HyTER: Meaning-Equivalent Semantics for Translation Evaluation

Markus Dreyer
SDL Language Weaver
6060 Center Drive, Suite 150
Los Angeles, CA 90045, USA
mdreyer@sdl.com

Daniel Marcu
SDL Language Weaver
6060 Center Drive, Suite 150
Los Angeles, CA 90045, USA
dmarcu@sdl.com

Abstract

It is common knowledge that translation is an ambiguous, 1-to-n mapping process, but to date, our community has produced no empirical estimates of this ambiguity. We have developed an annotation tool that enables us to create representations that compactly encode an exponential number of correct translations for a sentence. Our findings show that naturally occurring sentences have billions of translations. Having access to such large sets of meaning-equivalent translations enables us to develop a new metric, HyTER, for translation accuracy. We show that our metric provides better estimates of machine and human translation accuracy than alternative evaluation metrics.

1 Motivation

During the last decade, automatic evaluation metrics (Papineni et al., 2002; Snover et al., 2006; Lavie and Denkowski, 2009) have helped researchers accelerate the pace at which they improve machine translation (MT) systems. And human-assisted metrics (Snover et al., 2006) have enabled and supported large-scale U.S. government sponsored programs, such as DARPA GALE (Olive et al., 2011). However, these metrics have started to show signs of wear and tear.

Automatic metrics are often criticized for providing non-intuitive scores – few researchers can explain to casual users what a BLEU score of 27.9 means. And researchers have grown increasingly concerned that automatic metrics have a strong bias towards preferring statistical translation outputs; the NIST (2008, 2010), MATR (Gao et al., 2010) and WMT (Callison-Burch et al., 2011) evaluations held during the last five years have provided ample evidence that automatic metrics yield results that are inconsistent with human evaluations when comparing statistical, rule-based, and human outputs.

In contrast, human-informed metrics have other deficiencies: they have large variance across human judges (Bojar et al., 2011) and produce unstable results from one evaluation to another (Przybocki et al., 2011). Because evaluation scores are not computed automatically, systems developers cannot automatically tune to human-based metrics.

Table 1 summarizes the dimensions along which evaluation metrics should do well and the strengths and weaknesses of the automatic and human-informed metrics proposed to date. Our goal is to develop metrics that do well along all these dimensions. The fundamental insight on which our research relies is that the failures of current automatic metrics are not algorithmic: BLEU, Meteor, TER (Translation Edit Rate), and other metrics efficiently and correctly compute informative distance functions between a translation and one or more human references. We believe that these metrics fail simply because they have access to sets of human references that are too small. If we had access to the set of *all* correct translations of a given sentence, we could measure the minimum distance between a translation and the set. When a translation is perfect, it can be found in the set, so it requires no editing to produce a perfect translation. Therefore, its score should be zero. If the translation has errors, we can

2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 162–171,
Montréal, Canada, June 3-8, 2012.

Desiderata	Auto.	Manu.	HyTER
Metric is intuitive	N	Y	Y
Metric is computed automatically	Y	N	Y
Metric is stable and reproducible from one evaluation to another	Y	N	Y
Metric works equally well when comparing human and automatic outputs and when comparing rule-based, statistical-based, and hybrid engines	N	Y	Y
System developers can tune to the metric	Y	N	Y
Metric helps developers identify deficiencies of MT engines	N	N	Y

Table 1: Desiderata of evaluation metrics: Current automatic and human metrics, proposed metric.

efficiently compute the minimum number of edits (substitutions, deletions, insertions, moves) needed to rewrite the translation into the "closest" reference in the set. Current automatic evaluation metrics do not assign their best scores to most perfect translations because the set of references they use is too small; their scores can therefore be perceived as less intuitive.

Following these considerations, we developed an annotation tool that enables one to efficiently create an exponential number of correct translations for a given sentence, and present a new evaluation metric, HyTER, which efficiently exploits these massive reference networks. In the rest of the paper, we first describe our annotation environment, process, and meaning-equivalent representations that we create (Section 2). We then present the HyTER metric (Section 3). We show that this new metric provides better support than current metrics for machine translation evaluation (Section 4) and human translation proficiency assessment (Section 5).

2 Annotating sentences with exponential numbers of meaning equivalents

2.1 Annotation tool

We have developed a web-based annotation tool that can be used to create a representation encoding an exponential number of meaning equivalents for a given sentence. The meaning equivalents are constructed in a bottom-up fashion by typing translation equivalents for larger and larger phrases. For example, when building the meaning equivalents for the Spanish phrase "el primer ministro italiano Silvio Berlusconi", the annotator first types in the meaning equivalents for "primer ministro" – ⟨prime-minister; PM; prime minister; head of government; premier; etc.⟩; "italiano" – ⟨Italian⟩; and "Silvio Berlusconi" – ⟨Silvio Berlusconi; Berlusconi⟩. The tool creates a *card* that stores all the alternative meanings for a phrase as a determinized FSA and gives it a name in the target language that is representative of the underlying meaning-equivalent set: [PRIME-MINISTER], [ITALIAN], and [SILVIO-BERLUSCONI]. Each base card can be thought of expressing a semantic concept. A combination of existing cards and additional words can be subsequently used to create larger meaning equivalents that cover increasingly larger source sentence segments. For example, to create the meaning equivalents for "el primer ministro italiano" one can drag-and-drop existing cards or type in new words: ⟨the [ITALIAN] [PRIME-MINISTER]; the [PRIME-MINISTER] of Italy⟩; to create the meaning equivalents for "el primer ministro italiano Silvio Berlusconi", one can drag-and-drop and type: ⟨[SILVIO-BERLUSCONI] , [THE-ITALIAN-PRIME-MINISTER]; [THE-ITALIAN-PRIME-MINISTER] , [SILVIO-BERLUSCONI]; [THE-ITALIAN-PRIME-MINISTER] [SILVIO-BERLUSCONI] ⟩. All meaning equivalents associated with a given card are expanded and used when that card is re-used to create larger meaning-equivalent sets.

The annotation tool supports, but does not enforce, re-use of annotations created by other annotators. The resulting meaning equivalents are stored as recursive transition networks (RTNs), where each card is a subnetwork; if needed, these non-cyclic RTNs can be automatically expanded into finite-state acceptors (FSAs, see Section 3).

2.2 Data and Annotation Protocols

Using the annotation tool, we have created meaning-equivalent annotations for 102 Arabic and 102 Chinese sentences – a subset of the "progress set" used in the 2010 Open MT NIST evaluation (the average

sentence length was 24 words). For each sentence, we had access to four human reference translations produced by LDC and five MT system outputs, which were selected by NIST to cover a variety of system architectures (statistical, rule-based, hybrid) and performances. For each MT output, we also had access to sentence-level HTER scores (Snover et al., 2006), which were produced by experienced LDC annotators.

We have experimented with three annotation protocols:

- Ara-A2E and Chi-C2E: Foreign language natives built English networks starting from foreign language sentences.
- Eng-A2E and Eng-C2E: English natives built English networks starting from "the best translation" of a foreign language sentence, as identified by NIST.
- Eng*-A2E and Eng*-C2E: English natives built English networks starting from "the best translation", but had access to three additional, independently produced human translations to boost their creativity.

Each protocol was implemented independently by at least three annotators. In general, annotators need to be fluent in the target language, familiar with the annotation tool we provide and careful not to generate incorrect paths, but they do not need to be linguists.

2.3 Exploiting multiple annotations

For each sentence, we combine all networks that were created by the different annotators. We evaluate two different combination methods, each of which combines networks N_1 and N_2 of two annotators (see an example in Figure 1):

(a) Standard union U(N_1, N_2): The standard finite-state union operation combines N_1 and N_2 on the whole-network level. When traversing U(N_1, N_2), one can follow a path that comes from either N_1 or N_2.

(b) Source-phrase-level union SPU(N_1, N_2): As an alternative, we introduce SPU, a more fine-grained union which operates on sub-sentence segments. Here we exploit the fact that each annotator explicitly aligned each of her various subnetworks

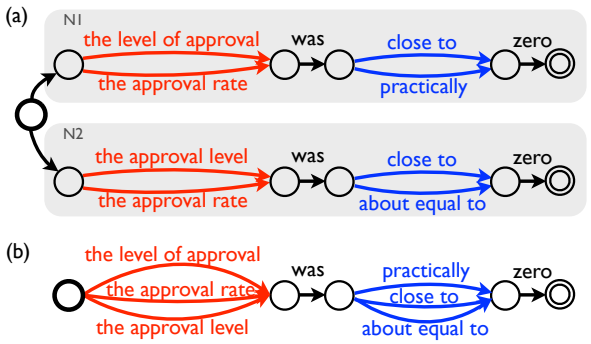

Figure 1: (a) Finite-state union versus (b) source-phrase-level union (SPU). The former does not contain the path "the approval level was practically zero".

for a given sentence to a source span of that sentence. Now for each pair of subnetworks (S_1, S_2) from N_1 and N_2, we build their union if they are *compatible*; two subnetworks S_1, S_2 are defined to be compatible if they are aligned to the same source span and have at least one path in common.

The purpose of SPU is to create new paths by mixing paths from N_1 and N_2. In Figure 1, for example, the path "the approval level was practically zero" is contained in the SPU, but not in the standard union. We build SPUs using a dynamic programming algorithm that builds subnetworks bottom-up, building unions of intermediate results. Two larger subnetworks can be compatible only if their recursive smaller subnetworks are compatible. Each SPU contains at least all paths from the standard union.

2.4 Empirical findings

Now that we have described how we created particular networks for a given dataset, we describe some empirical findings that characterize our annotation process and the created networks.

Meaning-equivalent productivity. When we compare the productivity of the three annotation protocols in terms of the number of reference translations that they enable, we observe that target language natives that have access to multiple human references produce the largest networks. The median number of paths produced by one annotator under the three protocols varies from 7.7×10^5 paths for Ara-A2E, to 1.4×10^8 paths for Eng-A2E, to 5.9×10^8 paths for Eng*-A2E; in Chinese, the me-

dians vary from 1.0×10^5 for Chi-C2E, to 1.7×10^8 for Eng-C2E, to 7.8×10^9 for Eng*-C2E.

Protocol productivity. When we compare the annotator time required by the three protocols, we find that foreign language natives work faster – they need about 2 hours per sentence – while target language natives need 2.5 hours per sentence. Given that target language natives build significantly larger networks and that bilingual speakers are in shorter supply than monolingual ones, we conclude that using target language annotators is more cost-effective overall.

Exploiting multiple annotations. Overall, the median number of paths produced by a single annotator for A2E is 1.5×10^6, two annotators (randomly picked per sentence) produce a median number of 4.7×10^7 (Union), for all annotators together it is 2.1×10^{10} (Union) and 2.1×10^{11} (SPU). For C2E, these numbers are 5.2×10^6 (one), 1.1×10^8 (two), and 2.6×10^{10} (all, Union) and 8.5×10^{11} (all, SPU).

Number of annotators and annotation time. We compute the minimum number of edits and length-normalized distance scores required to rewrite machine and human translations into translations found in the networks produced by one, two, and three annotators. We find that the length-normalized distances do not vary by more than 1% when adding the meaning equivalents produced by a third annotator. We conclude that 2-3 annotators per sentence produce a sufficiently large set of alternative meaning equivalents, which takes 4-7.5 hours. We are currently investigating alternative ways to create networks more efficiently.

Grammaticality. For each of the four human translation references and each of the five machine translation outputs (see Section 2.2), we algorithmically find the closest path in the annotated networks of meaning equivalents (see Section 3). We presented the resulting 1836 closest paths extracted from the networks (2 language pairs $\times 102$ sentences $\times 9$ human/machine translations) to three independent English speakers. We asked each English path to be labeled as grammatical, grammatical-but-slightly-odd, or non-grammatical. The metric is harsh: paths such as "he said that withdrawing *US force* without promoting security would be cataclysmic" are judged as non-grammatical by all three judges although a simple rewrite of "force" into "forces" would make this path grammatical. We found that 90% of the paths are judged as grammatical and 96% as grammatical or grammatical-but-slightly-odd, by at least one annotator. We interpret these results as positive: the annotation process leads to some ungrammatical paths being created, but most of the closest paths to human and machine outputs, those that matter from an evaluation perspective, are judged as correct by at least one judge.

Coverage. We found it somewhat disappointing that networks that encode billions of meaning-equivalent translations for a given sentence do not contain every independently produced human reference translation. The average length-normalized edit distance (computed as described in Section 3) between an independently produced human reference and the corresponding network is 19% for Arabic-English and 34% for Chinese-English across the entire corpus. Our analysis shows that about half of the edits are explained by several non-content words ("the", "of", "for", "their", ",") being optional in certain contexts; several "obvious" equivalents not being part of the networks ("that"–"this"; "so"–"accordingly"); and spelling alternatives/errors ("rockstrom"–"rockstroem"). We hypothesize that most of these ommissions/edits can be detected automatically and dealt with in an appropriate fashion. The rest of the edits would require more sophisticated machinery, to figure out, for example, that in a particular context pairs like "with"–"and" or "that"–"therefore" are interchangeable.

Given that Chinese is significantly more underspecified compared to Arabic and English, it is consistent with our intuition to see that the average minimal distance is higher between Chinese-English references and their respective networks (34%) than between Arabic-English references and their respective networks (19%).

3 Measuring translation quality with large networks of meaning equivalents

In this section, we present HyTER (Hybrid Translation Edit Rate), a novel metric that makes use of large reference networks.

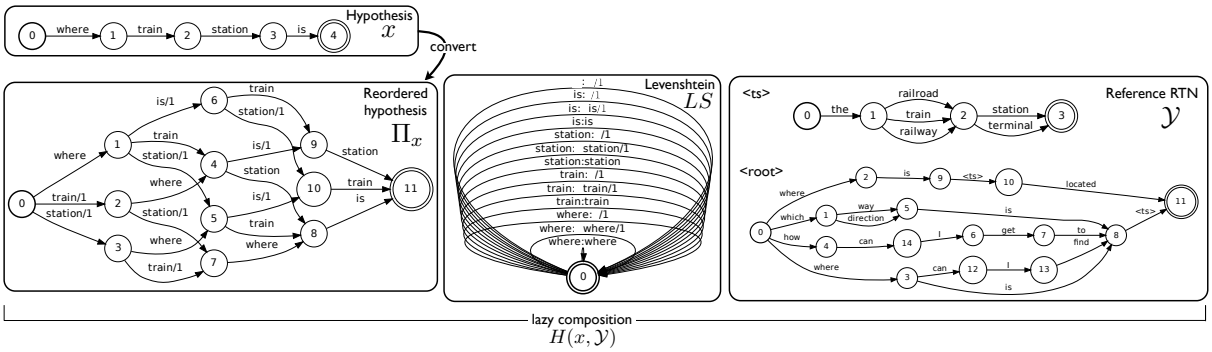

Figure 2: Defining the search space $H(x, \mathcal{Y})$ through (lazy) composition. x is a translation hypothesis *"where train station is"*, $\mathcal{Y}$ contains all correct translations. Π_x may be defined in various ways, here local reordering ($k = 3$) is used.

HyTER is an automatically computed version of HTER (Snover et al., 2006); HyTER computes the minimum number of edits between a translation x and an exponentially sized reference set $\mathcal{Y}$, which is encoded as a Recursive Transition Network (RTN). Perfect translations have a HyTER score of 0.

General Setup. The unnormalized HyTER score is defined as in equation (1) where Π_x is a set of permutations of the hypothesis x, $d(x, x')$ is the distance between x and a permutation x'—typically measured as the number of reordering moves between the two—and $\mathrm{LS}(\mathrm{x}', \mathrm{y})$ is the standard Levenshtein distance (Levenshtein, 1966) between x' and y, defined as the minimum number of insertions, deletion, and substitutions. We normalize uhyter by the number of words in the found closest path.

$$\mathrm{uhyter}(x, \mathcal{Y}) \stackrel{\text{def}}{=} \min_{\substack{x' \in \Pi_x, \\ y \in \mathcal{Y}}} d(x, x') + \mathrm{LS}(x', y) \quad (1)$$

We treat this minimization problem as graph-based search. The search space over which we minimize is implicitly represented as the Recursive Transition Network H (see equation (2)), where Π_x is encoded as a weighted FSA that represents the set of permutations of x with their associated distance costs, and LS is the one-state Levenshtein transducer whose output weight for a string pair (x,y) is the Levenshtein distance between x, and y, and the symbol $\circ$ denotes composition. The model is depicted in Figure 2.

$$H(x, \mathcal{Y}) \stackrel{\text{def}}{=} \Pi_x \circ LS \circ \mathcal{Y} \quad (2)$$

Permutations. We define an FSA Π_x that allows permutations according to certain constraints. Allowing *all* permutations of the hypothesis x would increase the search space to factorial size and make inference NP-complete (Cormode and Muthukrishnan, 2007). We use local-window constraints (see, e.g., Kanthak et al. (2005)), where words may move within a fixed window of size k; these constraints are of size $O(n)$ with a constant factor k, where n is the length of the translation hypothesis x.

Lazy Evaluation. For efficiency, we use lazy evaluation when defining the search space $H(x, \mathcal{Y})$. This means we never explicitly compose Π_x, LS, and $\mathcal{Y}$. Parts of the composition that our inference algorithm does not explore are not constructed, saving computation time and memory. Permutation paths in Π_x are constructed on demand. Similarly, the reference set $\mathcal{Y}$ is expanded on demand, and large parts may remain unexpanded.[1]

Exact Inference. To compute $\mathrm{uhyter}(x, \mathcal{Y})$, we define the composition $H(x, \mathcal{Y})$ and can apply any shortest-path search algorithm (Mohri, 2002). We found that using the A* algorithm (Hart et al., 1972) was the most efficient; we devised an A* heuristic similar to Karakos et al. (2008).

Runtime. Computing the HyTER score takes 30 ms per sentence on networks by single annotators (combined all-annotator networks: 285 ms) if no

[1] These on-demand operations are supported by the OpenFst library (Allauzen et al., 2007); specifically, to expand the RTNs into FSAs we use the `Replace` operation.

Metric	Arabic-English Human mean	Machine mean	m/h	Chinese-English Human mean	Machine mean	m/h
[100-0]-BLEU, 1 ref	59.90	69.14	1.15	71.86	84.34	1.17
[100-0]-BLEU, 3 refs	41.49	57.44	1.38	54.25	75.22	1.39
[100-0]-Meteor, 1 ref	60.13	65.70	1.09	66.81	73.66	1.10
[100-0]-Meteor, 3 refs	55.98	62.91	1.12	62.95	70.68	1.12
[100-0]-TERp, 1 ref	35.87	46.48	1.30	53.58	71.70	1.34
[100-0]-TERp, 3 refs	27.08	39.52	1.46	41.79	60.61	1.45
HyTER U	18.42	34.94	1.90	27.98	52.08	1.86
HyTER SPU	17.85	34.39	1.93	27.57	51.73	1.88
[100-0]-Likert	5.26	50.37	9.57	4.35	48.37	11.12

Table 2: Scores assigned to human versus machine translations, under various metrics. Each score is normalized to range from 100 (worst) to 0 (perfect translation).

reordering is used. These numbers increase to 143 ms (1.5 secs) for local reordering with window size 3, and 533 ms (8 secs) for window size 5. Many speedups for computing the score with reorderings are possible, but we will see below that using reordering does not give consistent improvements (Table 3).

Output. As a by-product of computing the HyTER score, one can obtain the closest path itself, for error analysis. It can be useful to separately count the numbers of insertions, deletions, etc., and inspect the types of error. For example, one may find that a particular system output tends to be missing the finite verb of the sentence or that certain word choices were incorrect.

4 Using meaning-equivalent networks for machine translation evaluation

We now present experiments designed to measure how well HyTER performs, compared to other evaluation metrics. For these experiments, we sample 82 of the 102 available sentences; 20 sentences are held out for future use in optimizing our metric.

4.1 Differentiating human from machine translation outputs

We score the set of human translations and machine translations separately, using several popular metrics, with the goal of determining which metric performs better at separating machine translations from human translations. To ease comparisons across different metrics, we normalize all scores to a number between 0 (best) and 100 (worst). Table 2 shows the normalized mean scores for the machine translations and human translations under multiple automatic and one human evaluation metric (Likert). The quotient of interest, m/h, is the mean score for machine translations divided by the mean score for the human translations: the higher this number, the better a metric separates machine from human produced outputs.

Under HyTER, m/h is about 1.9, which shows that the HyTER scores for machine translations are, on average, almost twice as high as for human translations. Under Likert – a score assigned by human annotators who compare pairs of sentences at a time–, the quotient is higher, suggesting that human raters make stronger distinctions between human and machine translations. The quotient is lower under the automatic metrics Meteor (Version 1.3, (Denkowski and Lavie, 2011)), BLEU and TERp (Snover et al., 2009). These results show that HyTER separates machine from human translations better than alternative metrics.

4.2 Ranking MT systems by quality

We rank the five machine translation systems according to several widely used metrics (see Figure 3). Our results show that BLEU, Meteor and TERp do not rank the systems in the same way as HTER and humans do, while the HyTER metric yields the correct ranking. Also, separation between the quality of the five systems is higher under HyTER, HTER, and Likert than under alternative metrics.

4.3 Correlations with HTER

We know that current metrics (e.g., BLEU, Meteor, TER) correlate well with HTER and human judg-

Arabic-English									
Size	Likert	Meteor 1	Meteor 4	BLEU 1	BLEU 4	TERp 1	TERp 4	HyTER U (r5)	HyTER SPU (r5)
1	.653	.529	.541	.512	.675	.452	.547	.643 (.661)	.647 (.655)
2	.645	.614	.636	.544	.706	.599	.649	.733 (.741)	.735 (.732)
4	.739	.782	.804	.710	.803	.782	.803	.827 (.840)	.831 (.838)
8	.741	.809	.822	.757	.818	.796	.833	.827 (.828)	.830 (.825)
16	.868	.840	.885	.815	.887	.824	.862	.888 (.890)	.893 (.894)
32	.938	.945	.957	.920	.948	.930	.947	.938 (.935)	.940 (.936)
64	.970	.973	.979	.964	.973	.966	.968	.964 (.960)	.966 (.961)
Chinese-English									
Size	Likert	Meteor 1	Meteor 4	BLEU 1	BLEU 4	TERp 1	TERp 4	HyTER U (r5)	HyTER SPU (r5)
1	.713	.495	.557	.464	.608	.569	.594	.708 (.721)	.668 (.681)
2	.706	.623	.673	.569	.655	.639	.651	.713 (.716)	.702 (.701)
4	.800	.628	.750	.593	.734	.651	.726	.822 (.825)	.820 (.814)
8	.810	.745	.778	.783	.808	.754	.754	.852 (.856)	.854 (.845)
16	.881	.821	.887	.811	.884	.826	.844	.912 (.914)	.914 (.908)
32	.915	.873	.918	.911	.930	.851	.911	.943 (.942)	.941 (.937)
64	.950	.971	.976	.979	.973	.952	.970	.962 (.958)	.958 (.957)

Table 3: Document-level correlations of various scores to HTER. Meteor, BLEU and TERp are shown with 1 and 4 references each, HyTER is shown with the two combination methods (U and SPU), and with reordering (r5).

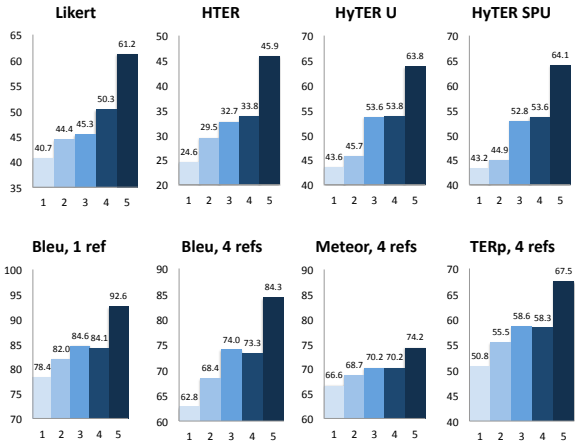

Figure 3: Five MT systems (Chinese-English), scored by 8 different metrics. The x-axis shows the five systems, the y-axis shows the [100-0] normalized scores, with 0 corresponding to a perfect translation. (Note that the scale is similar in all eight graphs.) HTER and HyTER show a similar pattern and similar ranking of the systems.

ments on large test corpora (Papineni et al., 2002; Snover et al., 2006; Lavie and Denkowski, 2009). We believe, however, that the field of MT will be better served if researchers have access to metrics that provide high correlation at the sentence level as well. To this end, we estimate how well various metrics correlate with the Human TER (HTER) metric for corpora of increasingly larger sizes.

Table 3 shows Pearson correlations between HTER and various metrics for scoring documents of size s =1, 2, 4, ..., and 64 sentences. To get more reliable results, we create 50 documents per size s, where each document is created by selecting s sentences at random from the available 82 sentences. For each document, there are 5 translations from different systems, so we have 250 translated documents per size. For each language and size, we score the 250 documents under HTER and the other metrics and report the Pearson correlation. Our results show that for large documents, all metrics correlate well with HTER. However, as the sizes of the documents decrease, and especially at the sentence level, HyTER provides especially high correlation with HTER as compared to the other metrics. As a side note, we can see that using reordering when computing the HyTER score does not give consistently better results – see the (r5) numbers, which searched over hypothesis permutations within a local window of size 5; this shows that most reorderings are already captured in the networks. In all experiments, we use BLEU with plus-one smoothing (Lin and Och, 2004).

5 Using meaning-equivalent networks for human translation evaluation

In this section, we present a use case for the HyTER metric outside of machine translation.

5.1 Setup and problem

Language Testing units assess the translation proficiency of thousands of applicants interested in performing language translation work for the US Government. Job candidates typically take a written test in which they are asked to translate four passages (i.e., paragraphs) of increasing difficulty into English. The passages are at difficulty levels 2, 2+, 3, and 4 on the Interagency Language Roundable (ILR) scale.[2] The translations produced by each candidate are manually reviewed to identify mistranslation, word choice, omission, addition, spelling, grammar, register/tone, and meaning distortion errors. Each passage is then assigned one of five labels: Successfully Matches the definition of a successful translation (SM); Mostly Matches the definition (MM); Intermittently Matches (IM); Hardly Matches (HM); Not Translated (NT) for anything where less than 50% of a passage is translated.

We have access to a set of more than 100 rules that agencies practically use to assign each candidate an ILR translation proficiency level: 0, 0+, 1, 1+, 2, 2+, 3, and 3+. For example, a candidate who produces passages labeled as SM, SM, MM, IM for difficulty levels 2, 2+, 3, and 4, respectively, is assigned an ILR level of 2+.

We investigate whether the assessment process described above can be automated. To this end, we obtained the exam results of 195 candidates, where each exam result consists of three passages translated into English by a candidate, as well as the manual rating for each passage translation (i.e., the gold labels SM, MM, IM, HM, or NT). 49 exam results are from a Chinese exam, 71 from a Russian exam and 75 from a Spanish exam. The three passages in each exam are of difficulty levels 2, 2+, and 3; level 4 is not available in our data set. In each exam result, the translations produced by each candidate are sentence-aligned to their respective foreign sentences. We applied the passage-to-ILR mapping rules described above to automatically create a gold overall ILR assessment for each exam submission. Since the languages used here have only 3 passages each, some rules map to several different ILR ratings. Table 4 shows the label distribution at the ILR assessment level across all languages.

Lang.	**0**	**0+**	**1**	**1+**	**2**	**2+**	**3**	**3+**
Chi.	0.0	8.2	40.8	65.3	59.2	10.2	4.1	0.0
Rus.	0.0	2.8	12.7	42.3	60.6	46.5	25.4	5.6
Spa.	0.0	1.3	33.3	66.7	88.0	24.0	4.0	0.0
All	0.0	3.6	27.7	57.4	70.8	28.7	11.8	2.1

Table 4: Percentage of exams with ILR levels 0, 0+, ..., 3+ as gold labels. Multiple levels per exam possible.

5.2 Experiments

We automatically assess the proficiency of candidates who take a translation exam. We treat this as a classification task where, for each translation of the three passages, we predict the three passage assessment labels as well as one overall ILR rating.

In support of our goal, we asked annotators to create an English HyTER network for each foreign sentence in the exams. These HyTER networks then serve as English references for the candidate translations. The median number of paths in these HyTER networks is 1.6×10^6 paths/network.

In training, we observe a set of submitted exam translations, each of which is annotated with three passage-level ratings and one overall ILR rating. We develop features (Section 5.3) that describe each passage translation in its relation to the HyTER networks for the passage. We then train a classifier to predict passage-level ratings given the passage-level features that describe the candidate translation. As classifier, we use a multi-class support-vector machine (SVM, Krammer and Singer (2001)). In decoding, we observe a set of exams without their ratings, derive the features and use the trained SVM to predict ratings of the passage translations. We then derive an overall ILR rating based on the predicted passage-level ratings. Since our dataset is small we run 10-fold cross-validation.

5.3 Features

We define features describing a candidate's translation with respect to the corresponding HyTER reference networks. Each of the feature values is computed based on a passage translation as a whole, rather than sentence-by-sentence. As features, we use the HyTER score, as well as the number of insertions, deletions, substitutions, and insertions-or-deletions. We use these numbers normalized by the

[2] See `http://www.govtilr.org/skills/AdoptedILRTranslationGuidelines.htm`.

Level	Measure	Baseline	HyTER-enabled
All	Accuracy	72.31	90.77
2 or better	Precision	85.62	82.11
	Recall	84.93	98.63
	F_1	85.27	89.62

Table 5: Predicting final ILR ratings for candidate exams.

length of the passage, as well as unnormalized. We also use n-gram precisions (for n=1,...,20) as features.

5.4 Results

We report the accuracy on predicting the overall ILR rating of the 195 exams (Table 5). The results in *2 or better* show how well we predict a performance level of 2, 2+, 3 or 3+. It is important to retrieve such relatively good exams with high recall, so that a manual review QA process can confirm the choices while avoid discarding qualified candidates. The results show that high recall is reached while preserving good precision. Since we have several possible gold labels per exam, precision and recall are computed similar to precision and recall in the NLP task of word alignment. $F_1(P,R) = \frac{2PR}{P+R}$ is the harmonic mean of precision and recall. The row *All* shows the accuracy in predicting ILR performance labels overall. As a baseline method we assign the most frequent label per language; these are 1+ for Chinese, and 2 for Russian and Spanish. The results in Table 5 strongly suggest that the process of assigning a proficiency level to human translators can be automated.

6 Discussion

We have introduced an annotation tool and process that can be used to create meaning-equivalent networks that encode an exponential number of translations for a given sentence. We have shown that these networks can be used as foundation for developing improved machine translation evaluation metrics and automating the evaluation of human translation proficiency. We plan to release the OpenMT HyTER networks to the community after the 2012 NIST Open MT evaluation. We believe that our meaning-equivalent networks can be used to support interesting research programs in semantics, paraphrase generation, natural language understanding, generation, and machine translation.

Acknowledgments

We thank our colleagues and the anonymous reviewers for their valuable feedback. This work was partially sponsored by DARPA contract HR0011-11-C-0150 and TSWG contract N41756-08-C-3020 to Language Weaver Inc.

References

C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. 2007. OpenFst: a general and efficient weighted finite-state transducer library. In *Proceedings of the 12th international conference on Implementation and application of automata*, page 1123.

O. Bojar, M. Ercegovčević, M. Popel, and O. Zaidan. 2011. A grain of salt for the wmt manual evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 1–11, Edinburgh, Scotland, July. Association for Computational Linguistics.

C. Callison-Burch, Ph. Koehn, Ch. Monz, and O. Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, Scotland, July. Association for Computational Linguistics.

G. Cormode and S. Muthukrishnan. 2007. The string edit distance matching problem with moves. *ACM Transactions on Algorithms (TALG)*, 3(1):1–19.

M. Denkowski and A. Lavie. 2011. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation*.

Q. Gao, N. Bach, S. Vogel, B. Chen, G. Foster, R. Kuhn, C. Callison-Burch, P. Koehn, C. Monz, K. Peterson, et al., 2010. *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics (MATR)*, pages 121–126.

P.E. Hart, N.J. Nilsson, and B. Raphael. 1972. Correction to "A formal basis for the heuristic determination of minimum cost paths". *ACM SIGART Bulletin*, pages 28–29, December. ACM ID: 1056779.

S. Kanthak, D. Vilar, E. Matusov, R. Zens, and H. Ney. 2005. Novel reordering approaches in phrase-based statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 167–174.

D. Karakos, J. Eisner, S. Khudanpur, and M. Dreyer. 2008. Machine translation system combination using

ITG-based alignments. *Proc. ACL-08: HLT, Short Papers (Companion Volume)*, page 8184.

K. Krammer and Y. Singer. 2001. On the algorithmic implementation of multi-class SVMs. In *Proc. of JMLR*.

A. Lavie and M. Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115.

L. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710.

C.Y. Lin and F.J. Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, page 501. Association for Computational Linguistics.

M. Mohri. 2002. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350.

J. Olive, C. Christianson, and J. McCary, editors. 2011. *Handbook of Natural Language Processing and Machine Translation*. DARPA Global Autonomous Language Exploitation. Springer.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

M. Przybocki, A. Le, G. Sanders, S. Bronsart, S. Strassel, and M. Glenn, 2011. *GALE Machine Translation Metrology: Definition, Implementation, and Calculation*, chapter 5.4, pages 583–811. Springer.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the Association for Machine Translation in the Americas*, pages 223–231.

M. Snover, N. Madnani, B. Dorr, and R. Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation at the 12th Meeting of the EACL.*