_{\text{antirel}} > 0$ in practice, lowering an anti-relevant unigram's θ_w (if $y = +1$) will proportionately lower its log-odds of being tagged with `I`, though not necessarily at the same rate as for relevant unigrams.[6]

There are other types of features that we could in theory use to enrich the feature set but purposely chose not to. Should ϕ also include traditional CRF emission features, which would recognize that particular words like `great` tend to be tagged as `I`? Such features would undoubtedly do a better job modeling the rationales and hence increasing the objective of Equation 4.9. However, we avoid features that provide a lot of information regarding the textual content, because, crucially, our *ultimate* goal is not to model the rationales, but to recover the classifier parameters θ. Thus, if `great` indeed tends to be highlighted in positive documents, then the model should not be permitted to explain this directly by increasing some feature $\phi_{\texttt{great}}$, but only indirectly by boosting $\theta_{\texttt{great}}$. We therefore permit our rationale model to consider only the two emission features g_{rel} and g_{antirel}, which see the words in $\vec{x}$ only through their θ-values.

4.3.2.2 p_ϕ's Transition Features

Annotators highlight more than just relevant words; they tend to mark full phrases, though perhaps taking care to exclude anti-relevant portions. p_ϕ models these phrases' shape, via several transition features. Most important are the 4 traditional CRF tag transition features $g_{\texttt{O-O}}, g_{\texttt{O-I}}, g_{\texttt{I-I}}, g_{\texttt{I-O}}$. For example, $g_{\texttt{O-I}}$ *counts* the number of `O-I` transitions in $\vec{r}$ (see Figure 4.6). Other things equal, an annotator with high $\phi_{\texttt{O-I}}$ tends to produce a large number of rationales. If $\phi_{\texttt{I-I}}$ is high, the annotator's rationales tend to be long phrases (i.e. including many irrelevant words around or between the relevant ones).

[6] Splitting into separate features g_{rel} and g_{antirel} allows us to model annotators who try to include highly relevant features, and exclude highly anti-relevant ones, to different degrees.

We also learn more refined versions of these features, which consider how the transition probabilities are influenced by the punctuation and syntax of the document $\vec{x}$ (independent of $\vec{\theta}$). These refined features are more specific and hence more sparsely trained. Their weights reflect deviations from the simpler, "backed-off" transition features such as $g_{\texttt{O-I}}$. (Again, see Figure 4.6 for examples.)

Conditioning on the left/right word. A feature of the form $g_{t_1(v)\text{-}t_2}$ is specified by a pair of tag types $t_1, t_2 \in \{\texttt{I}, \texttt{O}\}$ and a vocabulary word type v. It counts the number of times a t_1–t_2 transition occurs in $\vec{r}$ conditioned on v appearing as the first of the two word tokens where the transition occurs. Similarly, a feature $g_{t_1\text{-}t_2(v)}$ has v appearing as the second of the two word tokens where the transition occurs.

More formally, for any (t_1, t_2, v): l

$$g_{t_1(v)\text{-}t_2}(\vec{x}, y, \vec{r}, \vec{\theta}) = g_{t_1(v)\text{-}t_2}(\vec{x}, \vec{r}) = \sum_{m=1}^{M-1} I(r_m, r_{m+1} = t_1, t_2) \cdot I(x_m = v) \quad (4.14)$$

$$g_{t_1\text{-}t_2(v)}(\vec{x}, y, \vec{r}, \vec{\theta}) = g_{t_1\text{-}t_2(v)}(\vec{x}, \vec{r}) = \sum_{m=1}^{M-1} I(r_m, r_{m+1} = t_1, t_2) \cdot I(x_{m+1} = v) \quad (4.15)$$

Our experiments include $g_{t_1(v)\text{-}t_2}$ and $g_{t_1\text{-}t_2(v)}$ features that tie `I-O` and `O-I` transitions to the 4 most frequent punctuation marks v (comma, period, `?`, `!`). We also include five features that tie `O-I` transitions to the words *no*, *not*, *so*, *very*, and *quite*, since in our development data, those words were more likely than others to start rationales.[7]

Conditioning on syntactic boundary. We parsed each rationale-annotated training document[8] and marked each word bigram x_1-x_2 with three nonterminals: N_{End} is the nonterminal of the largest constituent that contains x_1 and not x_2, N_{Start} is the nonterminal of the largest constituent that contains x_2 and not x_1, and N_{Cross} is the nonterminal of the *smallest* constituent that contains both x_1 and x_2.

For a nonterminal N and pair of tag types (t_1, t_2), we define three features, $g_{t_1\text{-}t_2/\text{E}=N}$, $g_{t_1\text{-}t_2/\text{S}=N}$, and $g_{t_1\text{-}t_2/\text{C}=N}$, which count the number of times a t_1-t_2 transition

[7] These are the function words with count ≥ 40 in a random sample of 100 documents, and which were associated with the `O-I` tag transition at more than twice the average rate.

[8] We use the Collins parser (Collins, 1999). Each document has a single parse tree, whose root is `DOC`, with each sentence being a child of `DOC`.

occurs in $\vec{r}$ with N matching the N_{End}, N_{Start}, or N_{Cross} nonterminal, respectively. Our experiments include these features for 11 common nonterminal types N (`DOC`, `TOP`, `S`, `SBAR`, `FRAG`, `PRN`, `NP`, `VP`, `PP`, `ADJP`, `QP`).

4.4 Experimental Results

4.4.1 Experimental Design

Our learning curves show accuracy after training on $T < 9$ folds (i.e. $200T$ documents), for various values of T. The reported accuracy for training on T folds is the average of 9 different experiments with different (though overlapping) training sets that cover folds 0–8:

$$\frac{1}{9}\sum_{i=0}^{8} acc(F_9 \mid h, F_{i+1} \cup \ldots \cup F_{i+T}) \tag{4.16}$$

where F_j denotes the jth fold, and $acc(Z \mid h, Y)$ means classification accuracy on the set Z after training on Y, with system hyperparameters h.

To evaluate whether two different training methods A and B gave significantly different average-accuracy values, we used a paired permutation test (generalizing a sign test). For each of the 200 test examples in fold 9, we measured (a_i, b_i), where a_i (resp. b_i) is the number of the 9 training sets under which A (resp. B) classified the example correctly. The p value is the probability that the absolute difference between the average-accuracy values would reach or exceed the observed absolute difference, namely $|\frac{1}{200}\sum_{i=1}^{200}\frac{a_i - b_i}{9}|$, if each (a_i, b_i) had an independent 1/2 chance of being replaced with (b_i, a_i), as per the null hypothesis that A and B are indistinguishable.

4.4.2 Tuning the Modified SVM's Hyperparameters

Each training example $\overrightarrow{x_i}$ is normalized to unit length before adding it to the training set.[9] Given a training document, we create several contrast documents, each by deleting exactly one rationale substring from the training document. Converting documents to feature vectors, we obtained an original example $\overrightarrow{x_i}$ and several contrast examples $\overrightarrow{v_{i1}}, \overrightarrow{v_{i2}}, \ldots$.[10] Again, our training method required each original document to be classified more confidently (by a margin μ) than its contrast documents.

We transformed this problem to an SVM problem and used the SVM^{light} software (Joachims, 1999) for training and testing, using the default linear kernel. As for the hyperparameters $h = (C, \mu, C_{contrast})$, for any given value of T we chose hyperparameters that maximize the following cross-validation performance:

$$h^* = \operatorname*{argmax}_h \sum_{i=0}^{8} acc(F_i \mid h, F_{i+1} \cup \ldots \cup F_{i+T}) \tag{4.17}$$

We used a simple alternating optimization procedure that begins at $\theta_0 = (1.0, 1.0, 1.0)$ and cycles repeatedly through the three dimensions, optimizing along each dimension by a local grid search with resolution 0.1. Of course, when training without rationales, we did not have to optimize μ or $C_{contrast}$.

[9] The vectors are normalized *before* prepending the 1 corresponding to the bias term feature (see end of subsection 4.2.1).

[10] The contrast examples were not normalized to precisely unit length, but instead were normalized by the same factor used to normalize $\overrightarrow{x_i}$. This conveniently ensured that the pseudoexamples $\overrightarrow{x_{ij}} \stackrel{\text{def}}{=} \frac{\vec{x}_i - \overrightarrow{v_{ij}}}{\mu}$ were sparse vectors, with 0 coordinates for all words not in the jth rationale.

4.4.3 Optimization of the Generative Model's $\vec{\theta}$ and $\vec{\phi}$

To train our generative model, we use L-BFGS to locally maximize the log of the objective function of Equation 4.9:

$$\sum_{i=1}^{n} \log p_\theta(y_i \mid x_i) - \frac{1}{2\sigma_\theta^2}\|\theta\|^2$$
$$+C(\sum_{i=1}^{n} \log p_\phi(r_i \mid x_i, y_i, \theta)) - \frac{1}{2\sigma_\phi^2}\|\phi\|^2 \quad (4.18)$$

Note that we define p_{prior} from Equation 4.9 to be a standard diagonal Gaussian prior, with variances σ_θ^2 and σ_ϕ^2 for the two sets of parameters. We choose $\sigma_\theta^2 = \frac{1}{5}$ based on the development data of A4–A6. As for σ_ϕ^2, different values did not affect the results, since we have a large number of {I,O} rationale tags to train relatively few ϕ weights, so we simply use $\sigma_\phi^2 = 1$ in all of our experiments.

Note the new C factor in Equation 4.18. Our initial experiments showed that optimizing Equation 4.18 without C led to an increase in the likelihood of the rationale data at the expense of classification accuracy, which degraded noticeably. This is because the second sum in Equation 4.18 has a much larger magnitude than the first: in a set of 100 training documents, it models around 74,000 binary {I,O} tags, versus the one hundred binary class labels. While we are willing to reduce the likelihood of the training classifications (the first sum) to a certain extent, focusing too much on modeling rationales (the second sum) is clearly not our ultimate goal, and so we optimize C on development data to achieve some balance between the two terms of Equation 4.18. In our experiments, we use $C = \frac{1}{300}$, again based on the development data of A4–A6.[11]

We take an iterative optimization approach for setting $\vec{\theta}$ and $\vec{\phi}$:

1. Initialize $\vec{\theta}$ to maximize Equation 4.18 but with $C = 0$ (i.e. based only on class data).

[11] C also balances our confidence in the classifications y against our confidence in the rationales r; either may be noisy.

2. Fix $\vec{\theta}$, and find $\vec{\phi}$ that maximizes Equation 4.18.

3. Fix $\vec{\phi}$, and find $\vec{\theta}$ that maximizes Equation 4.18.

4. Repeat 2 and 3 until convergence.

The L-BFGS method requires calculating the gradient of the objective function. The partial derivatives with respect to components of $\vec{\theta}$ and $\vec{\phi}$ involve calculating expectations of the feature functions, which can be computed in linear time (with respect to the size of the training set) using the forward-backward algorithm for CRFs.

4.4.4 Results

Figure 4.8 shows learning curves for training set sizes up to 1,600 documents for four methods: a log-linear baseline, an SVM baseline, the discriminative method of Section 4.2, and the generative method of Section 4.3. A data point in the figure reports an averaged accuracy over 9 experiments with different subsets of the training set (see 4.4.1). The top two curves indicate significant improvements in accuracy over the respective baselines when rationales are introduced using our two proposed approaches. Further, our generative method outperforms[12] our masking SVM method, which starts with a slightly better baseline classifier (an SVM) but incorporates the rationales more crudely.

Examining the learned weights $\vec{\phi}$ gives insight into annotator behavior. High weights include `I-O` and `O-I` transitions conditioned on punctuation, e.g. $\phi_{\texttt{I(.)-O}} = 3.55$,[13] as well as rationales ending at the end of a major phrase, e.g. $\phi_{\texttt{I-O/E=VP}} = 1.88$. The large emission feature weights, e.g. $\phi_{\text{rel}} = 14.68$ and $\phi_{\text{antirel}} = 15.30$, tie rationales closely to θ values, as hoped. For example, in Figure 4.6, the word $w =$ `succeeds`, with $\theta_w = 0.13$, drives up $p(\texttt{I})/p(\texttt{O})$ by a factor of 7 (in a positive document) relative to a word with $\theta_w = 0$.

[12] Differences are not significant at sizes 200, 1,000, and 1,600.
[13] When trained on folds F_4–F_8 with A0's rationales.

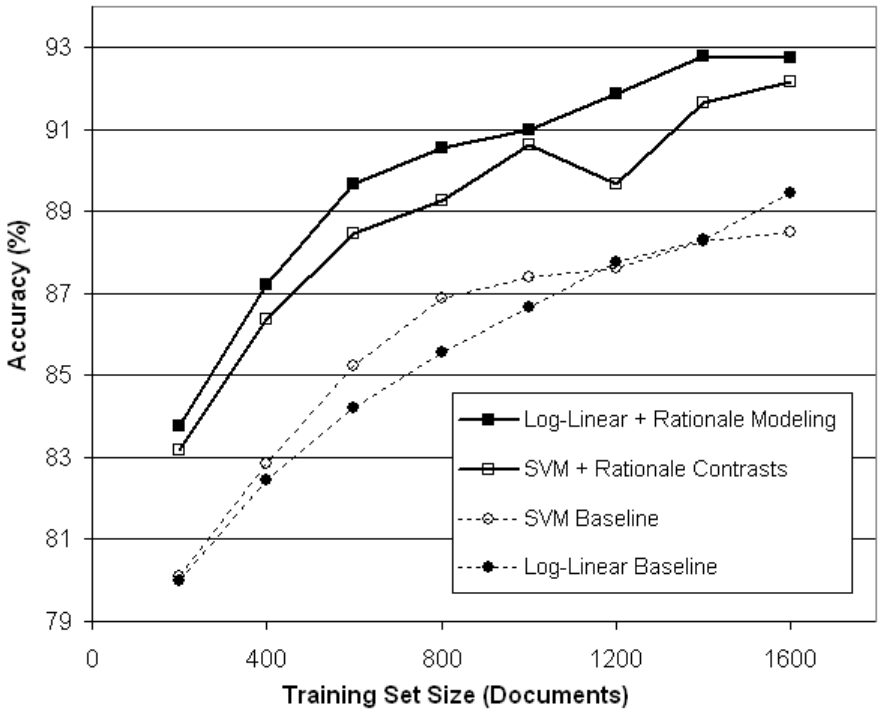


Figure 4.8: Classification accuracy curves for the 4 methods: the two baseline learners that only use class data, and the two learners that also utilize rationale annotations provided by annotator A0.

	Size (documents)	A0	A1	A2	A3
SVM baseline	100	72.0	72.0	72.0	70.0
SVM+contrasts	100	*75.0*	*73.0*	*74.0*	*72.0*
Log-linear baseline	100	71.0	*73.0*	71.0	70.0
Log-linear+rats	100	**76.0**	**76.0**	**77.0**	**74.0**
SVM baseline	20	63.4	*62.2*	60.4	62.6
SVM+contrasts	20	*65.4*	*63.4*	*62.4*	**64.8**
Log-linear baseline	20	63.0	*62.2*	60.2	62.4
Log-linear+rats	20	**65.8**	**63.6**	**63.4**	**64.8**

Table 4.1: Accuracy rates using each annotator's data. In a given column, a value in *italics* is not significantly different from the highest value in that column, which is **boldfaced**. Each size=20 result is an average over 5 experiments.

To show that the results generalized beyond annotator A0, we performed the same comparison for three additional annotators, A1–A3, each of whom provided class and rationale annotations on a small 100-document training set, and each of whom had their own $\vec{\phi}$.[14] Table 4.1 shows improvements for our methods over the baselines at 2 training set sizes.

Note that we learned different ϕ parameters for each annotator, reflecting their different annotation styles. We carried out two final sets of experiments to examine the variation in annotation style among the different annotators.

We first asked: What happens if we use one annotator's ϕ to learn another annotator's θ? For instance, say we already have ϕ_{A3}. When it is time to find θ_{A4}, can we just use ϕ_{A3}? How does that affect performance, compared to finding θ_{A4} jointly with A4's own ϕ?

The results are in Table 4.2. We see that, in general, attempting to find an annotator's $\vec{\theta}$ using another annotator's $\vec{\phi}$ usually leads to a slight drop in performance. This can be seen by noticing that moving from a diagonal entry in Table 4.2 to another entry within the same row usually results in a drop in performance.

[14] A0 and A1–A3 were blind test data—we developed our method (e.g. ϕ-features) using data from annotators A4–A6.

	ϕ_{A0}	ϕ_{A1}	ϕ_{A2}	ϕ_{A3}	Baseline
θ_{A0}	**76.0**	*73.0*	*74.0*	*73.0*	71.0
θ_{A1}	*73.0*	**76.0**	*74.0*	*73.0*	*73.0*
θ_{A2}	*75.0*	73.0	**77.0**	*74.0*	71.0
θ_{A3}	**74.0**	*71.0*	*72.0*	**74.0**	70.0

Table 4.2: Accuracy rate for an annotator's θ (rows) obtained when using some other annotator's ϕ (columns). Notice that the diagonal entries and the baseline column are taken from rows of Table 4.1 (size=100).

	ϕ_{A0}	ϕ_{A1}	ϕ_{A2}	ϕ_{A3}	Trivial model
$-L(r_{A0})$	**0.073**	0.086	*0.077*	0.088	0.135
$-L(r_{A1})$	0.084	**0.068**	*0.071*	**0.068**	0.130
$-L(r_{A2})$	0.088	0.084	**0.075**	0.085	0.153
$-L(r_{A3})$	0.058	**0.044**	*0.047*	**0.044**	0.111

Table 4.3: Cross-entropy per tag of rationale annotations $\vec{r}$ for each annotator (rows), when predicted from that annotator's $\vec{x}$ and $\vec{\theta}$ via a possibly different annotator's ϕ (columns). For comparison, the trivial model is a bigram model of $\vec{r}$, which is trained on the target annotator but ignores $\vec{x}$ and $\vec{\theta}$. 5-fold cross-validation on the 100-document set was used to prevent testing on training data.

We also asked: how well would one annotator's ϕ model another annotator's rationales? In Table 4.3, we report the cross-entropy of each annotator's rationale data when modeled by each of the annotators' ϕ. We see that modeling an annotator's rationales is best done using that annotator's own ϕ, as seen by noticing that moving from a diagonal entry to another entry within the same row usually results in an increase in cross-entropy (i.e. decrease in likelihood). Using a different annotator's ϕ still handily beats a trivial model that is based only on the fraction of `I` tags in the data.

بس ياريت الريس يهتم بالشئون الداخلية للبلد شوية و يصدر قانون محاسبة الوزراء و محاكمتهم
لكل صاقطه لاقطه الدعوة لا يخلو رجل بالمرأه الا كان الشيطان ثالثهما اترك الهرج بس واتباع الهوي يا محرر الحر
الحمدلله ع العقل !
ولكن الظاهرأن العقلية في النادي لم تعد تفكر إلا باحراز الرباعية وهذا مش منطقي.
اخس على هيك رجال
متى يتعلم حكامنا من الرجال أمثال أردوجان ؟ ؟ و النبي مش مكسوفين من أنفسكم ؟ ؟ ؟

Figure 4.9: Examples of Arabic sentences, with dialectal portions highlighted. Those examples, among others, were shown to annotators working on the rationale annotation task.

4.5 Case Study II: Dialect Identification

We applied our rational annotation approach to the Arabic dialect identification task of the previous Chapter, and collected rationales for the class labels on the dialectal sentences. In essence, we ask annotators to identify which portion of the sentence make it dialectal. The instructions read, in part:

> This task is for Arabic speakers who understand the different local Arabic dialects. Below, we show you some Arabic sentences. **Your task is to highlight the dialectal parts** within each sentence.

Annotators were also shown example Arabic sentences with their dialectal portions highlighted; Figure 4.9 shows some of those examples.

We group the sentences that have already been identified as dialectal into groups of 10, with each group appearing as a single screen on MTurk. For quality control purposes, we also insert 2 additional sentences that had previously been identified as *non*-dialectal, for a total of 12 sentences per HIT. This resulted in 4,461 HITs, each of which was completed by three annotators, with a reward of $0.15 per assignment.

The total cost of the annotation effort was $2,007.45 for rewards and $200.75 for Amazon's fees, for a total of $2,208.20. In addition to the 13,383 approved assignments, 2,091 assignments were rejected, resulting in a rejection rate of 13.5%. 198 workers participated in the effort, with 98 of them performing at least 10 HITs, and 35 of them performing at least 100 HITs. The most prolific annotator performed 2,415 HITs. The effort lasted about 50 days, requiring 633.49 man hours, for an hourly rate (excluding Amazon fees) of $3.17.

4.5.1 Results

Overall, annotators highlighted 56% of the words in the dialectal sentences, with 91% of sentences having at least one highlighted word. In the MSA control sentences, those percentages were expectedly much lower: only 8.0% of words were highlighted, with only 26% of sentences having at least one highlighted word. For each annotator, we measured their word-level precision and word-level recall, against other annotators, and found that there is a strong inverse correlation between the two.

This is a reflection of a spectrum of annotation style: high precision indicates a 'conservative' annotator that highlights relatively few words, resulting in lower recall, while high recall indicates a 'liberal' annotator that highlights many words, resulting in lower precision. Although some annotators do have both high precision and high recall (and some annotators have both low precision and low recall), the linear relationship is quite strong. The overall inter-annotator agreement rate is 73.9%. Figure 4.10 reflects the precision-recall trade-off.[15]

We applied the approaches outlined in Section 4.2 and Section 4.3 to the dialect

[15]Note that our generative model is able to account for this variance in annotation behavior, since the transition features account for the length and number of rationales.

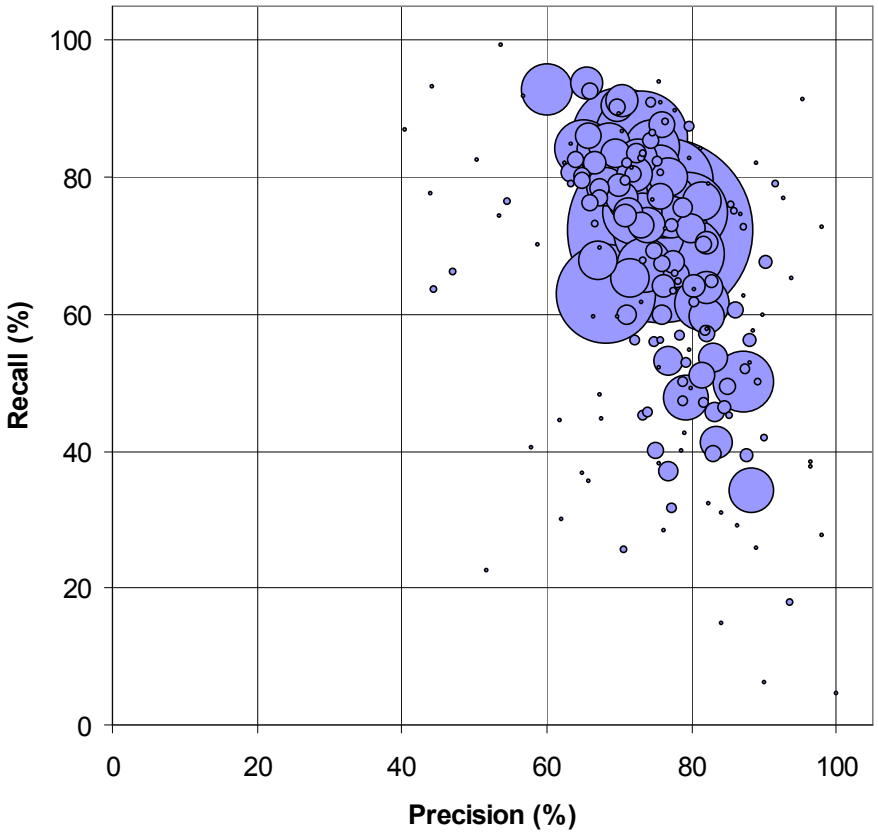


Figure 4.10: A precision-recall scatter plot for Turkers working on the dialect rationale annotation task. Each annotator is represented by a single data point, the size of which reflects the number of HITs performed by that annotator. Precision/recall values are not computed against a gold standard; rather, they are computed against other annotators.

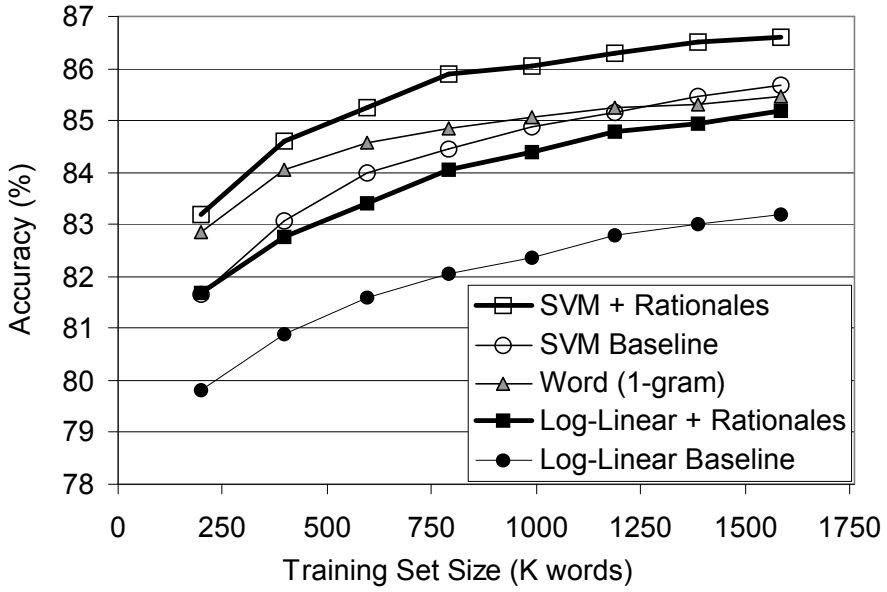


Figure 4.11: Learning curves for the MSA vs. dialect classification task for the methods presented in this Chapter, showing that incorporating rationales yields significant improvements in classification accuracy. For comparison, the top curve of Figure 3.9 is repeated here, to contrast this Chapter's methods with the language-model-based approach.

identification task, using the crowdsourced rationale annotations. Figure 4.11 shows learning curves for the two baselines and the two rationale-aided methods. As with the movie review classification task, incorporating rationales into the training of statistical learners yields significant improvements in classification accuracy rates.

4.6 Cost Analysis

We have seen that incorporating rationales into the training of classifiers yields significant accuracy improvements. However, since rationales require a more complex form of annotation, they are more costly to collect than plain class labels. Therefore, an important question is whether or not the improvement in performance justifies the additional cost.

Let's first discuss the movie review task, and analyze the benefit of added ratio-

nales. In a prior timing study of several annotators (Zaidan et al., 2007), we found that supplying the additional rationale annotations required roughly twice as much time as supplying class labels alone. This is a relatively low overhead, considering the large number of provided rationales (an average of 5–11 rationales per document, versus a single class label per document). Furthermore, the annotation interface required annotators to manually boldface the rationale segments in addition to selecting them, significantly increasing the time requirement. Nevertheless, the learning curves in Figure 4.8 indicate that the benefit of annotating T documents with rationales is as effective as annotating between $2T$ and $3T$ documents with class labels.[16] Therefore, even with a pessimistic estimate of the time needed to supply additional rationales, our results provide evidence that the added benefit of rationales justifies the cost of collecting them.

That said, we imagine that an annotator would become extremely efficient at providing class labels as they annotate more reviews, more so than they would at providing rationales. This would make the added cost of annotating rationales more difficult to justify. We note here that the movie review task seems to be almost a worst-case scenario for the annotation of rationales. Determining a reviewer's opinion is often quite easy, and may not even require reading the entire movie review. On the other hand, it is time-consuming to annotate several rationale segments across an entire review text.

This is in contrast to the dialect identification task, where training examples are much shorter, and our annotation interface automatically highlighted selected segments. Our experiments on dialect identification provide further and stronger evidence of the benefit of rationales. Let's examine the learning curves for the two SVM methods in Figure 4.11.[16] In particular, let's determine how much training data is needed for each SVM variant to achieve a particular accuracy rate. In Table 4.4, we consider the accuracy rates at the 2$^{\text{nd}}$, 3$^{\text{rd}}$, and 4$^{\text{th}}$ points of the modified SVM's learning curve, and estimate where those accuracy rates would be on the *standard* SVM's learning curve. We see strong evidence that the added benefit of rationales

[16]Note that in both classification tasks of this Chapter, the impact of adding rationales is more dramatic for the log-linear model than for the SVM.

Accuracy rate	Modified SVM		Standard SVM		Cost saving
	Training (words)	Cost	Training (words)	Cost	
84.60%	400k	$330	890k	$490	32.7%
85.25%	600k	$495	1,237k	$680	27.2%
86.30%	1,200k	$990	2,375k (est.)	$1,300	23.8%

Table 4.4: An annotation cost comparison between the standard SVM and the modified SVM, at three accuracy rates. The accuracy rates correspond to the 2nd, 3rd, and 4th points of the modified SVM's learning curve in Figure 4.11, along with the rough equivalents from the *standard* SVM's learning curve. The training size for the standard SVM in the last line (2,375k words) assumes that the rate of improvement after training size 1600k is the same as between sizes 800k–1600k. The cost columns assume a cost of 1.5¢ per segment for the modified SVM (to obtain rationales), and 1.0¢ per segment for the standard SVM (to obtain class labels). The average segment length is assumed to be 18.3 words.

justifies the cost, with significant savings in annotation cost to obtain comparable performance.

When deciding whether or not to collect rationales, another important factor besides cost is the amount of available training data. In both tasks examined in this Chapter, it was important to conduct the cost analysis above to justify collecting rationales, since it would be easy to simply collect more training examples and have them annotated with class labels. But consider a scenario where there is very little training data to begin with. For example, we might wish to classify movie or product reviews in a language other than English, for which little examples exist. We may be interested in the identification of very rare languages or dialects, for which no large dataset exists on the scale of the AOC. In such instances, the amount of training examples is a limiting factor, while annotation cost may not be. We would be interested in getting the most out of every training example, which would justify collecting rationales even if they were much more costly than class labels alone (assuming adding rationales improves performance).

4.7 Related Work

Our rationales resemble "side information" in machine learning, i.e. supplementary information about the target function that is available at training time. Past work generates such information, by *automatically* transforming the training examples in ways that are expected to preserve or alter the classification (Abu-Mostafa, 1995), sometimes having to manually annotate the extra examples (Kuusela and Ocone, 2004). Our approach differs because a human helps to generate the virtual examples.

Our work tries to extract extra knowledge from annotators by having them provide rationales. One could instead ask annotators to examine or propose some *features* instead of rationales. In document classification, Raghavan et al. (2006) show that feature selection by an oracle could be helpful, and that humans are both rapid and reasonably good at distinguishing highly useful n-gram features from randomly chosen ones. Druck et al. (2008) show annotators some features f from a fixed set, and ask them to choose a class label y such that $p(y \mid f)$ is as high as possible. Haghighi and Klein (2006) do the reverse: for each class label y, they ask the annotators to propose a few "prototypical" features f such that $p(y \mid f)$ is as high as possible.

In contrast, our approach does not force the annotator to evaluate the importance of features individually, nor in a global context outside any specific document, nor even to know the learner's feature space. Crucially, by not committing to a particular feature set at annotation time, our annotation setup does not restrict subsequent research on the dataset. Also, annotating features is only appropriate when the feature set can be easily understood by a human. n-gram features are simple enough to digest by an annotator, but this is not always the case, and it would be hard for annotators to understand and evaluate a complex syntactic configuration in NLP or a convolution filter in machine vision, for instance.

Besides work presented in this Chapter, several papers by other researchers have been published that make use of annotator rationales. Arora and Nyberg (2009) use our movie review rationales to perform feature selection for the same classification task. This was particularly important in their case since the set of features was very large (structured features based on POS tags and dependency relations). Yessenalina

et al. (2010) explore methods to automatically generate annotator rationales for the movie review task. Interestingly, those automatically-generated rationales perform at a comparable level to human-generated ones. That said, they use a previously-trained off-the-shelf opinion finder tool and polarity lexicons to construct those rationales.

Abedin et al. (2011) introduce *residue examples* (as opposed to our *contrast* examples), which are examples that contain only the rationales of a document. They are concerned with a multi-class problem that requires taking the text of an aviation incident report, and assigning one of 14 *shaping factors* to it: underlying causes of the incident (e.g. `Preoccupation`, `Resource Deficiency`). Annotators were asked to provide textual rationales that supported the shaping factor they chose, and those rationales were incorporated in the training of an SVM as in Section 4.2.[17] They managed to improve the classifier's F-score from 42.2 to 49.5.

Donahue and Grauman (2011) apply rationales in the *visual* domain, considering image classification tasks of scene identification and human face attractiveness. They consider two forms of rationales for images: visual rationales and textual rationales. In the first kind, an annotator specifies a spatial region of interest (using a polygon-drawing tool), such as marking a sink in a 'kitchen' image. In the second, an annotator is provided with a list of attributes about the image, such as "smiling" or "bushy eyebrows" for a facial image[18], and the annotator indicates which of those attributes most affected their class label. It is worth noting that they use MTurk to collect most of the rationale annotations.

4.8 Conclusion

We have presented a new type of annotations that we call **rationales**, and two machine learning algorithms that exploit them and improve existing baselines. Annotating rationales does not require the annotator to think about the feature space or know anything about it, making annotation easier and more flexible, and preserving

[17] They use a one-vs-all approach (Hsu and Lin, 2002) to generalize an SVM to a multi-class classifier.

[18] The attributes themselves are predicted automatically by separately-trained models.

the reusability of the annotated data.

The overhead of rationale annotation could be reduced by asking annotators to provide rationales only for the rarer classes, as focusing on where the data is sparsest could provide extra guidance where it is most needed. Another possibility is the collection of rationales via game play. In the visual domain, the Peekaboom game (von Ahn et al., 2006) was in fact built to elicit such approximate yet relevant regions of images. The abovementioned work by Donahue and Grauman relies on a similar annotation setup.

Besides taking advantage of crowdsourcing, this Chapter revolved around the second major theme of this thesis, that of creating and collecting *new types of annotations*, a theme that will appear again in later Chapters. This theme is tightly connected to that of crowdsourcing, since new types of data could be difficult to collect in a classical setting, for instance because large amounts of data are needed (as in the RYPT metric of the next Chapter). Also, the low overhead associated with crowdsourcing makes it much easier to collect small amounts of data necessary to conduct pilot proof-of-concept studies, the results of which can lead to further data collection, whether crowdsourced or not.

Chapter 5

Crowdsourcing Manual Evaluation of Machine Translation Systems

Great strides have been made in the field of Machine Translation (MT) since the days of Weaver (1949) and Bar-Hillel (1951). MT has established itself as one of the most visible and active fields of computational linguistics research. This success is in no small part due to the rise of *statistical* MT, starting with the work at IBM Research (Brown et al., 1990, 1993). Another factor that aided the advancement of MT research was the adoption of principled evaluation paradigms and metrics. While much of MT evaluation relies on a set of *automatic* metrics, *manual* evaluation of output still plays an important role. This Chapter will examine how crowdsourcing could aid in two methods for manual evaluation of MT output.[1]

The first method is human-targeted translation edit rate (HTER), of Snover et al. (2006), as we attempt to obtain good estimates of HTER by crowdsourcing the editing task to non-trained individuals. The results point out that our setup of the editing task, where a particular HTER-ranking of documents is desired, makes it difficult to perform consistently across untrained editors.

The second method is RYPT, a new metric that we introduce, in which annotators judge the acceptability of individual constituents in the output, and those judgments

[1]The first part of this Chapter is mostly based on Zaidan and Callison-Burch (2009), and the second part is based on Zaidan and Callison-Burch (2010).

Urdu: امریکی سائنسدانوں کی یہ تحقیق چوہوں پر کئے گئے تجربات کے بعد سامنے آئی ہے۔

English 1: This finding by American scientists has emerged as a result of experiments performed on rats.

English 2: This research of American scientists has come to light after experiments done on mice.

English 3: This research by American scientists has come out as a result of experiments conducted on mice.

English 4: This study of US scientists is the result of tests carried out on mice.

Figure 5.1: An Urdu source sentence with multiple correct English translations, produced by four different professional translators.

used to evaluate the sentence as a whole. Crowdsourcing comes into play as it allows the fast collection of those judgments, taking advantage of the **crowd** in crowdsourcing. We discuss ways to make RYPT a hybrid metric that is semi-automatic, by having it rely on a database of human judgments that can be reused to score new candidate translations, dramatically decreasing the amount of judgments that need to be collected. This in turn makes it a suitable candidate for use when tuning system parameters in the MERT phase.

5.1 Background: MT Evaluation

Research in machine translation (MT) requires the ability to quantify the output quality of a system. In simpler tasks like classification, measures such as accuracy and precision/recall are logical choices for evaluation, as they are easy to measure, and can distinguish even highly-similar systems from each other. In MT, a measure such as accuracy cannot be calculated even with a set of pre-existing reference translations, because a source sentence usually has more than one correct translation (e.g. Figure 5.1). Therefore, a strict measure that only rewards perfect matches is not applicable, and a fine-grained measure is required. In early work by Brown et al. (1990), only a small fraction of translations (5%) were exact matches, but 48% were found acceptable by human judges. Later evaluation setups (White and O'Connell, 1994; LDC, 2005) explored having annotators judge the *fluency* (i.e. grammar) and *adequacy* (i.e. meaning) of a translation, using numerical scales (Figure 5.2).

	Fluency: How do you judge the fluency of this translation? It is:
5	Flawless English
4	Good English
3	Non-native English
2	Disfluent English
1	Incomprehensible

	Adequacy: How much of the meaning expressed in the gold-standard translation is also expressed in the target translation?
5	All
4	Most
3	Much
2	Little
1	None

Figure 5.2: Fluency and adequacy scales in the LDC specification (LDC, 2005).

5.1.1 Automatic Evaluation Metrics

Manual evaluation of MT output was quickly observed to have several issues. First, judges must be given explicit training to be able to make meaningful judgments. Second, and in spite of training, different annotators could have very different ways of evaluating the same sentence, causing low inter-annotator agreement. Even *intra*-annotator agreement is difficult to achieve reliably (Snover et al., 2006; Callison-Burch et al., 2007). Most critically, manual evaluation is quite costly, requiring a considerable amount of time to actually collect judgments, as well as financial resources to pay annotators. Therefore, manual evaluations have sometimes been restricted to small test sets, and the evaluation phase is usually a drawn-out process.

It became clear that MT research could greatly benefit from automating the evaluation process, which would yield more consistent results and allow larger test sets (White, 2000). The adoption of *automatic metrics* to score output against reference translations meant that even small research teams could make measurable contributions to MT research. Indeed, the design of automatic metrics has itself become an important research topic within MT (Nießen et al., 2000; Doddington, 2002; Papineni et al., 2002; Turian et al., 2003; Banerjee and Lavie, 2005; Kauchak and Barzilay, 2006; Snover et al., 2006, 2009; Cer et al., 2010; Dorr, 2010; Liu et al., 2011), with entire workshops and shared tasks devoted to it (Przybocki et al., 2008; Callison-Burch et al., 2011)

Two prominent metrics are BLEU and TER, which have been shown to correlate well with human judgment of translation quality (Coughlin, 2003; Snover et al., 2006), and have been largely adopted by the community as standard evaluation metrics. We will briefly discuss them here, given their relevance to later parts of the Chapter.

5.1.1.1 The BLEU Metric

The BLEU metric[2] (Papineni et al., 2002) measures the proportion of word n-grams in the candidate translation that also appear in the reference translation. The BLEU score of a candidate c, measured against a reference translation r is:

$$BLEU(c,r) = BP(len(c), len(r)) \cdot \exp(\sum_{n=1}^{4} \frac{1}{4} \log p_n),$$

where p_n is the n-gram precision[3], and BP is a multiplicative brevity penalty in $[0.0, 1.0]$ meant to penalize short outputs (Figure 5.3), in order to discourage improving precision at the expense of recall. Note that the BLEU score is basically a similarity measure over strings, with higher values indicating higher translation quality. It falls in $[0.0, 1.0]$, but is usually reported as a 'score' in $[0, 100]$, with a BLEU of 0.345 reported as 34.5 (or, less often, as 34.5%).

BLEU is, by far, the most reported and cited metric in the literature. This is both due to historical reasons, as BLEU is one of the earliest metrics designed specifically for MT, as well as efficiency, as computing the BLEU score of a translation is linear in the length of the sentence.

5.1.1.2 The TER Metric

The TER metric[4] (Olive, 2005) is similar to word error rate (WER), and counts the number of edit actions required to transform a candidate translation into a reference translation. An edit action is the insertion, deletion, or substitution of a word, as in

[2] BLEU stands for **Bi**lingual **E**valuation **U**nderstudy.

[3] *Modified* precision, rather, which is based on clipped n-gram counts. This is relevant when an n-gram appears within the candidate more times than within any reference.

[4] According to Olive, TER stands for **T**ranslation **E**rror **R**ate, though the E could also stand for **E**dit (see Snover et al. (2006), footnote 2).

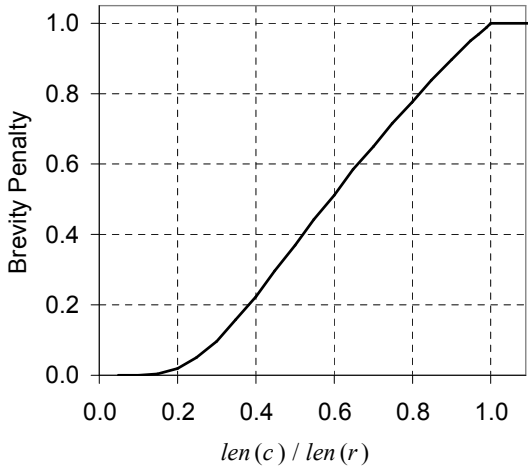


Figure 5.3: BLEU's brevity penalty as a function of candidate length $len(c)$ and reference length $len(r)$: $\exp(1 - \frac{len(r)}{len(c)})$ for $len(c) < len(r)$, and simply 1.0 otherwise.

WER. In addition, TER considers the shift of a sequence of words to be a single edit action. Assuming all edit actions have equal costs, TER is defined as:

$$TER(c, r) = \frac{\#\text{ edits needed for } c \to r}{\text{sentence length}} = \frac{\#\text{ ins} + \#\text{ dels} + \#\text{ subs} + \#\text{ shifts}}{len(r)},$$

where we note that the edit count is normalized by the length of the reference, not the candidate translation (or the edited version of it). Note that TER is an (asymmetrical) distance measure over strings, with lower values indicating higher translation quality. Note also that TER could have values exceeding unity if the candidate translation is particularly bad and longer than the reference. Like BLEU, it is often reported as a percentage score, with a TER of 0.501 reported as 50.1 or 50.1%.

Computing the number of WER-like edits takes quadratic time, and the addition of phrase shifts makes TER computation an NP-complete problem (Lopresti and Tomkins, 1997; Shapira and Storer, 2002). A heuristic search algorithm is usually used, meaning that reported TER scores are usually *approximate*, especially in the case of long sentences. This is in contrast to BLEU, which could be computed exactly.

5.1.2 Criticisms of Fully-Automatic Scoring

Despite the advantages of automatic metrics, they have a number of problematic aspects. Those issues stem from the requirement for a human-produced reference translation. The problem is not necessarily in obtaining a reference translation, but that it would merely be a single example of an acceptable translation, out of many possibilities. Hence, a good translation produced by an MT system might be penalized for differences in word choice and word order that are, in fact, acceptable. Metrics like Meteor (Banerjee and Lavie, 2005; Denkowski and Lavie, 2011) and TERp (Snover et al., 2009) (based on BLEU and TER, respectively) attempt to accommodate synonyms and reward inexact (stem) matches. However, such metrics require additional linguistic resources to operate, such as synonym and paraphrase tables.

Furthermore, a number of problematic aspects of BLEU in particular have been pointed out. Chiang et al. (2008) show there are real-life scenarios where the BLEU score behaves in a counter-intuitive manner. Callison-Burch et al. (2006) point out that BLEU tends to underestimate the output quality of rule-based systems, favoring phrase-based systems. These concerns raise doubts regarding the adequacy of automatic scoring as a proxy for human judgment. Manual evaluation, despite its inconvenience, is still the most trustworthy and sought-after type of evaluation.

5.1.3 Crowdsourcing Manual Evaluation

Crowdsourcing lends itself as a natural option for conducting manual evaluation, given the expected cost saving, and the parallelization speedup. In each of the next two subsections of the Chapter, we examine a method for manual evaluation, and discuss how crowdsourcing could be useful in applying that evaluation method. First we consider the task of editing MT output, for the purposes of computing HTER document ranking, as in the GALE evaluation procedure. We then introduce a new manual evaluation metric, RYPT, that relies on collecting acceptability judgments on individual components within the candidate translation, and propose that the metric could be incorporated in the MERT phase of the MT training pipeline.

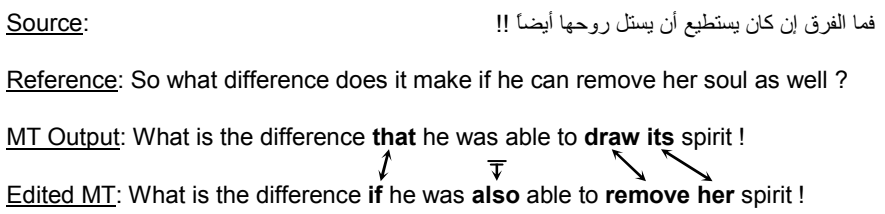


Figure 5.4: An example of minimally editing an MT output to match a pre-existing reference in meaning. The targeted editing requires only 4 actions, whereas matching the provided reference would require many more edits.

5.2 The HTER Metric

A realistic outlook for MT's benefit to automating translation might be as a *first pass* that is followed by an editing phase by a human. An MT system would be considered effective if its output requires a minimum amount of editing, since that reflects the amount of effort needed to make the translation acceptable. This makes TER a reasonable metric for MT evaluation, since it reflects the number of edits needed to make the MT output acceptable.

As with other automatic metrics, TER focuses too much on a single reference (or a few references) produced independently from the MT output. To address this issue, the GALE project relies on *human-targeted* translation edit rate (HTER), wherein the MT output is **scored against a post-edited version of itself**. Editors fluent in the target language are given the candidate translation and the pre-existing ('untargeted') reference, and they are instructed to make necessary changes to the candidate translation such that the edited version is an acceptable translation. Care is taken by the editors to make the smallest number of edits necessary, tolerating abbreviations, alternative word choices, and paraphrases. Figure 5.4 illustrates the minimal-editing principle: only 4 edit actions are required to correct the MT output, yet a full 14 edits are required to transform it to the provided NIST reference.[5]

Therefore, manual evaluation was a core component of evaluating GALE participants. Moreover, GALE's Go/No-Go criteria placed an emphasis on performing well

[5]Given the reference translation is 15 words long, HTER would be $\frac{4}{15} = 26.7\%$, whereas TER would be $\frac{14}{15} = 93.3\%$!

across all documents, and teams were judged by their output quality on the *tail documents*, i.e. documents with the highest HTER scores. Therefore, it was advantageous for a research team to evaluate their system using HTER, or at least determine the ranking of the documents according to HTER, for purposes of error analysis.

This posed a serious problem for system developers. Even though an annotation tool existed to aid editors in their task (e.g. displaying a self-updating edit count while editing is in progress), HTER was not an easy metric to calculate, as it requires hiring and training human editors. The time to perform the editing itself was also too prohibitively long for rapid development. For these reasons, participating teams in GALE resorted to BLEU and TER as proxies for HTER. While they correlate positively with HTER, these metrics are nowhere near perfect (Figure 5.5).

5.2.1 Crowdsourcing Editing

As discussed above, the main obstacles to performing HTER-like evaluation are the need to train the editors, and the cost associated with the editing effort itself, in terms of time and money. To make the process more feasible, we propose here that the editing task can be delegated to non-trained editors instead. The main idea is to mimic the real-world HTER setup, by supplying workers with the MT output that needs to be edited. The worker is also given a human reference (produced independently from the MT output), and the instructions ask the worker to modify the MT output to match the human reference in meaning and grammaticality, with the following guidelines:

> Your task: apply **as few edits as possible** to the "machine translation" so that it **matches the meaning** of the "human translation" and is **good English**.
>
> Guidelines:
>
> - Once edited, the text must have the same meaning as the human translation.
> - Once edited, the text must be in intelligible English.
> - *It is very important to us* that you **use as few edits as possible**. In some instances, no edits are needed.

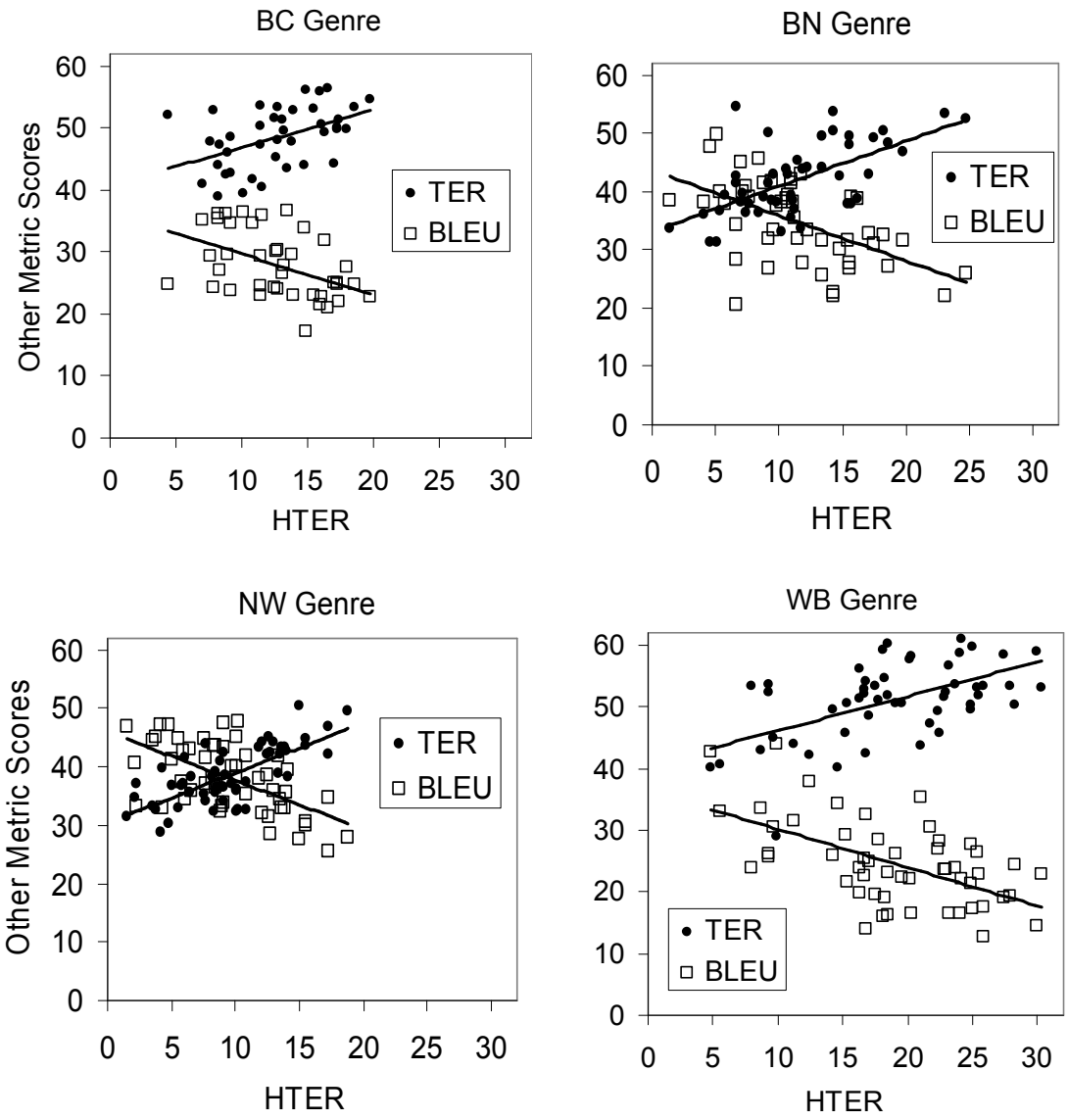


Figure 5.5: A scatter plot of documents' TER and BLEU scores vs. HTER score.

Figure 5.6 shows the annotator interface populated with some actual examples, with editing in progress. The submitted edited sentences can then be used as the references for calculating HTER. Under this setup, a competent and diligent worker would be able to closely match the editing behavior of the professionally trained editor. In the remainder of this section, we will investigate this setup of the editing task using real data from the GALE project, and evaluate the editing quality and document ranking prediction ability using crowdsourced edits.

5.2.2 Datasets

We obtained the unedited output of IBM's Rosetta team submitted to GALE's Arabic-to-English evaluation (Phase 3). The MT output consisted of the submitted translations of 2,153 Arabic sentences[6], spread across 195 documents in 4 different genres: broadcast conversations (BC), broadcast news (BN), newswire (NW), and weblogs (WB). Table 5.1 gives a summary of each of the four genres. Following submission to DARPA, this MT output was HTER-scored by human editors that were hired and trained by the LDC. Hence, each sentence in the MT output had a corresponding edited version of itself, provided by the LDC editor. We will consider those edits to be the gold-standard when evaluating the crowdsourced edits.

The 2,153 sentences were divided into groups of up to 15 sentences each, such that each group contained sentences from the same document, and a document was split into groups of as equally sized as possible. (E.g. a 30-sentence document is divided into two groups of 15, and a 31-sentence document is split into two groups of 10 and one group of 11.) This resulted in 225 groupings, each presented in a single screen, with an average of 9.6 sentences/screen.

We had each screen completed by 5 distinct Turkers, for a total of 10,765 post-edited sentences, provided by 466 distinct workers. With a reward of $0.25 per HIT, the total cost was $309.38 ($281.25 for rewards, and $28.13 for Amazon's commission), at a rate of about 35 post-edited sentences per dollar (or 2.9 pennies per sentence).

[6]The actual sentence count is 2,245, but 92 of those 'sentences' (almost all in the BC genre) were trivial to edit, such as a single period token, so we excluded them from the editing phase.

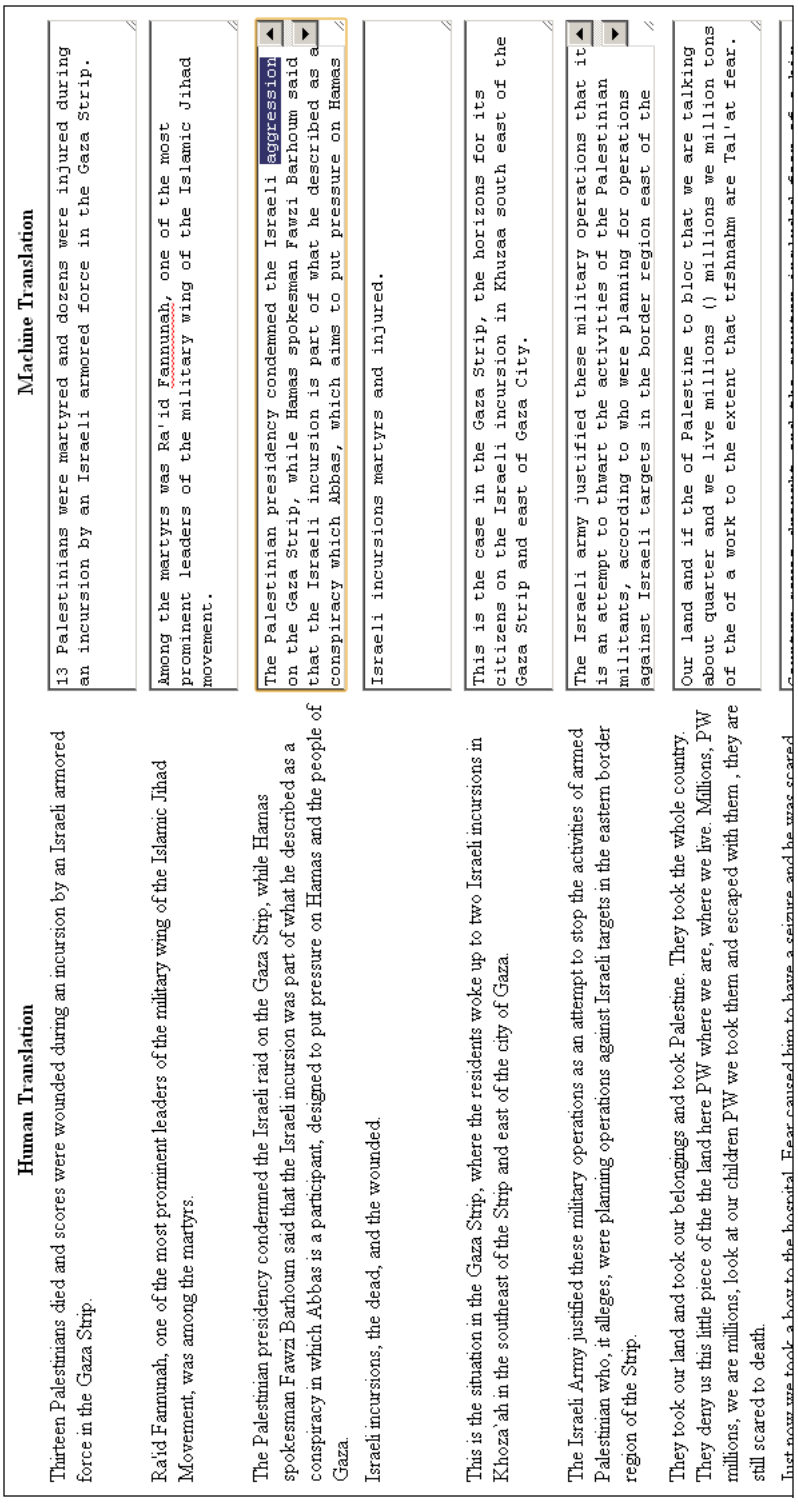


Figure 5.6: The interface for the editing task. The text boxes on the right are populated with the MT output, which are edited by the Turker using the reference translations on the left for guidance.

Genre	# Documents	# Sentences	Sentences/document	Words/sentence	Ave. doc. HTER	Ave. doc. TER
BC	40	632	15.8	28.3	12.7%	48.5%
BN	48	461	9.6	36.1	11.5%	42.1%
NW	54	470	8.7	39.5	9.5%	38.6%
WB	53	590	11.1	31.6	18.9%	51.1%
ALL	195	2,153	11.0	33.3	13.2%	44.9%

Table 5.1: Summary statistics for each genre in the dataset. The HTER values are based on the LDC editor's submissions, whereas the TER values are based on the pre-existing NIST references.

5.2.3 Turkers' Editing Behavior

Analyzing the submitted edits in aggregate gives an insight into how different annotators approach the editing task differently. Figures 5.7 and 5.8 quantify each editor's approach using the edit rate between their submissions and each of the original MT output (Turker Edit Rate) and the provided NIST reference (Turker to NIST Reference Edit Rate). We also take into account the edit rate of the professional editor, which is simply the HTER scores.

In Figure 5.7, we investigate the relationship between a Turker's edit rate (x-axis) and their edit rate from the NIST reference (y-axis). Note that in the latter case we use the "edit rate" as a distance metric, and not to imply that editors usually edit the NIST reference. That said, the plots paint the editor pool as a relatively diverse set of annotators. In particular, most editors fall above the $x = y$ line, indicating they attempt to perform minimal editing on the MT output. The few who fall below the $x = y$ line use the NIST reference as a starting point instead of the MT output. In some cases, it is indeed reasonable to edit the provided reference if the MT output is of particularly low quality. However, this most likely indicates an editor who was not

performing the task properly, especially if they completed a relatively large number of HITs.

In Figure 5.8, the y-axis represents the edit rate of the LDC editor. First note that the variance along that axis is noticeably lower than across the Turkers' edit rates. This is not surprising considering that the HTER values are, after all, based on a single editor's actions, whereas the worker pool within any given genre consists of 100+ editors. It is also notable that the overall edit rate of Turkers is quite close to the overall edit rate of the professional editor.

The editing behavior quantified in Figures 5.7 and 5.8 indicate that the Turkers are, in general, following the guidelines of the editing task correctly. The next subsection investigates whether or not this would help predict the document HTER-ranking.

5.2.4 Experiments

We investigate different methods of using the Turker edits to predict HTER. In particular, we are interested in how well we can predict the *ranking* of documents according to HTER. To measure the quality of a predicted ranking, we use Spearman's rank correlation coefficient, ρ, where we first convert the raw scores into ranks, and then use the following formula to measure correlation:

$$\rho(X,Y) = 1 - \frac{6 \cdot \Sigma_{i:1..n}((rank(x_i) - rank(y_i))^2)}{n(n^2-1)}$$

where n is the number of documents being scored, and each of X and Y is a vector of n (predicted) HTER scores. Notice that values for ρ range from -1 to $+1$, with $+1$ indicating perfect rank correlation, -1 perfect inverse correlation, and 0 no correlation. That is, for a fixed X (the LDC-based HTER score vector), the best-correlated Y is that for which $\rho(X,Y)$ is highest.

5.2.4.1 HTER Predictors

Once we have collected edits from the Turkers, how could we predict HTER from them? We could take the **minimum edit count**, if we could assume that all Turkers

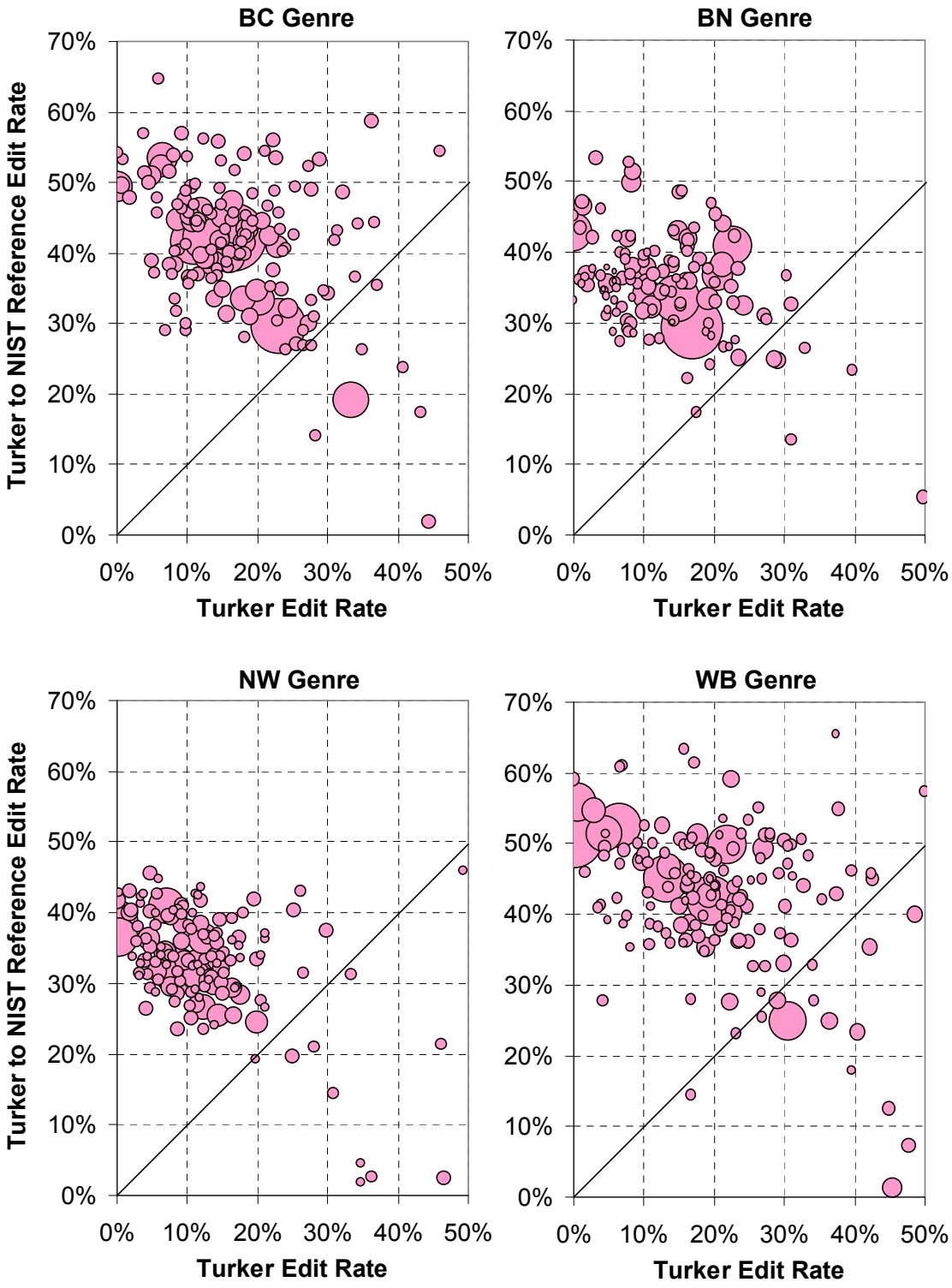


Figure 5.7: Scatter plots of Turker's edit rate (on the MT output) vs. the rate required to produce the NIST reference from their submissions. Each data point represents a single Turker, and its size reflects the number of sentences edited by that Turker.

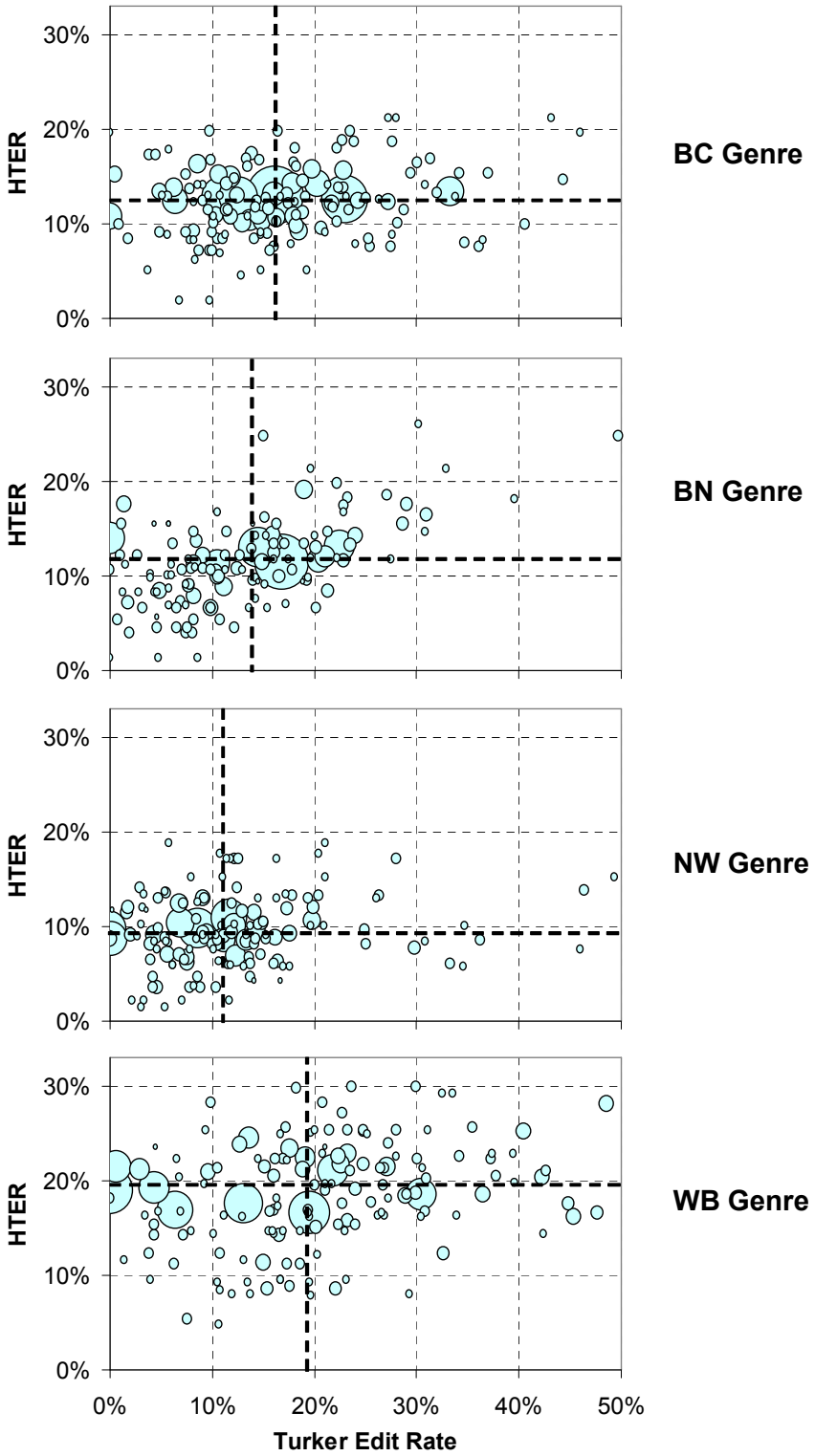


Figure 5.8: Scatter plots of Turker's edit rate (on the MT output) vs. the rate of the LDC editor (on the same MT output). The dashed lines represent the overall edit rate for the Turkers and for the LDC editor.

are doing the task faithfully (and doing it adequately). In this case, the editor making the least amount of edit actions is the one best following the provided instructions. Since this approach would be vulnerable to a single poor editor (e.g. who is submitting the MT output as is), another approach is to take the **average edit count**. This is the equivalent of taking a majority vote in discrete labeling tasks, which is quite effective in a crowdsourced setting (e.g. Sheng et al. (2008); Snow et al. (2008)).

We report results for each genre individually, since the genres vary quite a bit in difficulty. The ranking within each genre is also important since the GALE evaluation treats performance in each genre separately. Table 5.2 summarizes the results across the four genres. We also show the results of two oracle scenarios. The **sentence-level oracle** picks, for each individual sentence, the editor with the closest edit count to the LDC editor *on that sentence*. The **editor-level oracle** picks the editor whose *overall* edit rate (across all sentences) is closest to the LDC editor's edit rate.[7] Furthermore, we have available the edits of a **second LDC editor**, which would allow us to quantify inter-annotator agreement of professional editors, and hence establish a realistic expectation from the non-trained editors.

Overall, averaging the collected answers performs better than taking the minimum, which is not surprising. That said, the results show that it is quite difficult to achieve high performance, and, in all but the BN genre, the automatic metric baselines outperform the above two methods. Another discouraging finding is that the editor-level oracle lags significantly behind the sentence-level oracle, and does not reach the level of the second LDC editor (except in the WB genre).

It is worth noting here the performance of the second LDC editor in the BC and WB genres, which seems a bit unimpressive in absolute terms (0.638 and 0.607). Rather than indicating the lack of skill or effort on part of either LDC editor, this is strong evidence that the editing behavior will vary quite a bit across editors, even the highly-trained professional ones, whose editing is considered the gold-standard. This in turn warrants at least questioning the validity of this evaluation setup as a Go/No-Go criterion for a project on the scale of GALE.

[7] An editor's edit rate for each genre is computed separately.

	BC	BN	NW	WB
Turk Oracle (Sentence)	0.920	0.931	0.945	0.927
Turk Oracle (Editor)	0.534	0.786	0.634	0.662
Second LDC Editor	0.638	0.810	0.734	0.607
TER	**0.479**	0.560	**0.685**	0.473
BLEU	0.470	0.523	0.551	**0.513**
Average Turk Edits	0.395	**0.644**	0.310	0.380
Minimum Turk Edits	0.342	0.592	0.221	0.363
Random Turker	-0.070	0.357	0.127	0.323

Table 5.2: Document ranking correlation (Spearman's ρ) for the different HTER-predictors, across the four genres. The HTER values to be predicted are based on the gold-standard edits by the first LDC editor. A boldfaced value indicates the best value in a column (within the bottom part of the table). In all but the BN genre, automatic metric baselines outperform Turker editors.

5.2.4.2 Editor Calibration

The averaging method above treated all editors equally, and simply took the arithmetic mean of their edit count, giving the editor equal weights. If we can assume the presence of some gold-standard data for some of the sentences, it would be prudent to assign a weight to each editor that would quantify how well we expect them to match the professional's edit rate (and, in turn, the document ranking based on it).

To that end, we can treat a small portion of the sentences edited by a Turker as a *calibration set*. On that portion, we determine how closely the Turker matches the LDC editor, and weight them accordingly when predicting the number of edits of the rest of that editor's sentences. The assigned weight is a function of the difference in the two edit rates, centered around zero (i.e. maximum weight is given to the editor whose edit rate is closest to the professional's).

Figure 5.9 shows the effectiveness of the calibration method across different sizes of the calibration data. We show performance both when a **weighted average** is taken, as well as when only the **top-rated** editor (i.e. whose edit rate is closest to the professional's) is considered.

The experiments show a mixed set of results, though performance is improved over equal-weight averaging. Our method outperforms the TER baseline in the BN and WB genres, using as little as 10%–20% of the data as a calibration set. The WB results in particular are quite encouraging, at least relative to the other methods, as our method matches the ranking correlation of the second LDC editor. On the other hand, the results in the BC and NW genres are not as encouraging, especially in the BC genre, where 30% of the data is needed to even achieve $\rho = 0.50$, a relatively weak correlation. In NW, the TER baseline is simply so strong, that even the editor-level oracle seems unimpressive.

We observe here that the genre where Turkers could be most useful seems to be the genre that is most difficult for MT systems. The WB genre, involving the translation of blog content, was the most difficult for MT systems to translate, as shown by the high editing rate needed for correction (18.9%; see Table 5.1). In that genre, the TER baseline and the second LDC editor do not perform as well as in the news genre NW, which required the *least* amount of editing to correct the MT output.

To conclude, the editing task may seem cost-effective to carry out on MTurk, and there is evidence that some Turkers performed the task faithfully. However, we found it to be generally difficult to replicate the document ranking produced by the LDC editor. In retrospect, our effort should have included more rigorous quality control. Figure 5.8 shows that many Turkers had very low edit rates, indicating they were submitting sentences with little or no editing. Conversely, Figure 5.7 shows that several Turkers who have high "edit rates" were likely submitting the provided human references (indicated by low edit rates to the references). Both behaviors are indicative of spammy and careless editors, and this analysis should have been carried out while reviewing submissions for approval, not after concluding the data collection.

It is also important to point out that our editing interface was very simple. It is in stark contrast with the tool used by the LDC editors, which had a number of useful features to aid them in their task. Most critically, editors could see the impact of their edits in real time, as an edit count was displayed for them and updated automatically while they performed the task. This enabled them to truly focus on minimizing the

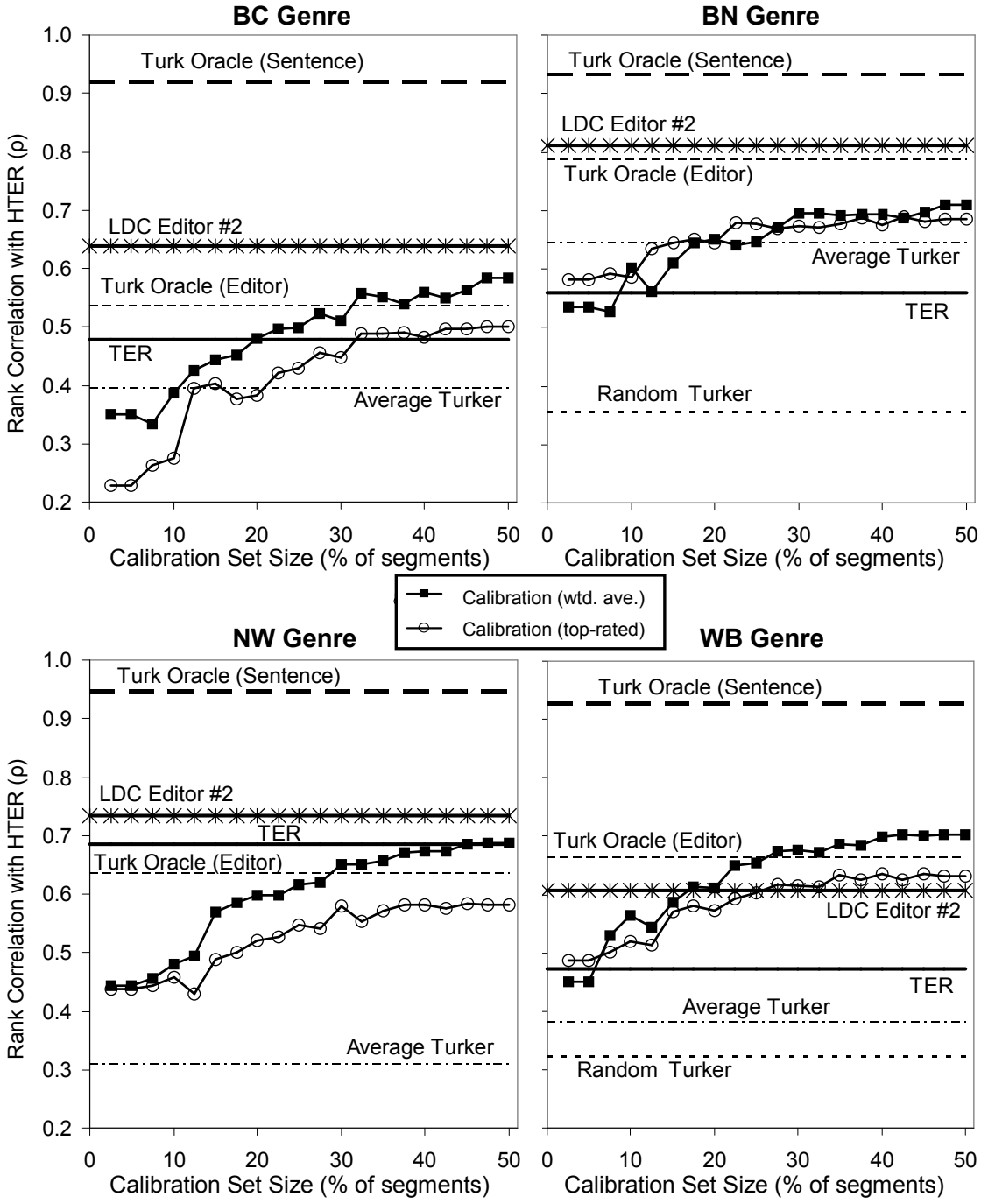


Figure 5.9: Document ranking correlation (Spearman's ρ) for the calibration approach, plotted against varying sizes of the calibration set. (The other methods do not rely on calibration, and are therefore represented by horizontal lines.)

number of edits as instructed, which the Turkers could not do.

5.3 The RYPT Metric

The fluency and adequacy evaluation mentioned in 5.1 involved evaluating a sentence in its entirety. Even though only a single judgment is made, the 1–5 scale allows some granularity. We will now suggest a new metric based on *binary* acceptability judgments, made on partial components of the candidate translation. When the judgments are aggregated, the sentence as a whole receives a continuous score.

The main idea is to reward syntactic constituents in the source sentence that get aligned to "acceptable" translations in the candidate sentence, and penalize constituents that do not. For instance, consider the source-candidate sentence pair of Figure 5.10. To evaluate the candidate translation, we first obtain the parse tree of the source sentence, and match each subtree with the corresponding substring in the candidate string. If the source substring covered by this subtree is translated into an acceptable substring in the candidate, that node gets a `YES` label. Otherwise, the node gets a `NO` label. The metric we propose is taken to be the ratio of `YES` nodes in the parse tree (or RYPT). The candidate in Figure 5.10, for instance, would get a RYPT score of $13/18 = 0.72$.

Note that the source segments to be judged are, by construction of RYPT, segments that are covered exactly by a subtree in the source parse tree. The motivation is that those units are syntactically important, and this probably also makes judging them easier – it is reasonable to assume that strings corresponding to syntactic constituents are easier to process by a human.[8] Making the judgments binary in type also makes the task easier for non-experts, in addition to simplifying quality control.

To justify its use as an evaluation metric, we need to demonstrate that RYPT is a good proxy for human judgment. But it is also important to show that we can obtain the `YES`/`NO` label assignments in an efficient and affordable manner. Case in point,

[8]Due to the nature of the decoder we use, some source segments map to candidate substrings that have gaps in them. We accommodate such queries, as long as the candidate substring has no more than one gap, spanning a maximum of four words.

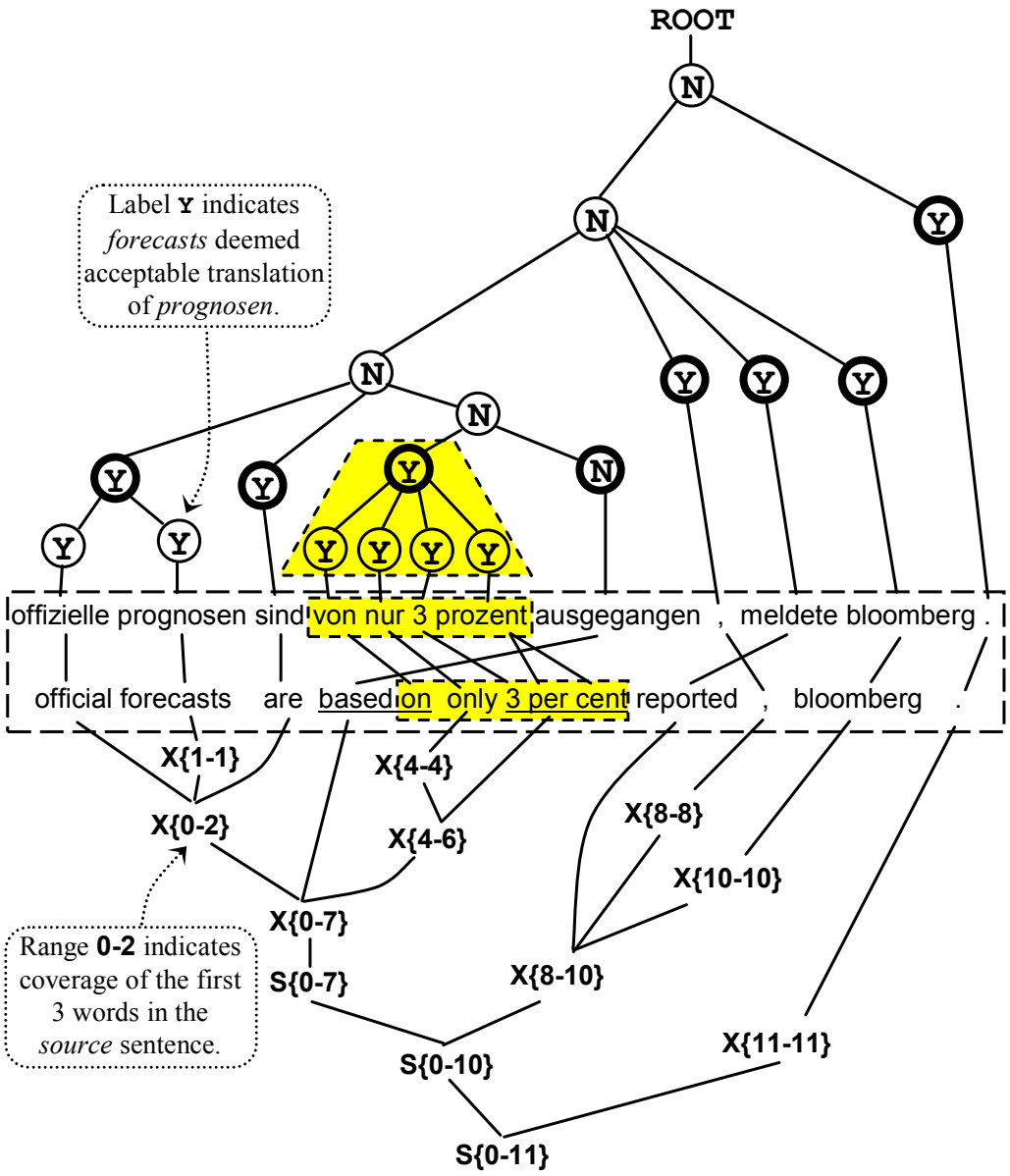


Figure 5.10: A source sentence and its parse tree (top), aligned with a candidate translation and its derivation tree (bottom). Nodes in the parse tree with a thick border correspond to the frontier node set with `maxLen` = 4 (of Algorithm 1). The human annotator only sees the portion surrounded by the dashed rectangle, including the highlighting (though excluding the word alignment links).

scoring the relatively short candidate in Figure 5.10 requires collecting 18 judgments, one per node in the source parse tree. Is it feasible to collect that many judgments, especially for longer sentences?

In this section, we will discuss how to generate queries given a source sentence and a candidate translation, and we will discuss how the number of those queries can be minimized, in order to minimize the amount of collected labels. In the following section, we relate RYPT to the MERT phase of MT training pipelines, and report on experiments that show RYPT to be a good substitute of human judgment.

5.3.1 Obtaining Source-to-Candidate Alignments

The first issue that must be tackled is the source-to-candidate alignments that allows us to identify which segments should be judged. For a given segment in the source sentence, how do we determine which segment of the candidate sentence aligns to it? If a word alignment existed between the source and the candidate, we could take the target substring to contain any word aligned to at least one word in the source segment. To obtain those word alignments, one could run a trained aligner (e.g. GIZA++) on the two sentences, but we propose a different approach here.

In our experiments, we use the Joshua MT system (Li et al., 2009), a hierarchical parsing-based MT system. It can be instructed to produce a translation's derivation tree instead of the surface string itself. Furthermore, each node in the derivation tree is associated with the two indices in the source sentence that indicate the source segment corresponding to this derivation subtree. Those indices are the numbers indicated in curly brackets in Figure 5.10.

Using this information, we can directly recover many of the phrasal alignments between the two sentences, such as the following two alignments in Figure 5.10:

(offizielle,prognosen,sind)—(official,forecasts,are)
(prognosen)—(forecasts)

There are other phrasal alignments that can be deduced from the structure of the tree indirectly, by systematically discarding source words that are part of another

phrasal alignment. For instance, the above two phrasal alignments can be combined to deduce another phrasal alignment:

(offizielle,sind)—(official,are)

This would allow us to extract the phrasal alignments of the sentence pair. Some of the resulting phrasal alignments are already one-to-one mappings, and reduce to word alignment links, such as the alignment (prognosen)—(forecasts). For the many other alignments that are still many-to-many, we convert them to one-to-one mappings as follows. By construction of the MT system, any deduced many-to-many mapping corresponds to a grammar rule that was applied when translating the source sentence. In turn, this means that this particular many-to-many mapping has occurred in the training parallel corpus at least once (otherwise, the mapping would not have appeared in a grammar rule). Therefore, we can recover the individual word alignments by consulting the parallel corpus from which the grammar rules were extracted, and searching for that particular mapping. Once found, we reuse the same word alignments deduced in training, prior to grammar extraction. Figure 5.11 illustrates this example of the proposed deduction method.

Our approach does require maintaining the word alignments obtained prior to rule extraction. That said, we emphasize here that this method of recovering word alignment from phrasal alignment is independent from the hierarchical and parsing-based nature of the Joshua system. The alignment deduction approach we suggest here can be applied to a different MT system as well, as long as that system provides phrasal alignment along with the output. In particular, a phrase-based system such as Moses can be modified in a straightforward manner to provide those phrasal alignments, and then our method could be applied to deduce word alignments.

5.3.2 Crowdsourcing Rypt Judgments

We first start with a pilot study that examines the annotation task in a crowdsourced setup. We consider a German-to-English translation task, and train a Joshua system using the Europarl dataset. We chose 50 source sentences from the WMT '08

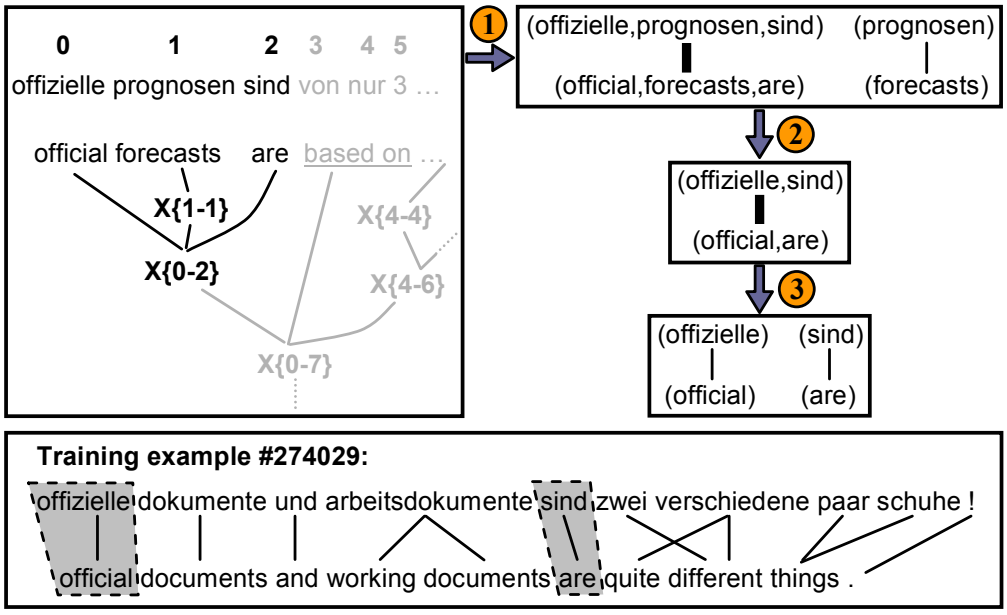


Figure 5.11: An example of resolving a phrasal (i.e. many-to-many) alignment to produce a word (i.e. one-to-one) alignment. Some phrasal alignments are extracted directly from the derivation tree (1), while more phrasal alignments can be deduced by elimination (2). Of those phrasal alignments that are not already one-to-one, word alignments can be found via lookup in the parallel training corpus (3), which had already been word-aligned prior to grammar extraction.

development set, and collected a complete set of judgments for those sentences and their top-300 candidates. We did so by generating a query for each node in the source parse tree, for source segments up to 7 words in length, and each possible translation of that source segment. The parse trees of the German side are obtained with the parser of Dubey (2005).

The queries were uploaded to MTurk, with instructions that read, in part:

> You are shown a "source" German sentence with a highlighted segment, followed by several candidate translations with corresponding highlighted segments. Your task is to decide if each highlighted English segment is an acceptable translation of the highlighted German segment.

Figure 5.12 shows the interface used to the provide `YES`/`NO` labels. In each HIT, the worker is shown up to 10 alternative translations of a highlighted source segment, with each alternative itself highlighted within a full candidate string in which it appears. To aid the worker in the task, they are also shown the reference translation, with a highlighted portion that is believed to correspond to the source segment in question. These reference segments are deduced using word alignments obtained with GIZA++.[9]

We collected five judgments for each of the 5,580 generated queries, for a total of 27,900 judgments. First, collecting multiple judgments allowed us to investigate inter-annotator agreement. In 68.9% of the queries, at least 4 of the 5 annotators chose the same label, signifying a high degree of inter-annotator agreement. The rate of full agreement (i.e. across all five annotators) is 34.1%, also quite high – if labels were provided randomly, the expected full agreement rate is only 6.25%.

Table 5.3 shows the label distribution for the queries, grouped by the word length of the source segment. Unsurprisingly, the longer the segment, the more likely the `NO` label is, since there is more room for the MT system to make an error that renders the translation unacceptable.

[9]These alignments are not always precise, and we do note that fact in the instructions. We also deliberately highlight the reference substring in a different color to make it clear that workers should judge a candidate substring primarily based on the source substring, not only the reference substring.

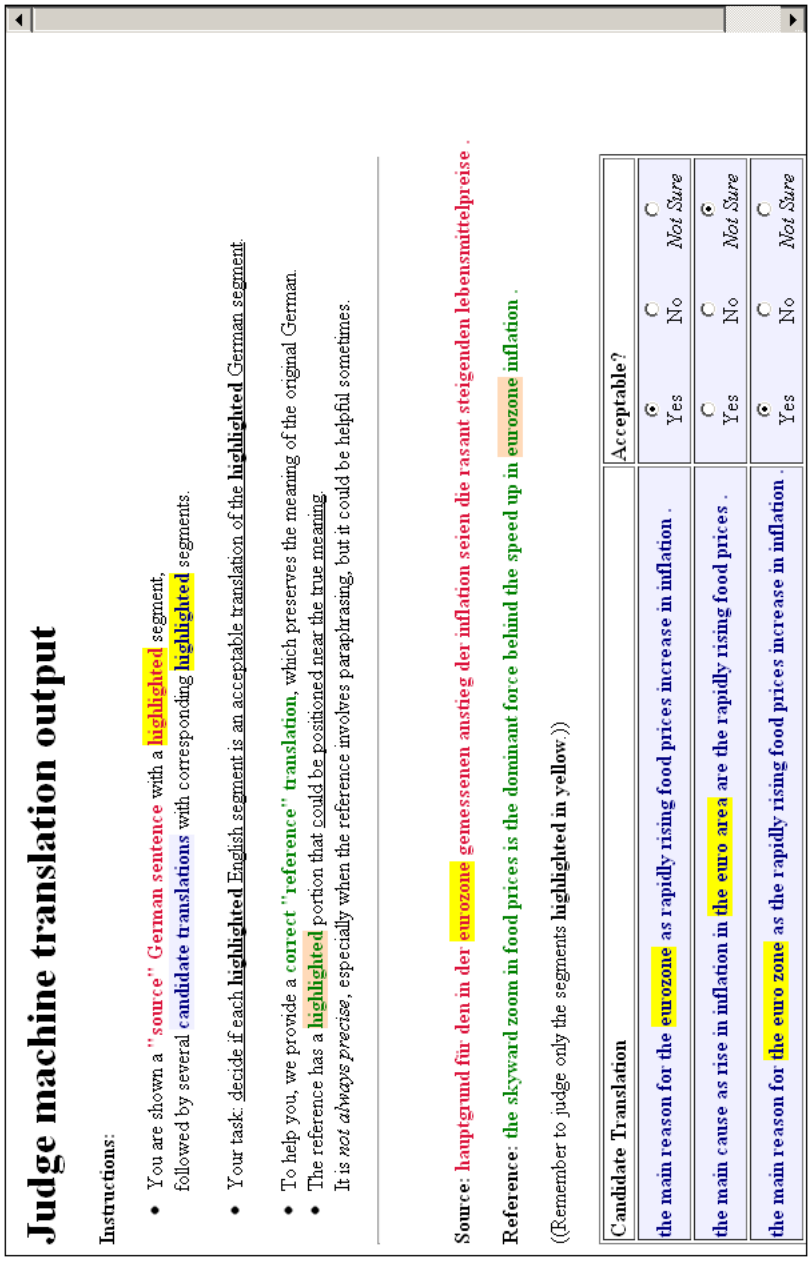


Figure 5.12: The interface for collecting acceptability judgments. Note that it allows us to present an annotator with multiple alternative translations at once.

Source Length (words)	% YES	% NO	% NOT SURE
1	50.5	46.5	2.6
2	40.8	55.8	2.7
3	37.4	59.8	2.4
4	31.3	65.6	2.6
5	32.2	65.0	1.9
6	25.3	71.7	2.2
7	25.9	70.6	2.9
All lengths (1–7)	40.70	56.21	2.56

Table 5.3: The label distribution for collected judgments, for each source substring length. A negligible portion of labels ($< 1\%$) were returned blank.

5.3.3 Label Percolation

Evaluating a candidate translation for the source sentence of Figure 5.10 would require obtaining 18 labels, one for each node in source the parse tree. Instead of querying a human for each one of those nodes, we propose to query only a subset of those nodes, and then **percolate** the obtained labels up and down the parse tree. If a node is labeled NO, this likely means that all its ancestors would also be labeled NO, and if a node is labeled YES, this likely means that all its descendents would also be labeled YES.

What segments of the source sentence should be chosen to be judged? We already indicated that we limit ourselves, by definition of RYPT, to segments that are covered exactly by a subtree in the source parse tree. Beyond that, our query selection strategy should attempt to maximize the amount of YES/NO percolation that would take place. We therefore ensure that for any 2 queries, the corresponding source segments do not overlap: such overlap means one subtree is completely contained within the other, and having both queries might be redundant given the percolation procedure. In short, we suggest selecting source segments so that they fully cover the entire source sentence, but have no overlap amongst them.

In one extreme, each query would correspond to an entire parse tree. This is not ideal since the overwhelming majority of the judgments will most likely be `NO`, which does not help in producing a meaningful score for most translations. In the other extreme, each query would correspond to a subtree rooted at a preterminal. This is also not ideal, since it would place too much emphasis on translations of individual words.

So we need a middle ground. We select a maximum-source-length `maxLen` to indicate how long we are willing to let a source segment be. Starting at the root of the parse tree, we propagate a "frontier" node set down the parse tree, until we end up with a set of nodes that fully cover the source sentence, have no overlap amongst them, and that each cover no more than `maxLen` source words. For instance, with `maxLen` set to 4, the frontier set of Figure 5.10 is the set of nodes with a thick border.

An algorithmic description of the procedure is provided in Algorithm 1. Each query in the resulting `frontierSet` is guaranteed to cover between one and `maxLen` source words, and guaranteed not to overlap with any other query in the set, which would allow us to take full advantage of the downward-`YES` and upward-`NO` percolation.

One question remains: what is a good value for `maxLen`? Here again, choosing too low or too high of a value would yield many queries that would not contribute much to label percolation, either because they are too high up the tree (and have a `NO` label) or too low down the tree (and have a `YES` label). We turn to the collected data to investigate this question as well.

Recall that we collected labels for each query of length up to 7 source words. For each value for `maxLen` up to 7, we ignore all but the queries that would be generated under that particular `maxLen` value, and percolate the labels up and down the tree whenever possible. This would allow us to examine the *applicability* of label percolation, i.e. how often it would actually happen, and the *validity* of label percolation, i.e. how often it is correct.

In Figure 5.13 we plot the coverage before and after percolation (middle two curves), and observe expansion in coverage across different values of `maxLen`, peaking at about +33% for `maxLen`= 4 and 5, with most of the benefit coming from `YES`

Algorithm 1 Constructing the frontier node set for a parse tree.

Require: A source parse tree T rooted at `ROOT`, and a maximum source length `maxLen`.

Ensure: A nonempty set `frontierSet`, containing a subset of the nodes in T.

1. Initialize the set `frontierSet` to the empty set.
2. Initialize the set `currNodes` to {`ROOT`}.
3. **while** `currNodes` is not empty **do**
4. Initialize the set `newNodes` to the empty set.
5. **for** each node N in `currNodes` **do**
6. **if** N covers $\leq$ `maxLen` source words **then**
7. Add N to `frontierSet`.
8. **else**
9. Add all the children of N to `newNodes`.
10. **end if**
11. **end for**
12. Set `currNodes` = `newNodes`
13. **end while**
14. Return `frontierSet`.

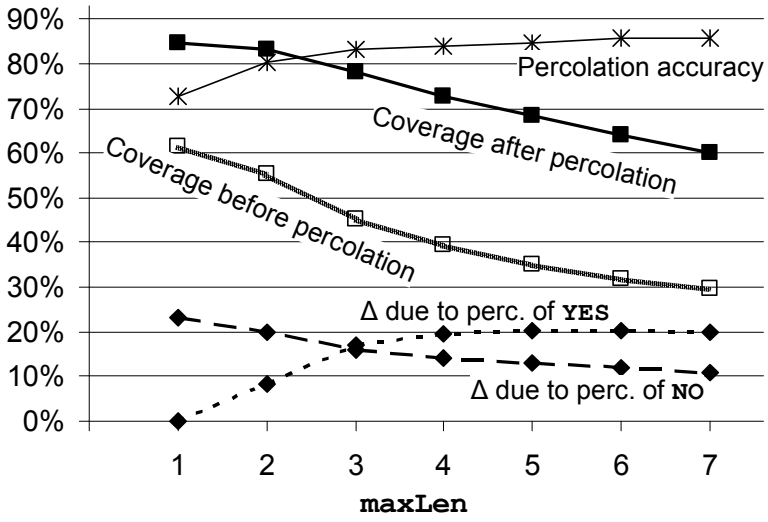


Figure 5.13: Label percolation under different `maxLen` values. The bottom two curves are the breakdown of the difference between the middle two. Accuracy is measured against majority votes.

percolation (bottom two curves).

We also measure the accuracy of labels deduced from percolation (top curve of Figure 5.13). We define a percolated label to be correct if it matches the label given by a majority vote over the original collected labels for that particular node. We find that accuracy at low `maxLen` values is significantly lower than at higher values (e.g. 72.6% vs. 84.1% for 1 vs. 4). This means a middle value such as 3 or 4 is optimal, with higher values also suitable if we wish to place more emphasis on translation fluency.

5.4 Human-Based Parameter Tuning of MT Systems

In the previous section, we presented a metric based on human judgment of translation quality, and introduced it in the context of evaluating MT output. In this section, we take this a step further and propose that RYPT could also be used in

the *tuning* phase of MT systems, which usually requires scoring many thousands of sentences. We illustrate that RYPT could be a viable metric to be used in the MERT phase, by making it a semi-automatic metric that relies on a *database* of human judgment, rather than immediate feedback for each candidate.

5.4.1 Minimum Error Rate Training

Most state-of-the-art MT systems (Och and Ney, 2002; Koehn et al., 2003; Chiang, 2007; Koehn et al., 2007; Li et al., 2009) rely on several models to evaluate the "goodness" of a given candidate translation in the target language. The MT system proceeds by searching for the highest-scoring candidate translation, as scored by the different model components, and returns that candidate as its top-1 candidate. Each of these models need not be a probabilistic model, and instead corresponds to a feature that is a function of a <candidate translation,source sentence> pair.

Treated as a log-linear model, we need to assign a weight for each of the features. Och (2003) shows that setting those weights should take into account the evaluation metric by which the MT system will eventually be judged. This is achieved by choosing the weights that maximize performance on a development set, as measured by that evaluation metric. This tuning step is known as the MERT phase (for **M**inimum **E**rror **R**ate **T**raining) in training pipelines of MT systems. The other insight of Och's work is that there exists an efficient search method to find such weights (see 5.4.2).

The MERT phase requires the scoring of thousands of candidate translations, as it relies on generating and scoring an n-best candidate list for each sentence in the tuning dataset. Automated evaluation, particularly using efficient metrics such as BLEU and TER, is necessary in MERT, since performing manual evaluation of thousands of sentences is not feasible.

In theory, one could imagine trying to optimize a metric like HTER during the MERT phase, but that would require the availability of an HTER automatic scorer, which, by definition, does not exist. If done manually, the scoring of thousands of candidates produced during MERT would literally take weeks, and cost a large sum of money.

As daunting as such a task seems for any human-based metric, we describe how RYPT could be a good candidate for MERT, by making certain approximations that take advantage of the redundancy in the n-best candidate lists produced during MERT. In this section, we describe how this redundancy can be used to our advantage to minimize the need to involve a human annotator. We furthermore show that RYPT is a better predictor of translation quality than BLEU, making it an excellent candidate for MERT tuning.

5.4.2 Och's Line Search Method

A common approach to translating a source sentence f in a foreign language is to select the candidate translation e that maximizes the posterior probability:

$$Pr(e \mid f) \stackrel{\text{def}}{=} \frac{\exp(s_\Lambda(e, f))}{\sum_{e'} \exp(s_\Lambda(e', f))}.$$

This defines $Pr(e \mid f)$ using a *log-linear model* that associates a sentence pair (e, f) with a feature vector $\Phi(e, f) = \{\phi_1(e, f), ..., \phi_M(e, f)\}$, and assigns a score

$$s_\Lambda(e, f) \stackrel{\text{def}}{=} \Lambda \cdot \Phi(e, f) = \sum_{m=1}^{M} \lambda_m \phi_m(e, f)$$

for that sentence pair, with the feature weights $\Lambda = \{\lambda_1, ..., \lambda_M\}$ being the parameters of the MT system. Therefore, the system selects the translation $\hat{e}$:

$$\hat{e} = \underset{e}{\operatorname{argmax}}\, Pr(e \mid f) = \underset{e}{\operatorname{argmax}}\, s_\Lambda(e, f). \tag{5.1}$$

Och (2003) provides evidence that Λ should be chosen by optimizing an objective function based on the evaluation metric of interest, rather than likelihood. Since the error surface is not smooth, and a grid search is too expensive, Och suggests an alternative, efficient, line optimization approach.

Assume we are performing a line optimization along the d^{th} dimension. Consider a foreign sentence f, and let the candidate set for f be $\{e_1, ..., e_K\}$. Recall from Equation 5.1 that the 1-best candidate at a given Λ is the one with maximum $\sum_{m=1}^{M} \lambda_m \phi_m(e_k, f)$. We can rewrite this sum as $\lambda_d \phi_d(e_k, f) + \sum_{m \neq d} \lambda_m \phi_m(e_k, f)$. The

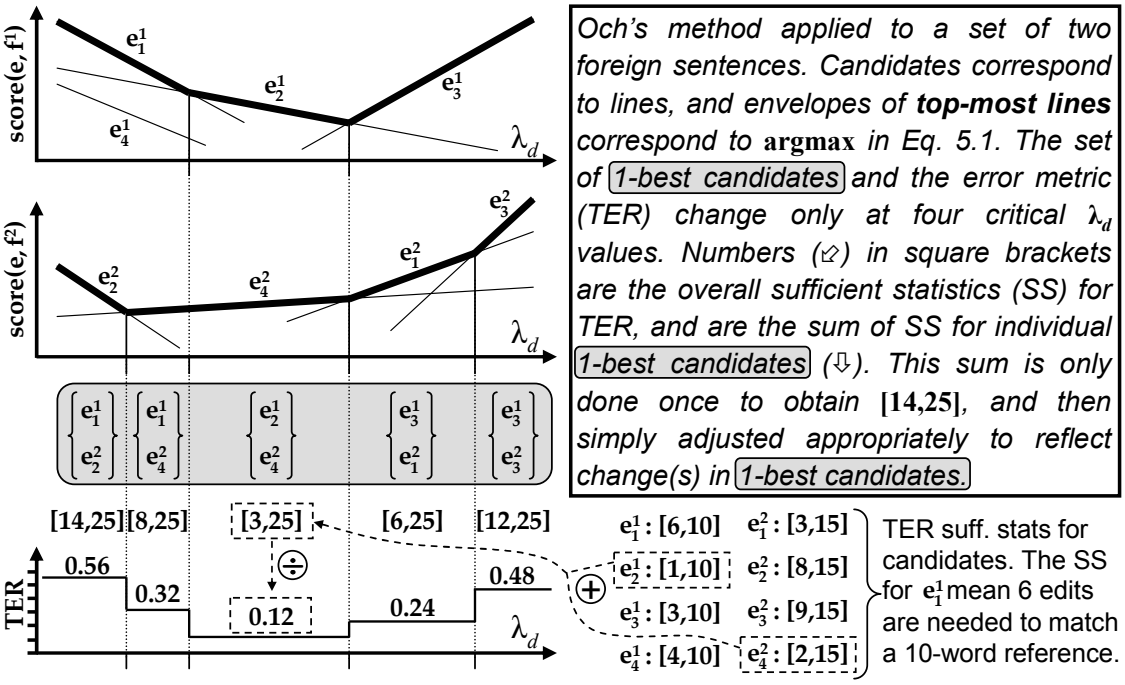


Figure 5.14: Och's method applied to a set of two foreign sentences. This figure is essentially a visualization of Equation 5.1. For clarity, we show here sufficient statistics for TER, since there are only 2 of them, but the metric optimized in MERT is usually BLEU.

second term is constant with respect to λ_d (the parameter being optimized), and so is $\phi_d(e_k, f)$. Renaming those two quantities $\texttt{offset}_\Lambda(e_k)$ and $\texttt{slope}(e_k)$, we get

$$s_\Lambda(e_k, f) = \texttt{slope}(e_k)\lambda_d + \texttt{offset}_\Lambda(e_k),$$

which is a linear equation in λ_d. Therefore, if we plot the score for a candidate translation vs. λ_d, that candidate will be represented by a line. If we plot the lines for *all* candidates (as in Figure 5.14), then the upper envelope of those lines would indicate the best candidate at any value for λ_d.

Therefore, the objective function is piece-wise linear across any of the M dimensions[10], meaning we only need to evaluate the objective at the critical points corresponding to line intersection points. Furthermore, we only need to calculate the

[10]Or, in fact, along any linear combination of the M dimensions.

sufficient statistics once, at the smallest critical point, and then simply adjust the sufficient statistics to reflect changes in the set of 1-best candidates.

This overview of Och's search method makes it clear why BLEU is such a suitable metric to be used in MERT, besides being the most widely reported metric in MT research. BLEU is particularly suitable for MERT because it can be computed quite efficiently, and its sufficient statistics are decomposable, as required by MERT.[11,12]

5.4.3 Building a Human Judgment Database

To be used in a scoring-intensive application like MERT, we need to minimize, as much as possible, the number of judgments that need to be collected to compute RYPT scores. The label percolation strategy of 5.3.3 certainly aids in that regard. We propose another approximation here that can also drastically minimize the number of human judgments.

The approximation takes advantage of the high level of similarity across the candidate translations of a single source sentence, and is based in taking judgments made on one candidate and reusing them when scoring a different candidate. Specifically, once we have a judgment for a source-candidate substring pair, that same judgment can be used across all candidates for this source sentence.

In other words, we build a database for each source sentence, where each entry is of the form:

$$< \text{source substring} , \text{source location} , \text{target substring} , \texttt{judgment} >$$

For example, the highlighted portion in Figure 5.10, along with the provided `YES` judgment, would result in the following database entry:

$$< \text{“von nur 3 prozent”} , [3, 6] , \text{“on only 3 per cent”} , \texttt{YES} >$$

[11] Note that for the sufficient statistics to be decomposable, the metric itself need not be – this is in fact the case with BLEU.

[12] Strictly speaking, the sufficient statistics need not be decomposable in MERT, as they can be recalculated at each critical point. However, this would slow down the optimization process considerably, since one cannot traverse the dimension by simply adjusting the sufficient statistics to reflect changes in 1-best candidates.

Note that the entries do not store the full candidate string, since we reuse a judgment across all the candidates of that source sentence. For instance, say we observe the following sentence pair:

der patient wurde isoliert .
the patient was isolated .

and we collect the following judgment:

< "der patient" , [0, 1] , "the patient" , `YES` >

where the [0, 1] represents the location of the source segment "der patient" within the source sentence. Then this judgment would also apply to any *other* candidate translation of this same source sentence, and all of the following substrings are **automatically** labeled `YES` as well:

the patient isolated .
the patient was in isolation .
the patient has been isolated .

Similarly, if we collect the judgment:

< "der patient" , [0, 1] , "of the patient" , `NO` >

from the sentence pair:

der patient wurde isoliert .
of the patient was isolated .

then this would also apply to any *other* candidate translation of the source sentence, and the following substrings are **automatically** labeled `NO` as well:

of the patient isolated .
of the patient was in isolation .
of the patient has been isolated .

5.4.4 Evaluating RYPT as an MT Metric

The two label approximating strategies (percolating labels up and down the tree, and using judgments across candidates) result in what would be an approximation of a 'true' RYPT score, but employing them considerably reduces the amount of data we need to collect. We are now ready to evaluate the metric's usage in a MERT-like setting, and examine how it performs as a proxy for human judgment.

From the WMT '08 German-English development set, we chose a subset of 250 source sentences, and generated queries based on the set of candidate sentences produced in the last iteration of a MERT run optimizing BLEU on the full 2051-sentence development set. We limit our queries to source segments corresponding to frontier nodes with `maxLen` = 4, as extracted by Algorithm 1. We obtain a total of 3,601 subtrees across the 250 sentences, for an average of 14.4 per sentence. On average, each of those subtrees has 3.65 alternative translations. In other words, we collected only $14.4 \times 3.65 = 52.6$ labels per source sentence to evaluate its 300 candidate translations.

Only about 4.8% of the judgments were returned as `NOT SURE` (or, occasionally, blank), with the rest split into 35.1% `YES` judgments and 60.1% `NO` judgments. The coverage we get before percolating labels up and down the trees is 39.4% of the nodes, increasing to a coverage of 72.9% after percolation. This is quite good, considering we only perform a single data collection pass, and considering that about 10% of the subtrees do not align to candidate substrings to begin with (e.g. single source words that lack a word alignment into the candidate string).

One critical question remains regarding the RYPT metric, which is whether or not the collected labels allow us to calculate a RYPT score that is reliably correlated with human judgment. We conducted an evaluation experiment to test RYPT's predictive power of human judgment, and compared it with BLEU's. Given the candidate set of a source sentence, we rerank the set according to RYPT and extract the top-1 candidate, and we rerank the candidate set according to BLEU, and extract the top-1 candidate. We refer to those two candidates as the RYPT- and BLEU-selected candidate, respectively. We then present the two candidates to human judges, and ask them to choose the one that is a more adequate translation. In the first experiment

	References shown; unrestricted		References **not** shown; restricted to DE workers	
Preferred candidate	# judgments	% judgments	# judgments	% judgments
Top-1 by RYPT	346	46.1	113	45.2
Top-1 by BLEU	270	36.0	73	29.2
Neither	134	17.9	64	25.6
Total	750	100.0	250	100.0

Table 5.4: Results of the two RYPT vs. BLEU comparison experiments. The left half corresponds to the experiment (open to all workers) where English references were shown, whereas the right half corresponds to the experiment (open only to workers living in Germany) where English references were *not* shown.

we describe, three judgments are collected per sentence pair comparison.

The results show that RYPT significantly outperforms BLEU when it comes to predicting human preference, with its choice prevailing in 46.1% of judgments vs. 36.0% for BLEU, with 17.9% judged to be of equal quality (left half of Table 5.4). This advantage is especially true when the judgments are grouped by sentence, and we examine cases of strong agreement among the three annotators (Table 5.5): whereas BLEU's candidate is strongly preferred in 32 of the candidate pairs (bottom 2 rows), RYPT's candidate is strongly preferred in about double that number: 60 candidate pairs (top 2 rows).

This is quite a remarkable result, given that BLEU, by definition, selects a candidate that has significant overlap with the reference, which is the same reference shown to the annotators in this RYPT vs. BLEU experiment. This means that BLEU has an inherent advantage, especially in comparisons where both candidates are more or less of equal quality, since annotators are encouraged (in the instructions) to make a choice even if the two candidates seem of be of equal quality at first glance. We hypothesize that an annotator, pressed to make such a choice, is likely to select the candidate that superficially 'looks' more like the reference, and declare it to be the 'better' of the two candidates. That candidate will most likely be the BLEU-selected one.

Aggregate	# sentences	% sentences	Aggregate	# sentences	% sentences
RYPT +3	45	18.0			
RYPT +2	15	6.0	RYPT +any	120	48.0
RYPT +1	60	24.0			
± 0	42	16.8	± 0	42	16.8
BLEU +1	55	22.0			
BLEU +2	5	2.0	BLEU +any	88	35.2
BLEU +3	28	11.2			
Total	250	100.0	Total	250	100.0

Table 5.5: Results of the RYPT vs. BLEU comparison experiment, grouped by sentence. This table corresponds to the left half of Table 5.4. The "aggregate" for a comparison is calculated from the three judgments collected for that comparison. For instance, an aggregate of "RYPT +3" means all three judges favored RYPT's choice, and "RYPT +1" means one more judge favored RYPT than did BLEU.

To test this hypothesis, we repeated the experiment but *without showing annotators the reference translations*. That is, annotators were supposed to make a judgment based only on the source sentence. Since this version of the task requires knowledge of German, we limited data collection to workers living in Germany. Only one judgment per source sentence was collected, since there are presumably significantly fewer Turkers based in Germany than worldwide Turkers who speak English.

The results of this second experiment support our hypothesis, and RYPT's advantage is even more pronounced: human judges prefer the RYPT-selected candidate 45.2% of the time, while BLEU's candidate is preferred only 29.2% of the time, with 25.6% judged to be of equal quality (right half of Table 5.4). Our hypothesis is further supported by the fact that most of the gain of the "equal-quality" category comes from BLEU, which loses 6.8 percentage points over the first experiment. On the other hand, RYPT's share remains largely intact, losing less than a single percentage point.

5.5 Related Work

As part of their annual shared task, the WMT workshop conducts manual evaluation of submitted MT systems by distributing the work across tens of volunteers. The main evaluation setup is one where the annotator ranks the outputs of five systems on a particular source sentence, from best to worst. Those rankings are aggregated to produce an overall ranking of the systems. In the beginning, WMT relied on a self-designed online portal, but moved to MTurk starting in 2010 (Callison-Burch et al., 2010), mainly to take advantage of MTurk's infrastructure. WMT did also explore paying Turkers to perform the task, and observed that the amount of collected labels could easily be doubled or tripled, allowing their results to achieve higher statistical significance.

Callison-Burch (2009) proposed evaluating machine translation quality on MTurk using a number of different evaluation setups, including HTER as computed from crowdsourced edits. The task was to predict the ranking of five MT *systems*, rather than 40-53 *documents* as we do here. Our rank prediction task is also more difficult, since the ranked systems in that work differed considerably in output quality from each other (the system set was limited to top-performing and bottom-performing systems).

The question of how to design an automatic metric that best approximates human judgment has received a lot of attention. NIST started organizing the Metrics for Machine Translation Challenge (MetricsMATR) in 2008, with the aim of developing automatic evaluation metrics that correlate highly with human judgment of translation quality. The WMT workshop also conducts an assessment of how well submitted automatic metrics correlate with human judgment.

Nießen et al. (2000) is an early work that constructs a database of translations and judgments. There, a source sentence is stored along with all the translations that have already been manually judged, along with their scores. They use this database to carry out "semi-automatic" evaluation in a fast and convenient fashion thanks to a tool they developed with a user-friendly GUI. Our particular setup, whereby individual components of the candidate translation are evaluated based on

correspondence with the source parse tree, was first featured as an evaluation setup in WMT, under the name constituent-based evaluation (Callison-Burch et al., 2007, 2008).

5.6 Conclusion

Being able to perform manual evaluation of MT system output in an efficient and consistent manner remains a challenge to the research community, due to its high expense, the required duration of time, and the variance in annotators' judgments. In this Chapter, we showed that crowdsourcing can certainly help alleviate the cost factor (time and money), and investigated the effectiveness of crowdsourcing two tasks: computing HTER scores to reproduce the document ranking from a professional editor, and evaluating output using RYPT, a novel metric we designed that is based on constituent-level acceptability judgments.

Our attempt to accurately predict HTER-based ranking of documents produced mixed results. Upon further investigation, we should have exercised more rigorous quality control during data collection, in order to quickly identify spammy editors and reject their work. On the other hand, our new RYPT metric is easier to collect judgments for, and is much more suited for a crowdsourced setting. Our experiments showed that it was feasible to collect enough judgments to compute RYPT scores by harvesting the power of the crowd. We furthermore demonstrated that RYPT is better than BLEU in approximating human judgment of translation quality. Those results pave the way for a manual-based metric that can be used to evaluate translations semi-automatically, and even tune system parameters in the MERT phase.

In the next Chapter, we investigate another annotation task that benefits MT systems, as we crowdsource the **translation** task to non-professional translators. Unlike our effort for the editing task of this Chapter, we will employ more comprehensive, rigorous, and sophisticated quality control strategies to distinguish good translations from bad ones. As a result, we will be able to obtain translations that are at such a high level of quality, that they fall within the range of professionally-produced trans-

lations. The contrast between the results of the editing task and the translation task will highlight the importance of quality control for crowdsourced data collection, which is a major theme of this thesis.

Chapter 6

Crowdsourcing Translation

In the context of training systems for machine translation (MT), a parallel corpus is indispensable, as it lies at the very core of the statistical approach to learning. As a result, the quality of a system's output is dependent on the size of the training dataset, as well as the dataset's relevance, coverage, and the quality of the translations contained within.

A parallel dataset is necessary not only at training time, but also when *evaluating* MT systems. The evaluation might be based on scoring outputs with automatic metrics such as BLEU and TER, or it might be a manual evaluation with a human component. Either way, a set of reference translations is needed, against which MT output is compared, in order to obtain an objective, quantitative measure of the MT output quality.

Given the high cost typically associated with creating such a parallel dataset, crowdsourcing might seem like an effective solution to collect translations instead. Unfortunately, naively collecting translations by crowdsourcing the task to non-professional translators yields disfluent, low-quality results if no quality control is exercised.

Creating translations is not an easy task. It represents the most complex task of this thesis, as it involves translating entire sentences, into a language other than most Turkers' mother language no less. In this Chapter, we demonstrate a variety of mechanisms that increase the translation quality to near professional levels.[1]

[1]Most of this Chapter is based on Zaidan and Callison-Burch (2011b). Section 6.5 is based on

Specifically, we solicit redundant translations and edits to them, and automatically select the best output among them. We propose a set of features that model both the translations and the translators, and use these features to score the collected translations, enabling us to discriminate between acceptable and unacceptable translations. We recreate the NIST 2009 Urdu-to-English evaluation set with Mechanical Turk, and quantitatively show that our models are able to select translations within the range of quality that we expect from professional translators.

The methodology is also extended to collect a large amount of English translations of dialectal Arabic sentences. The amount of data is large enough to train a machine translation system, one which dramatically outperforms a system trained on 100 times more MSA-only data.

6.1 Creating a Parallel Dataset

In natural language processing research, translations are most often used in statistical machine translation (SMT), where systems are trained using bilingual sentence-aligned parallel corpora. The evaluation of MT systems also relies on the existence of a parallel evaluation set, especially if the evaluation uses one or more automatic metrics.

Some sources of parallel data exist independently of MT research efforts, such as the Canadian Hansards (since Canadian Parliamentary proceedings must be published in both French and English), legislative texts of the European Parliament in the EU's 23 official languages, and UN documents in its six official languages.[2] Other sources, not as strictly "parallel", include book translations and news items across different languages. In the same vein, there are various options for creating new training resources for new language pairs, such as harvesting the web for translations or comparable corpora (Resnik and Smith, 2003; Munteanu and Marcu, 2005; Smith et al.,

Zbib et al. (2012), on which I am a co-author. The experiments were carried out by the co-authors affiliated with Raytheon BBN, and my contribution was helping BBN establish the pipeline and methodology for collecting crowdsourced translations.

[2]Or more: the Universal Declaration of Human Rights has been translated into more than 350 languages.

2010), or designing models that are capable of learning translations from monolingual corpora (Rapp, 1995; Schafer and Yarowsky, 2002; Haghighi et al., 2008).

SMT can be applied to any language pair for which there is sufficient data, and it has been shown to produce state-of-the-art results for language pairs like Arabic–English, where there is ample data. However, large bilingual parallel corpora exist for relatively few languages pairs. In theory, a parallel dataset could be created from scratch, by selecting some text in the source language, and commissioning it to be translated into the target language by a translation agency. But relatively little consideration is given to this idea, as it presents two issues, one of **cost**, and one of **feasibility**.

The **cost** associated with producing professional translations could be prohibitively high. Germann (2001) estimated the cost of hiring professional translators to create a Tamil–English corpus at $0.36/word. At that rate, translating enough data to build even a small parallel corpus like the LDC's 1.5 million word Urdu–English corpus would exceed half a million dollars. The turnaround time is also quite long, as the process might take months to complete.

The second issue, that of **feasibility**, is particularly relevant when one of the two languages is a less common language, because that severely limits the number of translation agencies with staff that is even capable of dealing with the language pair of interest – it is not necessarily easy to find a translation agency that employs staff fluent in, say, Thai. Languages such as Thai have been termed by the LDC as "less commonly taught languages", a group that includes languages as diverse as Urdu, Hungarian, and Yoruba (a language spoken in West Africa). While we have little to no parallel resources for these languages, the languages themselves are by no means "rare", as each has speakers numbering in the tens of millions (Table 6.1). Therefore, the existence of such resources would have a big impact.

6.1.1 Translation by Non-Professionals

In this Chapter, we explore the idea of creating low cost translations via crowdsourcing. We use MTurk to find a large group of non-professional translators, and

Language	Speakers (millions)
Bengali	207
Hungarian	15
Punjabi	57
Tamil	66
Thai	46
Urdu	60
Yoruba	20

Table 6.1: LDC's less commonly taught languages and their speaker counts (Encarta, 2007).

have them recreate an Urdu–English evaluation set at a fraction of the cost of professional translators. More critically, the low entry barrier puts foreign markets on the table as an option.

That said, soliciting translations from anonymous non-professionals carries a significant risk of poor translation quality, whereas hiring a professional translator ensures a degree of quality and care. Figure 6.1 shows how one Urdu headline was rendered disfluently by a Turker as *Barak* ***Obam*** *will* ***do a new policy*** *with Iran.* Another injected some sarcasm in their translation: *Barak Obama and America* ***weave new evil strategies*** *against Iran.* In general, the translations are done conscientiously, but many translations reflect non-native English.

Despite the frequency of low-quality, disfluent translations, we show that it is possible to obtain high-quality translations in aggregate by soliciting multiple translations, redundantly editing them, and then selecting the best of the bunch. Most existing quality control mechanisms for crowdsourcing employ some form of voting, assuming a discrete set of possible labels. This is not the case for translations, where the 'labels' are full sentences. The complexity arises mainly because the output is structured, and so the space of possible annotations is immense, and the space of possible outputs is diverse and complex.

We therefore need a different approach for quality control. To select the best translation, we use a machine learning-inspired approach that assigns a score to each

Src:	باراک اوبامہ امریکہ ایران کے ساتھ نئی حکمت عملی اپنائے گا
Ref1:	Barack Obama: America Will Adopt New Strategy with Iran
Ref2:	America to Adopt New Strategy for Iran: Barack Obama
Ref3:	Barack Obama: America Will Adopt a New Iran Strategy
Turk1:	Barak Obam will do a new policy with Iran.
Turk2:	Barack Obama: America will use a new policy towards Iran.
Turk3:	Barak Obama and America weave new evil strategies against Iran.

Figure 6.1: Several translations for an Urdu headline, produced by professional and non-professional translators. While we purposely chose bad translations for illustration purposes, such translations are not atypical. Our goal is to discriminate bad translations from good ones.

translation we collect. The scores discriminate acceptable translations from those that are not (and competent translators from those who are not). The scoring is based on a set of informative, intuitive, and easy-to-compute features.

First, we discuss reproducing the Urdu-to-English 2009 NIST evaluation set, and describe a principled approach to discriminate good translations from bad ones, given a set of redundant translations for the same source sentence. The original dataset already has professionally-produced reference translations, which allows us to objectively and quantitatively compare the quality of professional and non-professional translations, and the effectiveness of our selection methods. Later, we discuss crowdsourcing a dialectal Arabic translation task, in which we collect enough translations to train MT systems specialized to dialectal Arabic. Those systems dramatically outperform a system trained on MSA-only data, despite using much less training data.

6.2 Data Collection

We translated the Urdu side of the Urdu–English test set of the 2009 NIST MT Evaluation Workshop.[3] The set consists of 1,792 Urdu sentences from a variety of news and online sources. The set includes four different reference translations for each source sentence, produced by professional translation agencies. NIST contracted the LDC to oversee the translation process and perform quality control.

This particular dataset, with its multiple reference translations, is very useful because we can measure the quality range for professional translators, and it provides a measure of annotator agreement for the translation task. This in turn gives us an idea of whether or not the crowdsourced translations approach the quality of a professional translator, and also gives a realistic goal to aim for. Furthermore, the reference translations will help us actually evaluate the different filtering methods.

6.2.1 Translation HIT Design

We solicited English translations for the sentences in the Urdu side of the NIST dataset. Our HIT involved showing the worker a sequence of Urdu sentences, and asking them to provide an English translation for each one. A brief set of guidelines was provided, instructing the Turkers to make sure that their English translation:

- Does not add or delete any information from the original text
- Has the same meaning and style as the original
- Does not contain any spelling errors
- Is grammatical, natural-sounding English

The screen also included a short questionnaire section about the Turkers' language abilities. Figure 6.2 shows the interface, populated with Urdu source sentences. The reward was set at $1.00 per screen, or roughly $0.005 per word.

In our first collection effort, we solicited only one translation per Urdu sentence. After confirming that the task is feasible due to the large pool of workers willing

[3] http://www.itl.nist.gov/iad/894.01/tests/mt/2009/

Figure 6.2: The interface for the translation task, populated with Urdu source sentences.

and able to provide translations, we carried out a second collection effort, this time soliciting *three* translations per Urdu sentence (from three distinct translators). The interface was also slightly modified, in the following ways:

- Instead of asking Turkers to translate a full document (as in our first pass), we instead split the data set into groups of 10 sentences per HIT.
- We converted the Urdu sentences into images so that Turkers could not cheat by copying-and-pasting the Urdu text into an MT system.
- We collected information about each worker's geographic location, using a JavaScript plugin.

The translations from the first pass were of noticeably low quality, most likely due to Turkers using automatic translation systems. That is why we used images instead of text in our second pass, which yielded significant improvements. That said, we do not discard the translations from the first pass, and we do include them in our experiments.

6.2.2 Post-editing and Ranking HITs

In addition to collecting four translations per source sentence, we also collected **post-edited** versions of the translations, as well as **ranking judgments** about their quality.

Figure 6.3 gives examples of the unedited translations that we collected in the translation pass. These typically contain many simple mistakes like misspellings, typos, and awkward word choice. We posted another MTurk task where we asked workers to *edit the translations* into more fluent and grammatical sentences. We restrict the task to US-based workers to increase the likelihood that they would be native speakers of English. We used the same editing interface from the HTER task (of 5.2.1), though minimizing the number of edits was not of much importance in this case.

Avoiding dieting to prevent from flu	abstention from dieting in order to avoid Flu	Abstain from decrease eating in order to escape from flue	In order to be safer from flu quit dieting
This research of American scientists came in front after experimenting on mice.	This research from the American Scientists have come up after the experiments on rats.	This research of American scientists was shown after many experiments on mouses.	According to the American Scientist this research has come out after much experimentations on rats.
Experiments proved that mice on a lower calorie diet had comparatively less ability to fight the flu virus.	in has been proven from experiments that rats put on diet with less calories had less ability to resist the Flu virus.	It was proved by experiments the low calories eaters mouses had low defending power for flue in ratio.	Experimentaions have proved that those rats on less calories diet have developed a tendency of not overcoming the flu virus.
research has proven this old myth wrong that its better to fast during fever.	Research disproved the old axiom that " It is better to fast during fever"	The research proved this old talk that decrease eating is useful in fever.	This Research has proved the very old saying wrong that it is good to starve while in fever.

Figure 6.3: We redundantly translate each source sentence by soliciting multiple translations from different Turkers. These translations are put through a subsequent editing set, where multiple edited versions are produced. We select the best translation from the set using features that predict the quality of each translation and each translator.

We also asked US-based Turkers to *rank the translations*. We presented the translations in groups of four, and the annotator's task was to rank the sentences by fluency, from best to worst (allowing ties). When ranking the sentences, Turkers were asked to consider the following factors:

- Is the English reasonably good?
- Does the sentence make sense?
- Do the grammar and spelling require only minimal correction?
- Are proper nouns correctly capitalized (e.g. Obama, UN, USA, Pakistan)?

We collected redundant annotations in these two tasks as well. Each translation is edited three times (by three distinct editors). We solicited only one edit per translation from our first pass translation effort, since it was fairly obvious that effort resulted in a large number of poor translations. So, in total, we had 10 post-edited translations for each source sentence (plus the four original translations). In the ranking task, we collected judgments from five distinct workers for each translation group (i.e. each translation has five rank labels).

6.2.3 Data Collection Cost

We paid a reward of $0.10 to translate a sentence, $0.25 to edit a set of ten sentences, and $0.06 to rank a set of four translation groups. Therefore, we had the following costs:

- Translation cost: $716.80
- Editing cost: $447.50
- Ranking cost: $134.40

(If not done redundantly, those values would be $179.20, $44.75, and $26.88, respectively.)

In total, we managed to collect 7,000+ translations, 17,000+ edited translations, and 35,000+ rank labels.[4] We also use about 10% of the existing professional references in most of our experiments (see 6.3.2 and 6.3.3). If we estimate the cost at $0.30/word, that would roughly be an additional $1,000. Adding Amazon's 10% fee, this brings the grand total to slightly under $2,500.

52 different Turkers took part in the translation task, each translating 138 sentences on average. In the editing task, 320 Turkers participated, averaging 56 sentences each. In the ranking task, 245 Turkers participated, averaging 9.1 HITs each, or 146 rank labels (since each ranking HIT involved judging 16 translations, in groups of four).

6.3 A Selection Model for Quality Control

Our approach to building a translation set from the available data is to select, for each Urdu sentence, the one translation that our model believes to be the best out of the available translations. We examine various techniques for identifying that

[4]Data URL: http://www.cs.jhu.edu/~ozaidan/RCLMT.

best translation, and evaluate these selection techniques by comparing the selected Turker translations against existing professionally-produced translations. The more the selected translations resemble the professional translations, the higher the quality.

In this Section, we give the technical details of our selection strategy, by describing our scoring model and our feature set, and describing our evaluation strategy, and what makes a translation 'resemble' a reference translation.

6.3.1 Model Features

Our model selects one of the 14 English options generated by Turkers. For a source sentence s_i, our model assigns a score to each sentence in the set of available translations $\{t_{i,1}, ...t_{i,14}\}$. The chosen translation is the highest scoring translation:

$$tr(s_i) = tr_{i,j^*} \text{ s.t. } j^* = \underset{j}{\operatorname{argmax}}\, score(t_{i,j}) \tag{6.1}$$

where $score(.)$ is the dot product:

$$score(t_{i,j}) \stackrel{\text{def}}{=} \vec{w} \cdot \vec{f}(t_{i,j}) \tag{6.2}$$

Here, $\vec{w}$ is the model's weight vector (tuned as described below in 6.3.2), and $\vec{f}$ is a translation's corresponding feature vector. Each feature is a function computed from the English sentence string, the Urdu sentence string, the workers (translators, editors, and rankers), and/or the rank labels. We use 21 features, categorized into the following three sets.

Sentence-level (6 features). In essence, we can assume that native speakers are all proficient at understanding a source sentence. The difficulty with the translation task is that it may not be easy to produce good and natural-sounding English sentences, particularly because most of the Turkers performing our task were native Urdu speakers whose second language was English. Therefore, the first set of features attempt to discriminate good English sentences from bad ones.

- Language model features: each sentence is assigned a log probability and per-word perplexity score, using a 5-gram language model trained on the English Gigaword corpus.
- Sentence length features: a good translation tends to be comparable in length to the source sentence, whereas an overly short or long translation is probably bad. We add two features that are the ratios of the two lengths (one to penalize short sentences and one to penalize long ones).
- Web n-gram match percentage: we assign a score to each sentence based on the percentage of the n-grams (up to length 5) in the translation that exist in the Google N-Gram Database.
- Web n-gram geometric average: we calculate the average over the different n-gram match percentages (similar to the way BLEU is computed; see 5.1.1.1). We add three features corresponding to max n-gram lengths of 3, 4, and 5.
- Edit rate to other translations: a bad translation is not likely to be very similar to other translations, since there are many more ways a translation can be bad than for it to be good. So, we compute the average edit rate distance from the other translations (as defined by the TER metric; see 5.1.1.2).

Worker-level (12 features). We add *worker*-level features that evaluate a translation based on *who* provided it.

- Aggregate features: for each sentence-level feature above, we have a corresponding feature computed over *all* that worker's translations.
- Language abilities: we ask workers to provide information about their language abilities. We have a binary feature indicating whether Urdu is their native language, and a feature for how long they have spoken it. We add a pair of equivalent features for English.

- Worker location: two binary features reflect a worker's location, one to indicate if they are located in Pakistan, and one to indicate if they are located in India.[5]

Ranking (3 features). The third set of features is based on the ranking labels we collected (see 6.2.2).

- Average rank: the average of the five rank labels provided for this translation.
- Is-Best percentage: how often the translation was top-ranked among the four translations.
- Is-Better percentage: how often the translation was judged as the better translation, over all pairwise comparisons extracted from the ranks.

6.3.2 Parameter Tuning

After feature values are computed for the sentences, we must set the model's corresponding weight vector $\vec{w}$. Naturally, the weights should be chosen so that good translations get high scores, and bad translations get low scores. We optimize translation quality as measured by BLEU against a small subset (10%) of the reference (professional) translations, using the search method of Och (2003) (also described in 5.4.2).[6]

6.3.3 The Worker Calibration Feature

As indicated in 6.3.2, a small portion of the reference translations is used to perform weight tuning. Note that those translations are *not* needed to compute any of the features introduced in 6.3.1, since those features are functions only of the data we collect (and, for one kind of features, the source sentences).

[5] Close to 90% of translations come from translators located in those two countries. Amazon has enabled payments in Indian rupees, which has attracted a large demographic of workers from India (Ipeirotis, 2010a). Although it does not yet have direct payment in Pakistani Rupee, we found that a large contingent of the workers are located in Pakistan.

[6] Unlike Och's MERT algorithm, where new candidate translations are generated repeatedly based on updated weights, our set of candidate translations is *fixed* (14 per source sentence). Since both use the same linear search method, one can think of our optimization as a single 'iteration' of MERT.

Since we use a small portion of the reference translations to perform weight tuning, we can also use that data to compute another worker-specific feature. Namely, we can evaluate the competency of each worker by scoring their translations against the reference translations. We then use that feature for every translation given by that worker. The intuition is that workers known to produce good translations are likely to continue to produce good translations, and the opposite is likely true as well.

6.4 Experimental Results

6.4.1 Evaluation Strategies

To confirm that our approach is valid, we need to demonstrate that the obtained translations are of an acceptably high quality. Informal examination of the translations is clearly not adequate, and we must instead use specific quantitative measures. To measure the quality of the translations, we make use of the existing professional translations. Since we have *four* professional translation sets, we can calculate the BLEU score (Papineni et al., 2002) for one professional translator P_1 using the other three $P_{2,3,4}$ as a reference set. We repeat the process four times, scoring each professional translator against the others, to calculate the expected range of professional quality translation.

We can then see how a translation set T (chosen by our model) compares to this range by calculating T's BLEU scores against the same four sets of three reference translations. We will evaluate different strategies for selecting such a set T, and see how much each improves on the BLEU score, compared to randomly picking from among the Turker translations.

In a separate evaluation of the Turkers' translation quality, we use the crowdsourced translations to replace professional translations as references when scoring various submissions to the NIST MT evaluation. We measure the correlation (using Pearson's r) between BLEU scores of MT systems measured against non-professional translations, and BLEU scores measured against professional translations. Since the

main purpose of the NIST dataset was to compare MT systems against each other, this is a more direct fitness-for-task measure.

For purposes of measuring BLEU correlation, we chose the middle 6 systems (in terms of performance) submitted to the NIST evaluation, out of 12, as those systems were fairly close to each other, with less than 2 BLEU points separating them. Using all 12 systems would have artificially inflated correlation, due to the vast differences between the systems. For instance, the top system outperforms the bottom system by **15** BLEU points!

In the remainder of this Section, we establish the performance of professional translators, calculate oracle upper bounds on Turker translation quality, and carry out a set of experiments that demonstrate the effectiveness of our model and that determine which features are most helpful.

Each number reported in this Section is an average of four numbers, corresponding to the four possible ways of choosing 3 of the 4 reference sets. Furthermore, each of those 4 numbers is itself based on a five-fold cross validation, where 80% of the data is used to compute feature values, and 20% used for evaluation. The 80% portion is used to compute the aggregate worker-level features. For the worker calibration feature, we use the references for 10% of the data (which is within the 80% portion).

6.4.2 Translation Quality: BLEU Scores Against Professionals

We first evaluated the reference sets against each other, to quantitatively gauge the level at which translations would approach "professional quality". On average, evaluating one reference set against the other three gives a BLEU score of 42.38 (Figure 6.4). A Turker set of translations scores 28.13 on average, which highlights the loss in quality when collecting translations from amateurs. To emphasize how drastic a gap this is, we note that the output of a state-of-the-art machine translation system (the syntax-based variant of Joshua (Li et al., 2010)) achieves a score of 26.91, a mere 1.22 worse than the Turkers.

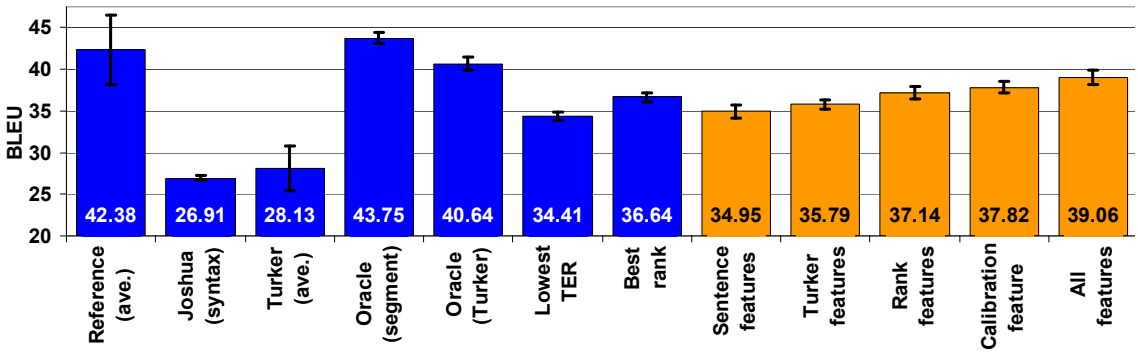


Figure 6.4: BLEU scores for different selection methods, measured against the reference sets. Each score is an average of four BLEU scores, each calculated against three LDC reference translations. The five right-most bars are colored in orange (and black font) to indicate selection over a set that includes both original translations as well as edited versions of them.

We also perform two types of oracle experiments, to determine if there exist high-quality Turker translations in the first place. The first oracle operates on the segment level: for each source segment, choose from the four translations the one that scores highest against the reference sentence. The second oracle scenario is a worker-based approach: for each source segment, choose from the four translations the one provided by the worker whose translations (over all sentences) score the highest. The two oracles achieve BLEU scores of 43.75 and 40.64, respectively – easily within the range of professional translators.

We then examined two voting-inspired methods, since taking a majority vote usually works well when dealing with MTurk data. The first is to select the translation with the minimum average TER against the other three translations, since that would be a 'consensus' translation. The second method is to select the translation that received the best average rank, using the rank labels assigned by other Turkers (see 6.2.2). These approaches achieve BLEU scores of 34.41 and 36.64, respectively.

The main set of experiments evaluated the features from 6.3.1 and 6.3.3. We applied our approach using each of the four feature types: sentence features, Turker features, rank features, and the calibration feature. That yielded BLEU scores ranging from 34.95 to 37.82. With all features combined, we achieve a higher score of 39.06, which is within the range of scores for the professional translators.

6.4.3 Fitness for a Task: Ranking MT Systems

In our second evaluation setup, we evaluated the selection methods by measuring correlation with the references, in terms of BLEU scores assigned to outputs of MT systems. This would mirror a real-world application of the created dataset, since the Urdu–English dataset was used by NIST to evaluate the submitted outputs of several MT systems.

Table 6.2 shows how well each selection strategy mimics the reference translations in ranking the submissions (by the systems' BLEU scores). For each strategy, we compute the correlation between its ranking and the ranking obtained using the reference sets. As with the previous evaluation setup, we report the average of four such correlations, one for each possible choice of 3 out of the 4 refrence sets.

The results tell a fairly similar story as evaluating with BLEU: references and oracles naturally perform very well, and the loss in quality when selecting arbitrary Turker translations is largely eliminated using our selection strategy. The benefit of the different types of features is also similar.

Interestingly, when using the output of either Joshua system as a reference set, the performance is quite abysmal. Even though their BLEU scores are comparable to the Turker translations (Figure 6.4), they are much worse when distinguishing closely matched MT systems from each other.[7]

6.4.4 Analysis

The oracle results indicate that there is usually an acceptable translation from the Turkers for a given Urdu sentence. Since the oracles select from a small group of only 4 translations per source segment, they are not overly optimistic or completely out of reach, and rather reflect the true potential of the collected translations.

The results indicate that, although some features are more useful than others, much of the benefit from combining all the features can be obtained from any one set of features, with the benefit of adding more features being somewhat orthogonal.

[7]It should be noted that the two Joshua systems were *not* part of the six MT systems we scored in the correlation experiments.

Selection Method	Pearson's r^2 ($\pm$ std. dev.)
Reference (ave.)	0.81 $\pm$ 0.07
Joshua (hiero)	0.04 $\pm$ 0.05
Joshua (syntax)	0.08 $\pm$ 0.09
Turker (ave.)	0.60 $\pm$ 0.17
Oracle (segment)	0.81 $\pm$ 0.09
Oracle (Turker)	0.79 $\pm$ 0.10
Lowest TER	0.50 $\pm$ 0.26
Best rank	0.74 $\pm$ 0.17
Sentence features	0.56 $\pm$ 0.21
Turker features	0.59 $\pm$ 0.19
Rank features	0.75 $\pm$ 0.14
Calibration feature	0.76 $\pm$ 0.13
All features	0.77 $\pm$ 0.11

Table 6.2: The ability of different selection methods to reproduce a BLEU ranking of 6 MT systems, measured by average correlation between a method's ranking and that obtained using the reference sets. Each value is the average of four such correlation calculations, one for each possible choice of 3 out of the 4 reference sets. The first line is also an average of four values, one per reference set (measured against the other three).

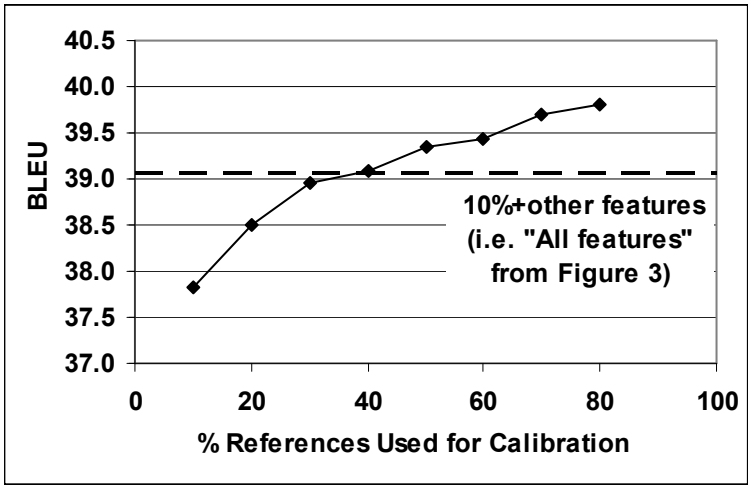


Figure 6.5: The effect of varying the amount of calibration data (and using only the calibration feature). The 10% point (BLEU = 37.82) and the dashed line (BLEU = 39.06) correspond to the two right-most bars of Figure 6.4.

We performed a series of experiments exploring the calibration feature, varying the amount of gold-standard references from 10% all the way up to 80%. As expected, the performance improved as more references were used to calibrate the translators (Figure 6.5). What's particularly important about this experiment is that it shows the added benefit of the other features: We would have to use 30%–40% of the references to get the same benefit obtained from combining the non-calibration features and only 10% for the calibration feature (dashed line in the Figure; BLEU = 39.06).

While the combined cost of our data collection effort ($2,500; see 6.2.3) is quite low considering the amount of collected data, it would be more attractive if the cost could be reduced further without losing much in translation quality. To that end, we investigated lowering cost along two dimensions: eliminating the need for professional translations, and decreasing the amount of edited translations.

The professional translations are used in our approach for computing the worker calibration feature (subsection 6.3.3) and for tuning the weights of the other features. We use a relatively small number of translations for this purpose, but we investigate a different setup whereby no professional translations are used at all. This eliminates the worker calibration feature, but, perhaps more critically, the feature weights must

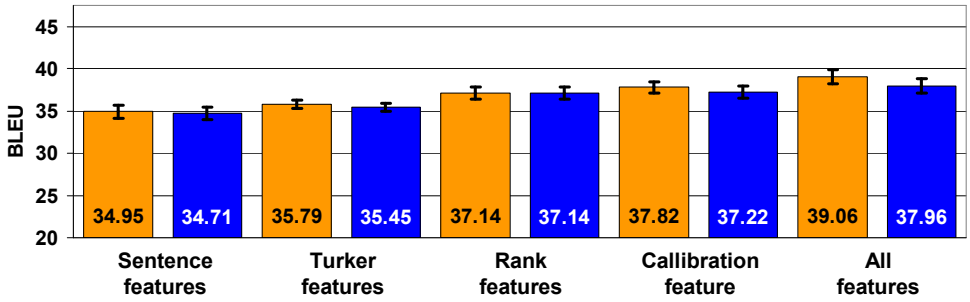


Figure 6.6: BLEU scores for the five right-most setups from Figure 6.4, constrained over the original translations (blue columns, white font). The scores from Figure 6.4 are repeated here for clarity (orange columns, black font).

be set in a different fashion, since we cannot optimize BLEU on reference data anymore. Instead, we use the rank labels (from 6.2.2) as a proxy for BLEU, and set the weights so that better ranked translations receive higher scores.

Note that the rank features will also be excluded in this setup, since they are perfect predictors of rank labels. On the one hand, this means no rank labels need to be collected, other than for a small set used for weight tuning, further reducing the cost of data collection. On the other hand, this leads to a significant drop in performance, yielding a BLEU score of 34.86.

Another alternative for cost reduction would be to reduce the number of collected edited translations. To that end, we first investigate completely eliminating the editing phase, and considering only unedited translations. In other words, the selection will be over a group of four English sentences rather than 14 sentences. Completely eliminating the edited translations has an adverse effect, as expected (Figure 6.6). Another option, rather than eliminating the editing phase altogether, would be to consider the edited translations of only the translation receiving the best rank labels. This would reflect a data collection process whereby the editing task is delayed until after the rank labels are collected, with the rank labels used to determine which translations are most promising to post-edit (in addition to using the rank labels for the ranking features). Using this approach enables us to greatly reduce the number of edited translations collected, while maintaining good performance, obtaining a BLEU score of 38.67.

It is therefore our recommendation that crowdsourced translation efforts adhere to the following pipeline: collect multiple translations for each source sentence, collect rank labels for the translations, and finally collect edited versions of the top ranked translations.

6.5 Crowdsourcing Translation of Dialectal Arabic

Our success in obtaining high-quality translations via crowdsourcing has encouraged other crowdsourced translation efforts in other languages. We participated in one such effort (Zbib et al., 2012), which focused on dialectal Arabic. The effort utilizes Arabic-speaking Turkers to perform dialect identification (using the methodology described in Chapter 3) followed by translation of dialectal sentences (using the methodology described in this Chapter). The Arabic source sentences are chosen from a large monolingual corpus consisting of the combination of ten Arabic LDC datasets. Following a filtration step to discard sentences that are either non-Arabic or can be easily identified as being MSA (due to high MSA content), a resulting set of 4M Arabic words is obtained.

The set of documents was passed through a dialect identification pipeline with similar quality control measures as described in Chapter 3. We kept the documents identified as being either Levantine (28% of documents) or Egyptian (11% of documents), and passed them on to a translation step.[8] This corresponded to a set of roughly 1.5M words.

The translation throughput reached a level of 200k words/week, thanks to a relatively high reward (in MTurk standards) and consistently quick reviewing of submitted translations. The total cost, including that of dialect identification (and an additional sentence-segmentation step), came out to about $44k, at a rate of $0.03/word.

[8]Documents in the Gulf/Iraqi dialect were not chosen even though they formed a plurality of documents (43% of documents), since an overwhelming majority of the Arabic-speaking Turkers were located outside the Gulf region and Iraq.

As with the Urdu-English translation effort, this is a full order of magnitude less expensive than the cost of having the sentences professionally translated.

Unlike the Urdu-English effort though, the amount of translated data was large enough to train statistical machine translation systems for Dialectal Arabic-to-English. To illustrate the benefit of using dialectal training data, several systems were trained using different datasets for training, and evaluated on both Egyptian and Levantine test sets. Table 6.3 shows a clear advantage for systems trained using the dialectal data over a system trained using only MSA data, in spite of training on up to 100 times more data. Example translations are given in Figure 6.7. The best performance is achieved when training on all the dialect data (which combines Levantine and Egyptian), with gains of several BLEU points. Interestingly, adding MSA training data has no added benefit – in fact this results in slight BLEU score drops.

We also investigated using MSA as a bridge language, by transforming the dialectal Arabic into MSA first, and then using an MSA-trained system to translate into English. For that purpose, we crowdsourced a task to manually transform the Levantine test set into MSA, to establish an optimistic estimate of what could be done automatically. When translating the transformed sentences with an MSA-trained system, the performance is only improved when the amount of dialectal training data is scarce. With no dialectal training data, performance improves by 2.3 BLEU points, but as more dialectal training data is added, the transformation actually hurts performance. This is an indication that matching the domain is also a very important factor, and the problem with an MSA-trained system goes beyond simple vocabulary coverage.

6.6 Related Work

Callison-Burch (2009) proposed several ways to evaluate MT output on MTurk. One such method was to collect reference translations to score MT output. It was only a pilot study (50 sentences in each of several languages), but it showed the possibility of obtaining high-quality translations from non-professionals. As a follow-

Arabic (Levantine): لهيك الجو كتيير كووول

Transliteration: *lhyk Aljw ktyyyr kwwwl*

Gloss: why the-weather veeery coool

MSA System: God you the atmosphere.

DA System: That's why the weather is very cool

Reference: this is why the weather is so cool

Arabic (Levantine): طول بالك عم نمزح

Transliteration: *Twl bAlk Em nmzH*

Gloss: lengthen mind-your are joking

MSA System: Do you think about a joke long.

DA System: Calm down we are kidding

Reference: calm down, we are kidding

Arabic (Egyptian): نفسي اطمئن عليه بعد ما شاف الصوره دي

Transliteration: *nfsY Atm}n Elyh bEd mA $Af AlSwrh dy*

Gloss: wish-my check on-him after that saw the-picture this

MSA System: Myself feel to see this image.

DA System: I wish to check on him after he saw this picture

Reference: I wish to be sure that he is fine after he saw this images

Arabic (Egyptian): انت بتعمل له اعلان ولا ايه ؟

Transliteration: *Ant btEml lh AElAn wlA Ayh ?*

Gloss: you make for-him advertisement or what ?

MSA System: You are working for a declaration and not ?

DA System: You are making the advertisement for him or what ?

Reference: Are you promoting it or what ?

Figure 6.7: Four dialectal Arabic sentences (two Levantine and two Egyptian) translated into English by two different MT systems. One system was trained on MSA-only data (**MSA System**), and the other was trained on the crowdsourced dialectal Arabic data (**DA System**).

Training Set	Training Set Size (words)	Egyptian Test		Levantine Test	
		BLEU	OOV	BLEU	OOV
MSA	150M	14.34	4.42%	12.29	5.53%
EGY	0.36M	19.04	4.62%	11.21	9.00%
LEV	1.10M	17.79	4.83%	**19.29**	3.31%
EGY+*LEV*	1.46M	**20.66**	2.85%	**19.29**	2.96%
EGY+*LEV*+MSA	150M+1.46M	20.09	**2.04%**	19.11	**2.27%**

Table 6.3: BLEU scores and OOV rates for two dialectal test sets, one Egyptian and one Levantine, using different training corpora. The *EGY* and *LEV* training components are crowdsourced translations, whereas the MSA component consists of professionally-produced translations from LDC data sources.

up, Bloodgood and Callison-Burch (2010) solicited a single translation of the NIST Urdu-to-English dataset we used. Their evaluation was similar to our correlation experiments, examining how well the collected translations agreed with the professional translations when evaluating three MT systems.

That paper appeared in a NAACL 2010 workshop organized by Callison-Burch and Dredze (2010), focusing on MTurk as a source of data for speech and language tasks. Two relevant papers from that workshop were by Ambati and Vogel (2010) and by Irvine and Klementiev (2010). The former focused on the design of the translation interface, in particular the impact of the provided context and of its quantity. The latter dealt with creating translation lexicons between English and 42 rare languages, for which we believe MTurk is perfect, given its international nature.

Two other workshops concerning crowdsourcing, focusing on translation in particular, were held at the University of Maryland (Bederson and Resnik, 2010), and at the 2010 meeting of the Association for Machine Translation in the Americas (Désilets, 2010). The goal was to facilitate discussion among a group of individuals with various backgrounds, computational and professional, regarding crowdsourcing efforts, future directions, and influence on the providers and users of translation services.

Resnik et al. (2010) explored a very interesting way of creating translations on MTurk, relying only on *monolingual* speakers. Speakers of the target language iter-

atively identified problems[9] in an initial machine translation output, and speakers of the source language paraphrased the corresponding source portion. The paraphrased source would then be re-translated to produce a different translation, hopefully more coherent than the original.

There are crowdsourced translation efforts outside of MTurk as well. IBM's n.Fluent translation software is able to learn new and better translations using feedback from IBM's employees using the software. As of 2009, more than 40M words had been translated, at a daily rate of up to 100k words (Fishkind, 2009). Facebook's `Translations` app[10] enables the translation of the social network's content into many languages, by crowdsourcing the translation task to users of the website. The app follows the translation step with a voting step, where other users can vote a submitted translation up or down.

6.7 Conclusion

We have demonstrated that it is possible to obtain high-quality translations from non-professional translators via crowdsourcing, and that the cost is an order of magnitude cheaper than professional translation. The translators are native speakers of the source language who have little to no professional translation experience, and who are not native speakers of the target language.

We presented a translation selection approach and demonstrated its effectiveness by reproducing the English translations in an Urdu-to-English evaluation set, and showing that we achieve translation quality approaching that of professionally-produced translations. Our quality control measures took a number of forms, from simple things like rendering text as images (to prevent copying and pasting into an MT system) to more involved measures like crowdsourcing an entire evaluation task to rank the translations.

Our effort to crowdsource translations of dialectal Arabic content resulted in the creation of the first parallel corpora dedicated to dialectal varieties of Arabic. The

[9] Problems can also be identified using an automatic method that is based on back-translation.
[10] http://www.facebook.com/translations

translation quality of an MT system trained on this data was significantly better than an MSA system trained on 100 times as much MSA-only data. The only way to collect such translations in large quantities, and do it efficiently and at a low cost, is via crowdsourcing.

Chapter 7

Conclusion

Using statistical learning methods has become a standard approach for solving most NLP tasks, but supervised learning often requires a set of training examples annotated by human judges with the correct answers. Given the high overhead and cost of creating such training sets, most research relies on existing datasets rather than create new ones. This proves to be a significant obstacle when tackling a new domain or a new language for which no or few training sets exist, creating a data bottleneck that could severely hinder further research.

In this thesis, we demonstrate that **crowdsourcing** is extremely beneficial to data collection and creation for NLP applications. We examined a variety of domains and investigated a spectrum of annotation tasks, from the simple (labeling) to the complex (translating entire sentences). The presented research is novel as it involves new groups of **annotators**, new types of **annotation schemes**, new types of **data**, and new **algorithms** to handle such data. We used crowdsourcing to create training datasets for tasks for which no training data had previously existed, and we showed that complex tasks can be delegated to non-professionals and completed at a fraction of the cost of hiring professionals. Even though crowdsourcing can result in some low-quality data, we presented several approaches for detecting low-quality data, and showed that high-quality data can be obtained when effective quality control measures are implemented.

7.1 Major Contributions

Human annotators possess extremely valuable knowledge. We should harness this knowledge by letting annotators express it in an easy and natural manner. At the same time, it is important that we are able to collect data affordably and efficiently. The take-home message of this thesis is that crowdsourcing allows us to achieve exactly that.

We implemented a range of strategies to exercise quality control over crowdsourced annotations, and were successful in producing data of high quality. Our strategies enable us to detect spammers performing the task unfaithfully, as well as annotators who are willing but unable to perform the task correctly. Our effort to crowdsource the translation task is an example of the importance of quality control. We used a machine-learning-based method for evaluating translators and their submissions, by designing a set of easy-to-compute and intuitive features to score each translation and each translator. We also crowdsourced an entire quality evaluation task to evaluate the collected translations. While unfiltered crowdsourced translations are not much better than MT system output, our approach allows us to select translations of much higher quality, resembling translations provided by professional translators.

We introduced several new types of annotation schemes, along with new models that can use the resulting data. For instance, we introduced *annotator rationales*, an entirely new type of data that allows an annotator to communicate much more information about their decision process, compared to providing only class labels. We designed and evaluated two methods that incorporate rationales into their training. The first is a modification to support vector machines that reflects an intuitive geometric way to incorporate rationales via *contrast examples*. The second involves explicitly modeling the rationale annotation process using a CRF, integrating the model into the training objective of a log-linear model. Both methods achieve significant accuracy improvements over the corresponding baselines, with the benefit of adding rationales justifying the cost to collect them

We showed that we can crowdsource NLP annotation tasks at a significantly lower cost than hiring professionals. We found that the cost of crowdsourcing annotation is

invariably a fraction of the cost of hiring and training professional annotators. Crowdsourcing the translation task cost an order of magnitude less than hiring professional translators. We collected labels for the Arabic dialect identification task at a cost of less than a penny per label. We collected translation acceptability judgments for our RYPT metric at a rate of 161 labels per dollar. Besides monetary cost, all of our tasks showed that we can collect annotations on MTurk *quickly*. We collected over 300k labels for the Arabic dialect identification task in under 5 months. In our translation task, throughput reached a level of 200k words per week.

We created several datasets that had previously been impossible to create. In the case of dialectal Arabic for example, we relied on a pool of workers located in Middle Eastern countries to create two novel datasets. We created a dataset for Arabic dialect identification, where each sentence is annotated for the level and type of dialectal content. Another dataset was created by soliciting English translations of dialectal Arabic content. In both cases, the resulting datasets were the largest known of their kind. The datasets were used to train dialectal Arabic language models and MT systems, which would have been impossible otherwise. We showed that these models, trained on dialectal data, dramatically outperform models trained on (much more) MSA-only data when evaluated on dialectal input.

7.2 The Future of Mechanical Turk

The need for a large and representative training set could deter researchers from taking on tasks and domains in which little or no annotated data exists. The cost associated with creating an annotated training set has traditionally been prohibitively high, in terms of monetary cost, effort put into hiring and training annotators, and time needed to collect the annotations. We submit that crowdsourcing can effectively tackle these challenges, and makes the creation of such datasets quick and affordable. We envision a day when crowdsourcing is as logical a choice for data creation as the more traditional and controlled "face-to-face" approach.

To reach that end, we believe the research community should focus on the following

issues:

- **Seeing beyond MTurk's cost reduction:** While the cost reduction associated with crowdsourcing is often dramatic, we feel that emphasizing that factor alone neglects other important advantages of crowdsourcing. Characterizing MTurk as simply a source of "cheap labor" is a red herring, as it distracts away from the **crowd** portion of crowdsourcing. With crowdsourcing, we have the ability to parallelize seemingly impossible annotation tasks and have them completed rapidly. This is because of the large number of Turkers, but also because MTurk is an on-demand service, where tasks start being completed almost instantly after posting them. Furthermore, we must acknowledge the fact that crowdsourcing breaks down several barriers of entry and gives access to international workers, who possess skill sets that are not accessible otherwise.

- **Effective knowledge and resource sharing:** Newcomers to crowdsourcing must get a feel for how to best attract annotators to their tasks, in terms of how high monetary rewards should be, and how the task should be designed to make it appealing to Turkers. Beyond that, however, those Requesters must establish a name for themselves from scratch, design their own set of qualifications, and build a community of 'loyal' Turkers who are on the lookout for more tasks. It would be greatly beneficial if such resources could be shared between researchers in a streamlined and efficient manner, for example via a repository of qualifications and tests that are available for use by all researchers, and by designing an effective feedback system that allows Requesters to evaluate Turkers (beyond Amazon's simple Approval Rating measure).

- **A better reputation system.** The previous point essentially advocated measures to help protect against spammy Turkers. Unfortunately, MTurk's current reputation system is not very powerful, as it reports very simple metrics about a Turker's past work and does not give a comprehensive history. For instance, Turkers who are proficient at a particular task (and have submitted a lot of approved work for it) might not be at all suitable for another task, yet there would

be nothing to indicate that fact in their approval rating. The CrowdFlower service alleviates this issue to some degree, by making it easy for Requesters to embed control items in their MTurk HITs. The service uses those items not only to verify the quality of submitted work, but also to track a Turker's history *for that specific task*. This makes it possible to identify proficient Turkers even if they don't have an extensive history of approved HITs by other Requesters.

- **Iterative data filtering:** Most data quality approaches rely on collecting redundant annotations, and attempting to predict and correct deviations in an annotator's work from the ideal behavior. That said, one of the most effective mechanisms of quality control is the design of subsequent data filtering annotation tasks (performed by humans as well) that follow the main data collection phase itself, such as simple up/down votes or edits by native speakers. This approach has not received much attention within the research community, possibly because it is harder to formulate theoretically and could be task-specific to some degree. It would be imperative to design software tools that can streamline the iterative filtering process. Ideally, such software would be used as a black box, and would become standard practice when crowdsourcing annotation tasks. One tool built to accomplish that is TurKit[1], and it would be very beneficial if it is adopted and expanded by the research community.

- **Re-examining the concept of "gold-standard" data:** Researchers almost always assume that there exists a gold-standard for any annotation task. Such gold-standard data are provided by professionals or well-trained judges who follow a particular, sometimes strict set of guidelines, and their labels are treated as the ground truth. However, there are times when gold-standard data should be taken with a grain of salt, and it may not always be trusted blindly. One such example was the editing task of Chapter 5, where the LDC editor's actions were treated as a gold standard, under which Turkers seemed unable to perform the editing task adequately. Indeed, there is evidence that many of our editors were not performing the task properly. But the task was difficult to replicate even for

[1] http://groups.csail.mit.edu/uid/turkit/

a second professional editor, who was also hired by the LDC and underwent the same training. In that spirit, we recommend that research on quality control place more emphasis on devising filtering strategies that do not rely on gold-standard data. This would also allow us to filter noisy labels in tasks for which no gold-standard labels exist in the first place.

7.3 Future Work

We established that crowdsourced efforts benefit from collecting redundant annotations for purposes of quality control. This is usually done in a uniform fashion: collect k labels for each item, where k is held constant for all the items. One useful research topic would be to identify when another label is actually needed to begin with. Should we start with a very small value for k, say 2, and only collect a third label when the first two annotators disagree? Sheng et al. (2008) do investigate this question,[2] but let's take the idea to an extreme. It might even be possible to collect a *single* label per item. We would use other measures (gold-standard data, label distribution) to identify the most clearly faithful annotators, and simply trust their answers. Only items labeled by other annotators are then passed on to the next phase for redundant label collection.

Turning now to the particular tasks of the thesis, our dialect identification experiments (Chapters 3 and 4) did not consider data sources other than the Arabic Online Commentary Dataset (AOC). That said, AOC-trained classifiers can be used to classify data from other data sources, such as Twitter or Facebook posts. The letter-based models in particular might be more suitable, since they are less domain-dependent than word models. Either way, it would be worthwhile to have a human judge annotate a small set of posts from the new data source. For one thing, that would help evaluate the models' performance. Another practical use would be to adjust the priors of the classifier, to reflect the new source's MSA/dialect distribution. It is conceivable that the fraction of dialectal content in Twitter and Facebook posts

[2]Code available on GitHub: https://github.com/ipeirotis/Get-Another-Label

is even higher than in the AOC.

There are also non-lexical features that would be useful for dialect identification. Namely, some personal traits of the communicants (e.g. gender, age, status) can be helpful in determining whether they are likely to use dialect or not, and some traits (e.g. location) would certainly go a long way in determining *which* dialect they use. Unfortunately, the AOC dataset has very little such information. Furthermore, readers were not required to register, and so a single handle (such as a common first name) could be used by many readers. (Conversely, the same reader may use several handles.) It is therefore difficult to even aggregate posts by reader, for purposes of computing priors for an individual. In other data sources (Facebook, Twitter, forums) commenters are more differentiable from each other than in the AOC, and we therefore recommend taking full advantage of communicant features. Another helpful clue would be the topic being discussed. For example, dialectal Arabic is more likely to appear when discussing entertainment or sports, while MSA would be preferred in topics on politics or religion. In the case of the AOC, topic-prediction models do not have to be limited to the comments, as they can also examine the article being discussed.

One of our classification methods has an advantage beyond its ability to classify documents. The CRF that we used to model rationales as a tag sequence (Chapter 4) could be extended to *generate* those sequences for a given document, hence predicting where the rationales in a document are. In the case of dialect identification, this would be used for classification on a sub-sentential level, to identify dialectal portions of Arabic text. Such information could be very useful in machine translation, for example, where such segments would be translated using a dialect-trained system. In the case of movie (or product) reviews, a rationale prediction model would be able to highlight the most important snippets of the review.

Rationales are a form of side information that helped learners estimate better parameters. Other data we collected can also be helpful side information, even though we did not incorporate them into the training of a statistical learner. Specifically, the datasets of Chapter 5 were sentence edits and acceptability judgments, which we

collected to compute HTER scores and RYPT scores, respectively. These judgments, by definition, highlight the deficiencies of an MT system by pointing out problems in its output. We therefore propose that such data could help *improve* MT output rather than simply evaluate it. For example, the RYPT judgments can be incorporated directly into the training of an MT system, as they could help in computing better probabilities for phrase table entries, especially those corresponding to less frequent phrases. One can also imagine that the collected judgments could play a role in improving MT output in a *post hoc* fashion. For example, the collected edits can be used to train automatic 'editors' good at detecting sentence positions likely to require editing (and that could suggest likely edit actions). Such a tool could be useful even for (human) post-editors, since it would help them spend less of their time finding problems in the output, and focus instead on actually fixing them.

We truly believe that crowdsourcing will play a pivotal role in future efforts to create parallel translation datasets, especially for rare languages for which obtaining professional translations is difficult. The translations we collected (Chapter 6) are significantly cheaper than professional translations, with the cost most likely growing sublinearly with time, as we manage to establish a group of trusted translators known to produce high-quality translations. If we wish to build datasets for such rare languages, and have them comparable in size to, say, the Europarl Parallel Corpus (Koehn, 2005), we should give serious consideration to a $0.5M option of crowdsourcing translation, versus a $5M option of hiring professional translators.

Appendix A

The Buckwalter Transliteration Scheme

The Arabic transliteration scheme used in the thesis (particularly in Chapter 3) is the Buckwalter transliteration (BT) mapping, designed by lexicographer Tim Buckwalter in the 1990s (Buckwalter, 2002). BT relies on ASCII characters to represent Arabic orthography, by designating a single, distinct ASCII character for each Arabic letter.

Figure A.1 lists the ASCII characters used in BT. The given list is divided into four sections: vowels, forms of the *hamzah* (glottal stop), consonants, and pharyngealized consonants. Pharyngealized consonants are 'thickened' versions of other, more familiar consonants, voiced such that the pharynx or epiglottis is constricted during the articulation of the sound. Those consonants are present in very few languages and are therefore likely to be unfamiliar to most readers, which is why they are placed in a separate section – there is no real distinction in Arabic between them and other consonants.

BT also allows for the expression of short vowels and other Arabic diacritics, but since those diacritics are only rarely expressed in written (and typed) form, we omit them for clarity. Readers interested in the full table can find it on Buckwalter's website: http://www.qamus.org/transliteration.htm.

	ASCII	Arabic	Pronunciation Guide
	A	ا	The vowel '**a**' (e.g. f**a**ther or c**a**t)
→	*p*	ة	The vowel '**a**' (only appears at word's end, e.g. Al-Manam**ah**)
→	*Y*	ى	The vowel '**a**' (only appears at word's end, e.g. Mon**a**)
	w	و	The vowel '**o**' (e.g. h**o**me, s**oo**n), or the consonant '**w**' (e.g. **w**ait)
	y	ي	The vowel '**e**' (e.g. t**ee**n, r**ai**n), or the consonant '**y**' (e.g. **y**es)
	'	ء	Various forms of the Arabic letter *hamzah* , which is the glottal stop (the consonantal sound in '**u**h-**o**h', and the allophone of '**t**' in some pronunciations of bu**tt**on). Determining which form is appropriate depends on the location of the *hamzah* within the word, and the vowels immediately before and after it.
	|	آ	
	>	أ	
	&	ؤ	
	<	إ	
	}	ئ	
→	*$*	ش	**sh**oe
→	*	ذ	**th**e
	b	ب	**b**a**b**y
	d	د	**d**a**d**
	f	ف	**f**ather
→	*g*	غ	French Pa**r**is (guttural)
	H	ح	a raspier version of '**h**' (IPA: voiceless pharyngeal fricative)
	h	ه	**h**ouse
	j	ج	**j**ump or bei**g**e
	k	ك	**k**iss
	l	ل	**l**eaf
	m	م	**m**o**m**
	n	ن	**n**u**n**
	q	ق	like a '**k**' further back in the throat (IPA: voiceless uvular stop)
	r	ر	Scottish bo**rr**ow (rolled)
	s	س	**s**un
	t	ت	**t**en
→	*v*	ث	**th**ink
→	*x*	خ	German Ba**ch**, Spanish o**j**o
	z	ز	**z**ebra
	D	ض	Pharyngealized '**d**'
→	*E*	ع	Pharyngealized glottal stop (IPA: voiced pharyngeal fricative)
	S	ص	Pharyngealized '**s**'
	T	ط	Pharyngealized '**t**'
	Z	ظ	Pharyngealized '**th**' (of **th**e)

Figure A.1: The ASCII-to-Arabic mapping used in Buckwalter transliteration. Most mappings are straightforward; a few non-obvious mappings are highlighted above with an arrow (→) next to them.

Bibliography

Muhammad Arshad Ul Abedin, Vincent Ng, and Latifur Rahman Khan. Learning cause identifiers from annotator rationales. In *Proceedings of IJCAI*, pages 1758–1763, 2011.

Yaser S. Abu-Mostafa. Hints. *Neural Computation*, 7:639–671, 1995.

Vamshi Ambati and Stephan Vogel. Can crowds build parallel corpora for machine translation systems? In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, pages 62–65, 2010.

Shilpa Arora and Eric Nyberg. Interactive annotation learning with indirect feature voting. In *Proceedings of the NAACL HLT Student Research Workshop and Doctoral Consortium*, pages 55–60, 2009.

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 MT and/or Summarization*, pages 67–72, 2005.

Y. Bar-Hillel. The present state of research on mechanical translation. *American Documentation*, 2(4):229–237, 1951.

Ben Bederson and Philip Resnik. Workshop on crowdsourcing and translation. http://www.cs.umd.edu/hcil/monotrans/workshop/, 2010.

Fadi Biadsy, Julia Hirschberg, and Nizar Habash. Spoken Arabic dialect identification

using phonotactic modeling. In *Proceedings of the EACL Workshop on Computational Approaches to Semitic Languages*, pages 53–61, 2009.

Michael Bloodgood and Chris Callison-Burch. Using Mechanical Turk to build machine translation evaluation sets. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, pages 208–211, 2010.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Tim Buckwalter. Buckwalter Arabic transliteration. http://www.qamus.org/transliteration.htm, 2002.

Chris Callison-Burch. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of EMNLP*, pages 286–295, 2009.

Chris Callison-Burch and Mark Dredze. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, pages 1–12, 2010.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluating the role of Bleu in machine translation research. In *Proceedings of EACL*, pages 249–256, 2006.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. (meta-) evaluation of machine translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, pages 136–158, 2007.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. Further meta-evaluation of machine translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, pages 70–106, 2008.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar F. Zaidan. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, 2010.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar F. Zaidan. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the EMNLP Workshop on Statistical Machine Translation*, pages 22–64, 2011.

William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR-94*, pages 161–175, 1994.

Daniel Cer, Daniel Jurafsky, and Christopher Manning. The best lexical metric for phrase-based statistical MT system optimization. In *Proceedings of NAACL HLT*, pages 555–563, 2010.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. Reading tea leaves: How humans interpret topic models. In *Proceedings of NIPS*, 2009.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.

David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33 (2):201–228, 2007.

David Chiang, Mona Diab, Nizar Habash, Owen Rambow, and Safiullah Shareef. Parsing Arabic dialects. In *Proceedings of EACL*, pages 369–376, 2006.

David Chiang, Steve DeNeefe, Yee Seng Chan, and Hwee Tou Ng. Decomposability of translation metrics for improved evaluation and efficient algorithms. In *Proceedings of EMNLP*, pages 610–619, 2008.

Kenneth Church and Robert Mercer. Introduction to the special issue on computational linguistics using large corpora. *Computational Linguistics*, 19(1):1–24, 1993.

Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

Corinna Cortes and Vladimir N. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Deborah Coughlin. Correlating automated and human assessments of machine translation quality. In *Proceedings of MT Summit IX*, pages 63–70, 2003.

Hal Daumé III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.

A. Philip Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28, 1979.

Ofer Dekel and Ohad Shamir. Vox populi: Collecting high-quality labels from a crowd. In *Proceedings of COLT*, 2009a.

Ofer Dekel and Ohad Shamir. Good learners for evil teachers. In *Proceedings of ICML*, pages 233–240, 2009b.

Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Michael Denkowski and Alon Lavie. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the EMNLP Workshop on Statistical Machine Translation*, pages 85–91, 2011.

Alain Désilets. AMTA 2010 workshop on collaborative translation: Technology, crowdsourcing, and the translator perspective. http://www.wiki-translation.com/AMTA+2010+Workshop, 2010.

Mona Diab, Nizar Habash, Owen Rambow, Mohamed Altantawy, and Yassine Benajiba. COLABA: Arabic dialect annotation and processing. In *Proceedings of the LREC Workshop on Semitic Language Processing*, pages 66–74, 2010.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of HLT*, pages 138–145, 2002.

Jeff Donahue and Kristen Grauman. Annotator rationales for visual recognition. In *Proceedings of the International Conference on Computer Vision*, 2011.

Pinar Donmez, Jaime G. Carbonell, and Jeff G. Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of KDD*, pages 259–268, 2009.

Bonnie J. Dorr. Machine translation evaluation and optimization. In Joseph Olive, John McCary, and Caitlin Christianson, editors, *Handbook of Natural Language Processing and Machine Translation*. Springer, 2010. Part 5.

Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of ACM Special Interest Group on Information Retrieval (SIGIR)*, 2008.

D. Christopher Dryer, Chris Eisbach, and Wendy S. Ark. At what cost pervasive? A social computing view of mobile computing systems. *IBM Systems Journal*, 38(4): 652–676, 1999.

Amit Dubey. What to do when lexicalization fails: Parsing German with suffix analysis and smoothing. In *Proceedings of ACL*, pages 314–321, 2005.

Susan Dumais. Using SVMs for text categorization. *IEEE Intelligent Systems Magazine*, 13(4), 1998.

T. Dunning. Statistical identification of language. Technical Report MCCS 94-273, New Mexico State University, 1994.

Encarta. http://www.webcitation.org/5kwPyXpQ1, 2007.

Ari Fishkind. Made in IBM labs: IBM researchers lower language barrier with text translator. http://www-03.ibm.com/press/us/en/pressrelease/28887.wss, November 2009.

Andrew Frank and Arthur Asuncion. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2010.

Nikesh Garera and David Yarowsky. Modeling latent biographic attributes in conversational genres. In *Proceedings of ACL*, pages 710–718, 2009.

Ulrich Germann. Building a statistical machine translation system from scratch: How much bang for the buck can we expect? In *Proceedings of the ACL Workshop on Data-Driven Machine Translation*, 2001.

Dan Gillick and Yang Liu. Non-expert evaluation of summarization systems is risky. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, 2010.

Nizar Habash. Four techniques for online handling of out-of-vocabulary words in Arabic-English statistical machine translation. In *Proceedings of ACL, Short Papers*, pages 57–60, 2008.

Nizar Y. Habash. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool, 2010.

Aria Haghighi and Dan Klein. Prototype-driven learning for sequence models. In *Proceedings of HLT-NAACL*, pages 320–327, 2006.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL*, 2008.

Jeff Howe. The rise of crowdsourcing. *Wired*, 14(6), 2006.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

Pei-Yun Hsueh, Prem Melville, and Vikas Sindhwani. Data quality from crowdsourcing: A study of annotation selection criteria. In *Proceedings of the NAACL HLT Workshop on Active Learning for Natural Language Processing*, pages 27–35, 2009.

Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *In Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 64–67, 2010.

Panos Ipeirotis. New demographics of Mechanical Turk. http://behind-the-enemy-lines.blogspot.com/2010/03/new-demographics-of-mechanical-turk.html, 2010a.

Panos Ipeirotis. Mechanical Turk: Now with 40.92% spam. http://www.behind-the-enemy-lines.com/2010/12/mechanical-turk-now-with-4092-spam.html, 2010b.

Ann Irvine and Alexandre Klementiev. Using Mechanical Turk to annotate lexicons for less commonly used languages. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, pages 108–113, 2010.

Mukund Jha, Jacob Andreas, Kapil Thadani, Sara Rosenthal, and Kathleen McKeown. Corpus creation for new genres: A crowdsourced approach to PP attachment. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, 2010.

Rong Jin and Zoubin Ghahramani. Learning with multiple labels. In *Proceedings of NIPS*, pages 897–904, 2003.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142, 1998.

Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Chris Burges, and Alex Smola, editors, *Advances in Kernel Meth-*

ods – Support Vector Learning, pages 169–185. MIT Press, 1999. Available at http://svmlight.joachims.org.

Michael Kaisser and John Lowe. Creating a research collection of question answer sentence pairs with Amazon's Mechanical Turk. In *Proceedings of LREC*, 2008.

David Kauchak and Regina Barzilay. Paraphrasing for automatic evaluation. In *Proceedings of the HLT-NAACL*, pages 455–462, 2006.

Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with Mechanical Turk. In *Proceedings of SIGCHI*, 2008.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, 2005.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133, 2003.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL, Demo and Poster Sessions*, pages 177–180, 2007.

Pirkko Kuusela and Daniel Ocone. Learning with side information: PAC learning bounds. *Journal of Computer and System Sciences*, 68(3):521–545, 2004.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289, 2001.

Edith Law and Luis von Ahn. *Human Computation*. Number 3 in Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, June 2011.

LDC. Linguistic data annotation specification: Assessment of fluency and adequacy in translations. http://www.ldc.upenn.edu/Projects/TIDES/Translation/TransAssess04.pdf, 2005.

Yun Lei and John H. L. Hansen. Dialect classification via text-independent training and testing for Arabic, Spanish, and Chinese. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(1):85–96, 2011.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of ACM-SIGIR*, pages 3–12, 1994.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar F. Zaidan. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the EACL Workshop on Statistical Machine Translation*, pages 135–139, 2009.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Ann Irvine, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Ziyuan Wang, Jonathan Weese, and Omar F. Zaidan. Joshua 2.0: A toolkit for parsing-based machine translation with syntax, semirings, discriminative training and other goodies. In *Proceedings of the ACL Workshop on Statistical Machine Translation and MetricsMATR*, pages 133–137, 2010.

Chang Liu, Daniel Dahlmeier, and Hwee Tou Ng. Better evaluation metrics lead to better machine translation. In *Proceedings of EMNLP*, pages 375–384, 2011.

Daniel Lopresti and Andrew Tomkins. Block edit models for approximate string matching. *Theoretical Computer Science*, 181(1):159–179, 1997.

Matthew Marge, Satanjeev Banerjee, and Alexander Rudnicky. Using the Amazon Mechanical Turk for transcription of spoken language. In *Proceedings of ICASSP*, 2010.

Ian McGraw, Alexander Gruenstein, and Andrew Sutherland. A self-labeling speech

corpus: Collecting spoken words with an online educational game. In *Proceedings of INTERSPEECH*, 2009.

Rada Mihalcea and Carlo Strapparava. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of ACL-IJCNLP, Short Papers*, 2009.

Joanna Mrozinski, Edward Whittaker, and Sadaoki Furui. Collecting a Why-question corpus for development and evaluation of an automatic QA-system. In *Proceedings of ACL*, pages 443–451, 2008.

Robert Munro, Steven Bethard, Victor Kuperman, Vicky Tzuyin Lai, Robin Melnick, Christopher Potts, Tyler Schnoebelen, and Harry Tily. Crowdsourcing and language studies: The new generation of linguistic data. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, 2010.

Dragos Munteanu and Daniel Marcu. Improving machine translation performance by exploiting comparable corpora. *Computational Linguistics*, 31(4):477–504, 2005.

Preslav Nakov. Noun compound interpretation using paraphrasing verbs: Feasibility study. In *Proceedings of the International Conference on Artificial Intelligence: Methodology, Systems and Applications (AIMSA)*, pages 103–117, 2008.

Sonja Nießen, Franz Josef Och, Gregor Leusch, and Hermann Ney. An evaluation tool for machine translation: Fast evaluation for MT research. In *Proceedings of LREC*, pages 39–45, 2000.

Scott Novotney and Chris Callison-Burch. Cheap, fast and good enough: Automatic speech recognition with non-expert transcription. In *Proceedings of NAACL*, pages 207–215, 2010.

Scott Novotney, Rich Schwartz, and Sanjeev Khudanpur. Unsupervised Arabic dialect adaptation with self-training. In *Interspeech*, pages 1–4, 2011.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167, 2003.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*, pages 295–302, 2002.

Joseph Olive. Global autonomous language exploitation (GALE), 2005. DARPA/IPTO Proposer Information Pamphlet.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*, pages 271–278, 2004.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.

Kishore Papineni, Salim Poukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, 2002.

Manoj Parameswaran and Andrew B. Whinston. Social computing: An overview. *Communications of the Association for Information Systems*, 19:762–780, 2007.

Jason Pontin. Artificial intelligence, with help from the humans. *The New York Times*, March 25, 2007.

Mark Przybocki, Kay Peterson, and Sebastian Bronsart. Official results of the NIST 2008 "Metrics for MAchine TRanslation" challenge (MetricsMATR08). http://nist.gov/speech/tests/metricsmatr/2008/results/, 2008.

Alexander J. Quinn and Benjamin B. Bederson. Human computation: A survey and taxonomy of a growing field. In *Proceedings of CHI*, 2011.

Hema Raghavan, Omid Madani, and Rosie Jones. Active learning on both features and instances. *Machine Learning Research*, 7:1655–1686, 2006.

Reinhard Rapp. Identifying word translations in non-parallel texts. In *Proceedings of ACL*, 1995.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. Collecting image annotations using Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*, 2010.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Machine Learning Research*, 11:1297–1322, April 2010.

Philip Resnik and Noah Smith. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380, 2003.

Philip Resnik, Olivia Buzek, Chang Hu, Yakov Kronrod, Alex Quinn, and Benjamin Bederson. Improving translation via targeted paraphrasing. In *Proceedings of EMNLP*, pages 127–137, 2010.

Joel Rose. Some turn to 'Mechanical' job search. http://marketplace.publicradio.org/display/web/2009/06/30/pm_turking/, 2009.

Sara Rosenthal, William J. Lipovsky, Kathleen McKeown, Kapil Thadani, and Jacob Andreas. Semiautomated annotation for prepositional phrase attachment. In *Proceedings of LREC*, 2010.

Wael Salloum and Nizar Habash. Dialectal to standard Arabic paraphrasing to improve Arabic-English statistical machine translation. In *Proceedings of the EMNLP Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*, pages 10–21, 2011.

Charles Schafer and David Yarowsky. Inducing translation lexicons via diverse similarity measures and bridge languages. In *Proceedings of CoNLL*, pages 146–152, 2002.

Dana Shapira and James A. Storer. Edit distance with move operations. In *Proceedings of the Annual Symposium on Computational Pattern Matching*, pages 85–98, 2002.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of KDD*, pages 614–622, 2008.

Jason R. Smith, Chris Quirk, and Kristina Toutanova. Extracting parallel sentences from comparable corpora using document level alignment. In *Proceedings of NAACL HLT*, pages 403–411, 2010.

Padhraic Smyth, Usama M. Fayyad, Michael C. Burl, Pietro Perona, and Pierre Baldi. Inferring ground truth from subjective labelling of Venus images. In *Proceedings of NIPS*, pages 1085–1092, 1994.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*, 2006.

Matthew Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of the EACL Workshop on Statistical Machine Translation*, pages 259–268, 2009.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast – but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*, pages 254–263, 2008.

Alexander Sorokin and David Forsyth. Utility data annotation with Amazon Mechanical Turk. In *Proceedings of Computer Vision and Pattern Recognition Workshop*, 2008.

Clive Souter, Gavin Churcher, Judith Hayes, John Hughes, and Stephen Johnson. Natural language identification using corpus-based models. *Hermes Journal of Linguistics*, 13:183–203, 1994.

Joseph P. Turian, Luke Shen, and I. Dan Melamed. Evaluation of machine translation and its evaluation. In *Proceedings of MT Summit IX*, pages 386–393, 2003.

Brijesh Verma, Hong Lee, and John Zakos. *An Automatic Intelligent Language Classifier*, volume 5507 of *Lecture Notes in Computer Science*, pages 639–646. SpringerLink, 2009.

Luis von Ahn. *Human Computation*. PhD thesis, Carnegie Mellon University, 2005.

Luis von Ahn, Ruoran Liu, and Manuel Blum. Peekaboom: A game for locating objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 55–64, 2006.

Radim Řehůřek and Milan Kolkus. *Language Identification on the Web: Extending the Dictionary Method*, volume 5449 of *Lecture Notes in Computer Science*, pages 357–368. SpringerLink, 2009.

Warren Weaver. Translation. In William N. Locke and A. Donald Booth, editors, *Machine Translation of Languages: Fourteen Essays (Reproduced; 1955)*, pages 15–23. Technology Press of the Massachusetts Institute of Technology, 1949.

John White and Theresa O'Connell. The ARPA MT evaluation methodologies: Evolution, lessons, and future approaches. In *Proceedings of AMTA*, 1994.

John S. White. Toward an automated, task-based MT evaluation strategy. In *Proceedings of the LREC Workshop on Machine Translation Evaluation*, 2000.

Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proceedings of NIPS*, 2009.

Ainur Yessenalina, Yejin Choi, and Claire Cardie. Automatically generating annotator rationales to improve sentiment classification. In *Proceedings of ACL, Short Papers*, pages 336–341, 2010.

Omar F. Zaidan and Chris Callison-Burch. Feasibility of human-in-the-loop minimum error rate training. In *Proceedings of EMNLP*, pages 52–61, 2009.

Omar F. Zaidan and Chris Callison-Burch. Predicting human-targeted translation edit rate via untrained human annotators. In *Proceedings of NAACL HLT*, pages 369–372, 2010.

Omar F. Zaidan and Chris Callison-Burch. The Arabic Online Commentary Dataset: An annotated dataset of informal Arabic with high dialectal content. In *Proceedings of ACL*, pages 37–41, 2011a.

Omar F. Zaidan and Chris Callison-Burch. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of ACL*, pages 1220–1229, 2011b.

Omar F. Zaidan and Chris Callison-Burch. Arabic dialect identification. *Computational Linguistics (submitted)*, 2012.

Omar F. Zaidan and Jason Eisner. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of EMNLP*, pages 31–40, 2008.

Omar F. Zaidan, Jason Eisner, and Christine Piatko. Using "annotator rationales" to improve machine learning for text categorization. In *Proceedings of NAACL HLT*, pages 260–267, 2007.

Rabih Zbib, Erika Malchiodi, Jacob Devlin, David Stallard, Spyros Matsoukas, Richarch Schwartz, John Makhoul, Omar F. Zaidan, and Chris Callison-Burch. Machine translation of Arabic dialects. In *Proceedings of NAACL HLT (accepted)*, 2012.

BIBLIOGRAPHY

Marc A. Zissman. Comparison of four approaches to automatic language identification of telephone speech. *IEEE Transactions on Speech and Audio Processing*, 4(1):31–44, 1996.

Vita

Omar Fayez Zaidan was born in Amman, Jordan in 1982, where he attended Al-Manhal Elementary School and then Jubilee High School, graduating in 2000. He completed his undergraduate studies at St. Lawrence University, graduating *summa cum laude* in 2004 with a B.Sc. in Computer Science (with Honors) and Mathematics (with Honors), with a minor in Chemistry. He then joined the Computer Science Ph.D. program at Johns Hopkins University, initially working on networks and quantum computing, under the supervision of Christian Scheideler. After his advisor's departure from Hopkins, he changed research topics and gradually became affiliated with the Center for Language and Speech Processing, working first with Jason Eisner and then Chris Callison-Burch, his thesis advisor. He received a M.S.Eng. in 2007 and completed his Ph.D. in 2011, both in Computer Science.

At Hopkins, he was head T.A. for several courses, including *Natural Language Processing* and *Artificial Intelligence*; taught the department's *Introduction to Java* summer offering in 2007; developed Z-MERT, an open source package for MT parameter tuning; created the *Arabic Online Commentary* dataset, consisting of over 50M words of Arabic reader commentary; and gained considerable experience with Amazon's Mechanical Turk. He was also member of the organizing committees for the Workshops for Machine Translation (WMT) in 2010 and 2011, and the North East Student Colloquium on Artificial Intelligence (NESCAI) in 2010.

Following his time at Hopkins, Omar joined the machine translation group at Microsoft Research (Redmond, WA) as a software engineer.