Extending Meteor With POS-based Word Classes

Andrew Wilkinson
July 20, 2015

1 Introduction

In this project, I modify the Meteor machine translation (MT) evaluation metric by replacing its distinction between content and function words with a more fine-grained distinction between part-of-speech categories. The hypothesis is that since the content/function word dichotomy is rather coarse, other ways of defining and weighting word classes may yield better overall results for the metric. I adapt the Meteor source code by integrating a POS (part-of-speech) tagger, classifying words into POS categories, and changing the way in which scores are calculated. I then learn parameter weights that optimize Meteor performance on training data, as measured by correlation with human judgments of rank. Finally, I present results that show an improvement in performance on test data compared to output from the default configuration.

2 Methodology

The latest version of Meteor (1.5) [1] has four tunable parameters α, β, γ, and δ that, respectively, govern the relative importance of precision and recall, determine the relationship between chunks and the penalty, set a maximum value for the penalty, and govern the relative importance of content and function words. This distinction between content and function words is central to the current Meteor equations, yet is a simple and somewhat arbitrary way of conceptualizing and quantifying a linguistic concept that is important to MT and to NLP in general.

In place of this, I use the Stanford Part-Of-Speech tagger (version 3.5.2) [2] as part of the Meteor project structure to tag words in the hypothesis translations and reference sentences. (Any POS tagger could theoretically be used. The Stanford tagger was relatively simple to implement, being written in Java as is Meteor; and it performs near the top in accuracy rankings.) This tagger outputs a total of 36 POS tags, corresponding to those used in the Penn Treebank project [3]. This is too numerous a set to be of use for the current purpose. An intermediate level of representation was needed that is more fine-grained than "content/function" but broader than 36 categories. I chose to define four broad POS categories, with each tag belonging to one and only one of them (Table 2.1).

Table 2.1: Initial POS categories—System 1

Adjective/Adverb		Noun		Verb		Other	
JJ	Adjective	NN	Noun, singular or mass	VB	Verb, base form	CC	Coordinating conj.
JJR	Adjective, comparative	NNS	Noun, plural	VBD	Verb, past tense	CD	Cardinal number
JJS	Adjective, superlative	NNP	Proper noun, singular	VBG	Verb, gerund or pres. part.	DT	Determiner
RB	Adverb	NNPS	Proper noun, plural	VBN	Verb, past part.	EX	Existential *there*
RBR	Adverb, comparative	PRP	Personal pronoun	VBP	Verb, non-3p sing. pres.	FW	Foreign word
RBS	Adverb, superlative	PRP$	Possessive pronoun	VBZ	Verb, 3p sing. pres.	IN	Preposition or sub. conj.
		WP	*Wh*-pronoun			LS	List item marker
		WP$	Possessive *wh*-pronoun			MD	Modal
		WRB	*Wh*-adverb			PDT	Predeterminer
						POS	Possessive ending
						RP	Particle
						SYM	Symbol
						TO	*to*
						UH	Interjection
						WDT	*Wh*-determiner

Deciding upon these categories was based firstly on linguistic intuition. Nouns, verbs, adjectives, and adverbs are traditional delineations of grammar cross-linguistically, and the clearer the lines are between those categories, the more such a categorization will "reward" an MT system that does well translating members of each class of words. On the other hand, items like "predeterminer" and "particle" are less well-defined across grammars, less easily correctly identifiable by a POS tagger, more likely to be translated periphrastically by an MT system (if at all), and less important to comprehension and judgments of fluency (many function words belong to the tags classified in "Other"). Secondly, I wanted to keep the space of possible reasonably-defined categories under control. Since the number of possible categorizations of tags (even ones that make intuitive sense) gets very large very fast, it made sense to limit the set size to four and to be guided in fine-tuning the sets by what the data would show to be most promising.

In fitting this framework into the existing Meteor scoring metric, each POS category n, as above, is weighted by a new δ_n parameter, which replaces the δ and $(1-\delta)$ weights currently assigned to the content and function word categories, respectively. So, replacing the current precision and recall equations

$$P = \frac{\sum_i w_i \cdot (\delta \cdot m_i(h_c) + (1-\delta) \cdot m_i(h_f))}{\delta \cdot |h_c| + (1-\delta) \cdot |h_f|}, R = \frac{\sum_i w_i \cdot (\delta \cdot m_i(r_c) + (1-\delta) \cdot m_i(r_f))}{\delta \cdot |r_c| + (1-\delta) \cdot |r_f|}$$

are new equations given by

$$P = \frac{\sum_i w_i \cdot (\sum_n (\delta_n \cdot m_i(h_n)))}{\sum_n (\delta_n \cdot |h_n|)}, R = \frac{\sum_i w_i \cdot (\sum_n (\delta_n \cdot m_i(r_n)))}{\sum_n (\delta_n \cdot |r_n|)}$$

where words of each POS category are identified in the hypothesis translation (h_n) and reference sentence (r_n), weights w_i for the four "matcher" modules are applied to each category as before, $0 \leqslant \delta_n \leqslant 1 \ \forall \ \delta_n$, and $\sum_n \delta_n = 1$.

Implementing the new scoring protocol in the Meteor source code required adding a `Tagger` class to tag words, adding to and adapting existing methods in many classes to keep track of which words in a sentence belong to which POS category and fall under which matcher module, collecting new statistics, and using those statistics to calculate scores in a new way.

In order to learn and tune weights for the new delta parameters, I used the procedure that has been used in the past to tune the other parameters and the matcher weights. I obtained the Workshop on Statistical Machine Translation (WMT) data from the years 2012, 2013, and 2014, which include MT system submissions for a variety of language pairs, source and reference sentences, and human rankings of the relative quality of system outputs for individual source sentences from each language pair [4] . Using the provided `unroll_wmt_ranks.py` and `sgmlize.py` scripts, I extracted the relevant pairwise rankings from the datasets from each year and rendered the reference sentences and hypothesis translations in the format used by the `Trainer` class. (According to Denkowski & Lavie [5], the data from 2009 and 2010 were used previously in tuning; however, the files from those years were in a different format from the later years and it would have been too time-consuming to adapt them for my purposes.) I combined the 2013 and 2014 datasets to use as a training set, and used the 2012 dataset as a test set; details are given in Table 2.2.

Table 2.2: Datasets

	Source languages	MT systems	Total systems		Rankings	Total rankings	
2012	CS	6	49		11,021	44,345	
	DE	16			11,934		
	ES	12			9,796		
	FR	15			11,594		
2013	CS	12	94	142	85,469	514,132	615,235
	DE	23			128,668		
	ES	17			67,832		
	FR	19			80,741		
	RU	23			151,422		
2014	CS	5	48		15,732	101,103	
	DE	13			18,310		
	FR	8			18,603		
	HI	9			19,620		
	RU	13			28,838		

After appropriate adaptation, the `Trainer` class was then used on each dataset to iterate over all the possible configurations of the new delta parameters by increments of 0.05, and for each configuration used, output the correlation over that dataset with the human rankings. The correlation is measured using the "Kendall's tau" metric, where $\tau = \frac{C-(T-C)}{T}$. In this equation, C = the number of pairwise rankings that Meteor scored correctly. (For Meteor to be "correct" on a pairwise ranking, the translation given the higher score of the pair is also the translation rated as the better of the pair by the human.) T = the total number of pairwise rankings. A list of scores with corresponding new delta parameter weights, upon sorting from greatest to least, yields the optimal parameter configuration for the dataset and the Meteor score achieved with that configuration. This number can then be compared to the result obtained using standard Meteor with the default configuration; a score higher than the default indicates that this method is successful in yielding better correlation with human judgments.

3 Results

For all experiments I conducted, I kept the α, β, and γ parameters at their default values of 0.85, 0.2, and 0.6 respectively. The original δ parameter, which was replaced in my systems with the δ_n parameters, was always kept at its default value of 0.75 for the default system. I used the `-norm` argument throughout to tokenize and lowercase the input. I initially kept the matcher module weights at their default values, and later experimented with optimizing them, as will be seen. I first produced the optimized correlation using the combined 2013-2014 training set, with the POS categories defined as in Table 2.1. I call this configuration of POS categories System 1. Table 3.1 shows my baseline results.

Table 3.1: Training Set Results—System 1

		Default				System 1			
Matcher weights	Module number	1	2	3	4	1	2	3	4
	Value	1.0	0.6	0.8	0.6	1.0	0.6	0.8	0.6
New delta parameters	POS category					Adj/Adv	Noun	Other	Verb
	Value					0.3	0.35	0.1	0.25
Kendall's τ		0.23539				**0.24024**			
Correct rankings		380,028				**381,518**			
Percent correct rankings		61.77%				**62.01%**			

This instantiation of Meteor correctly ranks 1,490 more of the sentence pairs than the default version does. This is an increase of 0.392% over the default, and an increase of 0.242% of the total pairwise rankings—a modest success. I decided to define a few more configurations of POS categories to see if I could improve performance further. System 2 removed the tag MD from "Other" and added it to "Verbs"; System 3 did the same as System 2 and also removed the tags {PRP, PRP$, WP, WP$, WRB} from "Nouns" and

added them to "Other"; System 4 added the tags {PRP, PRP$, WP, WP$, WRB} to "Other" and removed all other tags except for {PDT, POS, WDT} from "Other." Table 3.2 shows the results.

Table 3.2: Training Set Results—Systems 2, 3, 4

		Default				System 2			
Matcher weights	Module number	1	2	3	4	1	2	3	4
	Value	1.0	0.6	0.8	0.6	1.0	0.6	0.8	0.6
New delta parameters	POS category					Adj/Adv	Noun	Other	Verb
	Value					0.3	0.35	0.1	0.25
Kendall's τ		0.23539				**0.24069**			
Correct rankings		380,028				**381,657**			
Percent correct rankings		61.77%				**62.03%**			

		System 3				System 4			
Matcher weights	Module number	1	2	3	4	1	2	3	4
	Value	1.0	0.6	0.8	0.6	1.0	0.6	0.8	0.6
New delta parameters	POS category	Adj/Adv	Noun	Other	Verb	Adj/Adv	Noun	Other	Verb
	Value	0.25	0.35	0.15	0.25	0.3	0.35	0.15	0.2
Kendall's τ		0.23771				0.22457			
Correct rankings		380,741				376,699			
Percent correct rankings		61.89%				61.23%			

System 2 yielded the best results. System 3 surpassed the default, but did not do as well as System 2; System 4 did not do as well as the default.

Thus far, the only difference in weights between my experimental systems and the default was the replacement of delta by the new delta parameters. For my next experiment, I kept the new delta parameters at the optimal values found by System 2, kept the change in POS categories (MD being added to "Verbs"), and iterated over the weights for matcher modules 2, 3, and 4. Table 3.3 shows the resulting System 5.

Table 3.3: Training Set Results—System 5

		Default				System 5			
Matcher weights	Module number	1	2	3	4	1	2	3	4
	Value	1.0	0.6	0.8	0.6	1.0	0.7	0.6	0.6
New delta parameters	POS category					Adj/Adv	Noun	Other	Verb
	Value					0.3	0.35	0.1	0.25
Kendall's τ		0.23539				**0.24069**			
Correct rankings		380,028				**381,962**			
Percent correct rankings		61.77%				**62.08%**			

Finally, I conducted experiments to see whether the values obtained from optimizing on the training set would also outperform the default on a different dataset. Table

Table 3.4: Test Set Results

		Default matcher weights				System 5 matcher weights			
Matcher weights	Module number	1	2	3	4	1	2	3	4
	Value	1.0	0.6	0.8	0.6	1.0	0.7	0.6	0.6
Kendall's τ		0.24731				0.24884			
Correct rankings		27,656				27,690			
Percent correct rankings		62.37%				62.44%			

		Default matcher weights				System 5 matcher weights			
Matcher weights	Module number	1	2	3	4	1	2	3	4
	Value	1.0	0.6	0.8	0.6	1.0	0.7	0.6	0.6
New delta parameters	POS category	Adj/Adv	Noun	Other	Verb	Adj/Adv	Noun	Other	Verb
—from training set	Value	0.3	0.35	0.1	0.25	0.3	0.35	0.10	0.25
Kendall's τ		*0.25710*				*0.25669*			
Correct rankings		*27,873*				*27,864*			
Percent correct rankings		*62.85%*				*62.83%*			

		Default matcher weights				System 5 matcher weights			
Matcher weights	Module number	1	2	3	4	1	2	3	4
	Value	1.0	0.6	0.8	0.6	1.0	0.7	0.6	0.6
New delta parameters	POS category	Adj/Adv	Noun	Other	Verb	Adj/Adv	Noun	Other	Verb
—optimized on test set	Value	0.25	0.4	0.05	0.3	0.25	0.4	0.05	0.3
Kendall's τ		**0.26030**				**0.26039**			
Correct rankings		**27,944**				**27,946**			
Percent correct rankings		**63.01%**				**63.02%**			

3.4 shows the results. Naturally, when I optimized the new delta parameters and the matcher weights on the test set, that yielded the best results (boldfaced cells). However, using the new delta parameters obtained from the training set (italicized cells) also outperforms the default configuration. It is unclear if optimizing the matcher weights made a significant difference across the trials.

4 Conclusion and Potential Future Work

After altering the code for Meteor to take part-of-speech categories into consideration when scoring candidate translations and calculating the correlations with human judgments of MT system quality that result, I am pleased to observe that this method gives consistent improvement over scoring using the content/function word distinction on both the training set and the test set. The amount of improvement seems small, measured in tenths of a percent of the total number of pairwise judgments in a dataset. I do not, however, have the experience to judge the utility of this level of improvement in terms of commercial or research application. It seems to me that no MT evaluation metric could hope to recreate human judgments with anything approaching 100% accuracy, because human judgments are subjective, fickle, and often contradictory. In any case, I have by no means exhausted this course of inquiry. There are a number of ways in which I could

see this work being improved upon.

First, the brute-force approach of searching over the whole parameter space more completely than I did here would probably yield higher correlations. However, the more variables there are, the more likely overfitting becomes. It is for this reason that I began by keeping all the old values at their previously-defined optima. I also think it unlikely that there exists a significant dependent relationship between all the variables; for some it is no doubt higher than for others. I chose to search over matcher weights because I felt that the types of matches they represent (stemmed, synonym, paraphrase) might have more interrelation with POS categories than the α, β and γ parameters do, yet the change from the default was modest. So this might represent only a limited source of reproducible further improvements.

Second, a scoring metric based on a POS tagger can only be as good as the tagger itself. In turn, the tagger will not perform as well on grammatically messy, disfluent input—the type of input that characterizes many, if not most, MT output sentences—as it will on fluent language (and even that yields plenty of errors). There is a lot of noise in this domain that prevents other NLP applications from functioning as well as they might. Perhaps there could be a way of leveraging the aligner, or using machine learning, to pass more-likely-to-be-correct tags from a reference to a hypothesis sentence. For instance, perhaps the tagger correctly identifies the word "fall" in the reference as a verb, but incorrectly in the hypothesis as a noun. If incorrect tags could be "overridden" and tagger accuracy improved for the system outputs, that might benefit the end result of this approach.

Third, there are a great many possible configurations of POS categories using the PTB tag set that I have not explored. Perhaps a five-way or three-way distinction would be better; maybe adverbs and adjectives do not belong together. Maybe pronouns should have their own category. My experiments showed that it is an easy thing to describe configurations that underperform the default. This lends credence to the underlying assumption that grammatical categories can be salient features for MT purposes in the first place. If it didn't matter which tags were grouped with which others, we would see little gain or drop in the results, and the values learned might end up about equal. Yet there is a trend in the results that is clearly visible when viewing the top 10 or 20 parameter configurations compared to the bottom tier: Nouns (as I defined them) are the most important—they receive the highest weight overall—followed by the adjective/adverb class and verb class at approximately the same level, followed distantly by "other." I could see results like these being used to more usefully prioritize and deprioritize accuracy in MT systems along grammatical-category lines.

References

[1] `https://www.cs.cmu.edu/~alavie/METEOR/pdf/meteor-1.5.pdf`

[2] `http://nlp.stanford.edu/software/tagger.shtml`

[3] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[4] http://www.statmt.org

[5] Michael Denkowski and Alon Lavie, "Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems", Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation, 2011.