A GrAF-compliant Indonesian Speech Recognition Web Service on the Language Grid for Transcription Crowdsourcing

Bayu Distiawan Trisedya
Faculty of Computer Science
Universitas Indonesia
b.distiawan@cs.ui.ac.id

Ruli Manurung
Faculty of Computer Science
Universitas Indonesia
maruli@cs.ui.ac.id

Abstract

This paper describes the development of an Indonesian speech recognition web service which complies with two standards: it operates on the Language Grid, ensuring process interoperability, and its output uses the LAF/GrAF format, ensuring data interoperability. It is part of a larger system, currently in development, that aims to collect speech transcriptions via crowdsourcing methods. Its utility is twofold: it exposes a functional speech recognizer to the web, and allows the incremental construction of a large speech corpus.

1 Background

In recent years, the initial groundwork for developing Indonesian speech recognition systems, i.e. development of phonetic models and dictionaries, as well as language and acoustic models, has been carried out (Baskoro and Adriani, 2008; Zahra et al., 2009; Huntley and Adriani, 2009). However, to build high-quality speech recognition systems, large collections of training data are needed. To achieve this, we can employ a strategy that has emerged in recent times, which capitalizes on the ubiquity of the Internet, known as crowdsourcing, i.e. relying on a large group of individuals to perform specific tasks. One successful example of this is the PodCastle project (Goto and Ogata, 2010).

This paper presents our initial efforts in developing a speech recognition system that utilizes the Language Grid platform (Ishida, 2005) to provide Indonesian speech recognition services accessible through the web and mobile devices in an efficient and practical manner, and support crowdsourcing of speech annotations through an interactive web application. Section 2 will provide an overview of the system, Sections 3 and 4 will discuss related standards, i.e. the Language Grid and the Linguistic Annotation Framework respectively, and Section 5 will present the developed speech recognition service. In Section 6 we briefly discuss the speech transcription crowdsourcing application.

2 System Overview

Building high-quality speech recognition systems requires a large collection of annotated training data in the form of spoken audio data along with validated speech transcriptions. Such resources are very costly to build, which typically involves skilled human resources such as linguistic experts. Our solution is to offer a speech recognition web service whose utility is twofold: it provides a valuable service to users, whilst allowing the construction of a large speech corpus. This service will be supplemented with an interactive web application for transcribing and correcting any arising speech recognition errors.

Furthermore, transcribed speech corpora are useful for many applications, but typically existing collections are restricted in their utility due to formatting issues of metadata. Adopting standards that ensure interoperability will maximize the

Proceedings of the 6th Linguistic Annotation Workshop, pages 67–74,
Jeju, Republic of Korea, 12-13 July 2012. ©2012 Association for Computational Linguistics

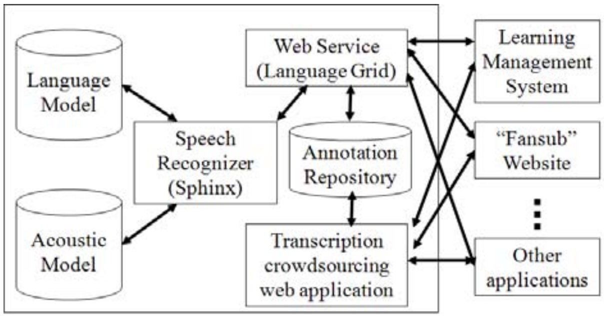

Language Model
Acoustic Model
Speech Recognizer (Sphinx)
Web Service (Language Grid)
Annotation Repository
Transcription crowdsourcing web application
Learning Management System
"Fansub" Website
⋮
Other applications

Figure 1. Overview of the system

usefulness of various resources. This can be achieved by integrating standards such as the Linguistic Annotation Framework (Ide and Romary, 2006), which focuses on data and annotation interoperability, with the Language Grid, which focuses on process interoperability. Some work in this area has already been done, e.g. by Hayashi et al. (2010) and Distiawan and Manurung (2010).

The Language Grid specification currently already includes support for speech recognition services. The defined web method requires four parameters: language identifier, speech data in Base 64 encoding, audio type, and voice type. However, the specification of the return results are not precisely defined. By providing return results of speech recognition using interoperable standards, e.g. based on GrAF (Ide and Suderman, 2007), which also provides crucial timestamp information for synchronization between audio and transcription, many further applications can be supported.

Figure 1 presents an overview of the system, which is enclosed in a rectangle. At its core is a speech recognition system, based on the CMU Sphinx open source system [1], which accesses previously developed resources such as a language model and an acoustic model. A standards-compliant "wrapper" web service exposes the functionality of this speech recognizer to the web, and aside from returning the results to the calling application, also stores the primary data along with its annotations in a RESTful annotation repository inspired by the DADA annotation server (Cassidy, 2008). These annotations are then served to a transcription crowdsourcing web application similar to the PodCastle project[2].

We envision various use cases for this system. One instance that we hope to implement is as a support to a Learning Management System, where lecture recordings are automatically transcribed and form valuable learning resources for students, who can also correct the transcriptions and make further annotations, similar to the SyNote project (Li et al., 2009). Another possible application is to support various "fansub" projects, which are Internet-based communities who provide user-created subtitles for TV shows and films in various languages.

3 The Language Grid

The Language Grid was developed in early 2005 involving many researchers from the National Institute of Information and Communication Technology (NICT), universities and research institutes around Kyoto (Ishida, 2005). The aim of the development of the Language Grid is to overcome the language barriers that often inhibit communication between people who have different languages. Many knowledge sources available on the Internet are written in different languages. This happens because there is no standard language used on the Internet: even English only accounts for 35% of the total Internet content. At the beginning of the development of the Language Grid was built machine translation which includes five languages: Chinese, Malaysian, Japanese, Korean, and English.

Researchers in various countries have developed language tools for the purposes of their own language, but unfortunately these resources are often not accessible to the public. In addition, these separate resources are only usable as atomic services that can only be used for a particular language. Therefore, the Language Grid seeks to combine resources that already exist for various languages so that they can be used by parties who need to combine them to become an integrated service. A simple example of integrated service is as follows. Imagine there are two language services for machine translation, Japanese - English (and vice versa) and Chinese - English (and vice versa). If both atomic services are deployed onto the Language Grid, it will be

[1] http://cmusphinx.sourceforge.net

[2] http://www.podcastle.jp

possible to construct a new service, i.e. Japanese - Chinese machine translation and vice versa by using English as an intermediary language.

There are two types of services on the Language Grid; the first is called the horizontal Language Grid, which combines existing language services using web services technology. The second is called the vertical Language Grid, which combines the language services on the horizontal language grid to support inter-cultural activities. An example of the vertical language grid is making a parallel text in the medical field to assist foreign patients at local hospitals (Ishida, 2005).

To support maximum interoperability on the Internet, the Language Grid relies on web services technology, in which there is WSDL, UDDI, and SOAP. The Language Grid has also been equipped with support services such as OWL ontologies, so the Language Grid has supported the Semantic Web and has been providing services for search and automatic configuration of the composite services.

Currently, the Language Grid already has a lot of services, including: Bilingual Dictionaries, Morphological Analyzer Services, Machine Translation, etc. The process of deploying and combining the language services that have been developed on the Language Grid is by the wrapping mechanism of the language resource so that it becomes a web service that can be accessed via a SOAP protocol. Rules and standards to perform the wrapping is already regulated and established by the Language Grid project through standard wrapping libraries. Until now the wrapping standard allows developers to do wrapping into a Java-based web service only using JAX-RPC library.

To combine language resources that are already available, the first step is to conduct the wrapping of language resources. A wrapper is a program that makes language resources accessible through a web service, by adjusting the input and output specifications of the NICT Language Service Interface. Thus, language resources can be registered as a language service on the Language Grid.

After the wrapper of the language resource has been completed, the wrapper is then deployed to a Language Grid Service Node, or a so-called server service provider, and will receive requests from a Language Grid Core Node or the so-called client service requester. Figure 2 shows an illustration of the data flow using the Language Grid wrapper. From Figure 2 we can see that when there is a request from a Language Grid Core Node, the Language Grid Service Node can access the language resources that have been wrapped or access another available language resource on another server using conventional HTTP and SOAP protocols, then return output according to a predetermined format.

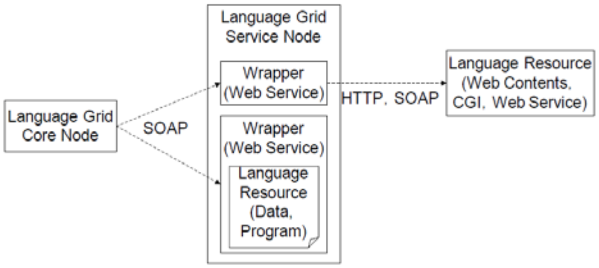

Language Grid Service Node
Wrapper (Web Service)
HTTP, SOAP
Language Resource (Web Contents, CGI, Web Service)
Language Grid Core Node
SOAP
Wrapper (Web Service)
Language Resource (Data, Program)

Figure 2. Configuration diagram of wrapper

4 Linguistic Annotation Framework

The Linguistic Annotation Framework (LAF) is a standard that provides the architecture for the creation, provision of annotation, and manipulation of linguistic resources so that encoders and annotators have the discretion to determine the format of annotation and facilitate the reuse of existing annotation. LAF was developed by ISO TC37 SC WG1-1. The two main objectives of LAF are the provision of tools to utilize and reuse linguistic data from a variety of applications at all levels of linguistic description, and the facilitation of the maintenance of a cycle of documents through various stages of the process and allowing the addition of information on existing data (Ide and Romary, 2003).

To achieve this, various principles are observed, i.e.:

1. The separation between data and annotations. Language data can only be read and not allowed to change its contents (read-only) and contains no annotations. All the annotations are contained in a separate document which is connected to the primary data (related documents). This approach is often called stand-off markup.
2. The separation between user annotation formats and a globally understood exchange,

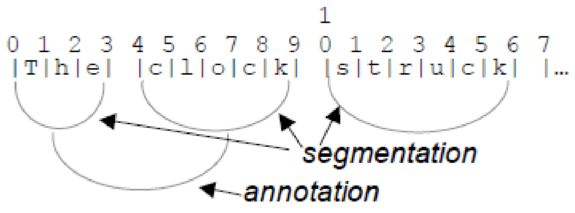
or ‘dump’, format. Users can use any format for annotations (XML, LISP, etc.). The only requirement is that the format should be mappable to the structure of data in the dump format.

3. The separation between the structure and contents of the dump format.

The Graph Annotation Format (GrAF) is one of the formats that implement the conceptual standard annotation of the Language Annotation Framework (LAF). GrAF utilizes graph theory to model the linguistic annotation that can provide the flexibility to create, represent, and incorporate some annotations into a single and integrated annotation. By utilizing a pivot LAF (dump) format, the user annotation can be transformed into a graph format. With the ability of transformation, GrAF can combine two or more annotations into a single unitary representation of annotation. To prove the concept, there have been some experiments conducted using several different annotation formats on the Wall Street Journal corpus (Ide and Suderman, 2007).

GrAF itself is an XML file that follows the general structure for the annotation that has been specified by the LAF. A GrAF document represents the structure of an annotation by two XML elements: <node> and <edge>. Either element, whether <node> or <edge>, can be labeled in accordance with the annotation information.

Annotations are saved in a separate graph from primary data. When the annotation is stored in GrAF format, then the process of merging annotations of the same primary data or annotation of the annotation reference to the primary data can be combined with existing graph merging algorithms that have been developed.



```
<!-- edges over primary data -->
<edge id="e1" from="0" to="3"/>
<edge id="e2" from="4" to="9"/>
<edge id="e2" from="10" to="16"/>
```

Figure 3. Segmentation and Construction of GrAF (Ide and Romary, 2006).

Besides the ease to the process of merging graphs, there are many other benefits obtained by the use of graph theory in the GrAF format, since a lot of software is readily available for graph manipulation, for example to show the relevance between node and edge visualization, graph traversal, as well as adding information in the graph.

One important part of GrAF is segmentation. Segmentation needs to be done because the primary data is separated from the annotation. Segmentation is performed on primary data to divide the primary data into smaller elements to be annotated. Segmentation in the primary document will eventually form a set of nodes and edges that form the basis of GrAF. Multiple segmentation documents can be defined over the primary data, and multiple annotation documents may refer to the same segmentation document (Ide and Romary, 2006). Figure 3 provides an illustration of segmentation and annotation.

There is no limitation or standard to perform segmentation. Segmentation in text documents are generally made to divide the document into a word or phrase. The word or phrase itself can still be segmented into smaller elements in the form of characters that form a word or phrase.

In its implementation, the text document segmentation is done by forming edges linking some contiguous tokens (characters) in a document. This is done by determining the position of tokens in a document. Then, the edge will be considered as a node (in this case it can be a word or phrase). The GrAF format requires the specification of the primary data location (i.e. URL) in order to interpret the segmentation information.

5 Integration of Language Grid Web Service and GrAF-based Annotation for Speech Recognition

To facilitate the integration between the services that are available on the Language Grid to provide a LAF-based standard annotation, we use the GrAF-aware Language Grid framework (Distiawan and Manurung, 2010). As shown in Figure 4, this

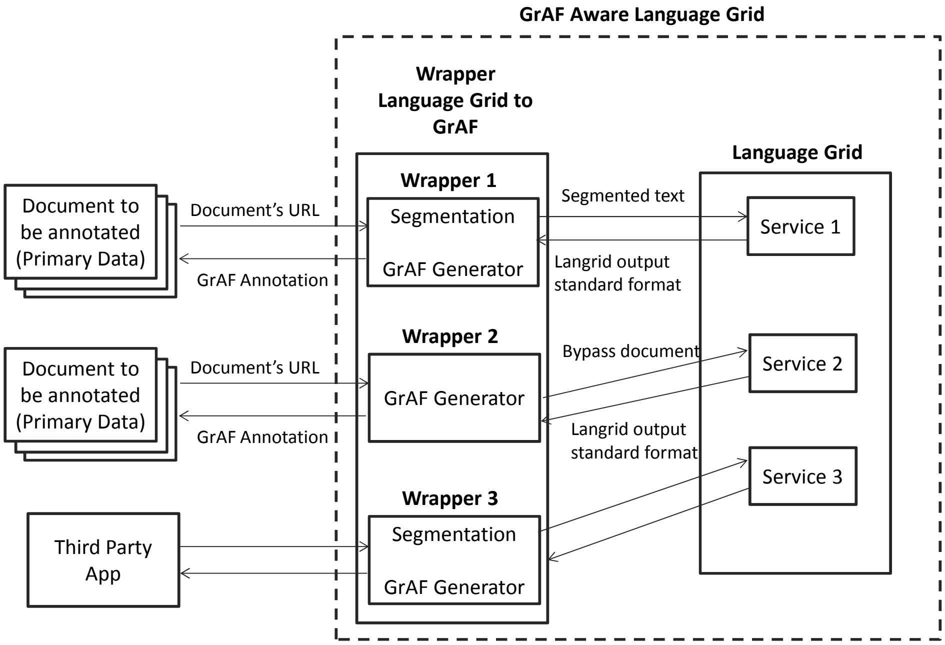


Figure 4. GrAF-aware Language Grid Framework

framework introduces an additional wrapping layer that carries out two processes, namely the segmentation of the primary data (if no segmentation exists previously) and the generation of the GrAF XML document. Segmentation is performed on the primary data, and forms the basis which further linguistic annotations refer to.

This wrapping layer is also responsible for recording the segmentation for matching the generation of GrAF XML documents to the native segmentation of the wrapped Language Grid service. After the segmentation is completed, the additional layer will send each element of segmentation results to obtain the annotation from the existing services on the Language Grid. The Language Grid output generated will then be used to fill the GrAF XML document.

This additional layer is developed as a web service so it is expected to be easily integrated into the Language Grid. One additional layer will correspond with one service on the Language Grid; this is to facilitate modularity and the reusability of additional layers in other applications.

The development of additional layers to combine services on the Language Grid with a standard GrAF annotation is done by using Java SOAP web services technology, but this does not rule out the possibility of an additional layer development using RESTFul web services technology. This additional layer service receives the URL of a document as input and will generate a GrAF XML document.

The first step is to carry out primary data segmentation, because this segmentation will link the information between the primary data with the secondary data, i.e. the Language Grid-produced annotations. For text documents, segmentation is performed by splitting the document into single words, where one word will be inserted into a single token that is marked with an edge tag. Each token has information about the beginning and end index positions relative to a particular document.

Since we are developing a service relating to audio primary data, we assume that the segmentation will be defined in terms of the timestamps when utterances occur in the primary media file, whether audio or video. Thus, an utterance token is marked with an edge tag, and contains information about the beginning and end timestamps.

The second step is communication with the web services on the Language Grid, which produces the linguistic annotation, e.g. in this case, speech recognition. For our purposes, this layer is developed against a previously developed speech recognition service on the Language Grid, which in turn uses the Sphinx open-source speech recognizer.

```
<container xmlns:graf="http://www.tc37sc4.org/graf/v1.0.6b">
  <header>
    <primaryData
      loc="http://fws.cs.ui.ac.id/fedora/objects/Speech:1/datastreams/FILE/content"
      type="audio/wav"/>
  </header>
  <graph>
    <edgeSet id="Speech Segmentation">
      <instant id="e1" from="0.35" to="0.7"/>
      <instant id="e2" from="0.7" to="1.15"/>
      <instant id="e3" from="1.15" to="1.57"/>
      ...
    </edgeSet>
    <edge id="t1" ref="e1">
      <fs type="token">
        <f name="word" sVal="lima"/>
      </fs>
    </edge>
    <edge id="t2" ref="e2">
      <fs type="token">
        <f name="word" sVal="empat"/>
      </fs>
    </edge>
    ...
  </graph>
</container>
```

Figure 5. Sample GrAF segmentation and annotation from the speech recognizer

The third stage consists of the mapping of the Language Grid service output to the initial segmentation produced during the first stage. This approach allows flexibility of utilizing all currently available services on the Language Grid.

Since the initial segmentation of an audio file into utterances is carried out by the speech recognition web service, it is more efficient to conflate the three steps into one: given the source audio file, the web service will pass it on to the Sphinx speech recognition module, which can be configured to output the timestamps of when utterances also occur. Thus, the output will consist of both the segmentation and the annotation.

Figure 5 provides an example of GrAF segmentation and annotation results given an input audio file that consists of an Indonesian utterance (specifically, someone utterring a telephone number).

We adopt GrAF because of its flexibility in the provision of multiple segmentation results. Our system can output the *n*-best recognition results from Sphinx, which will be used to provide alternative recommendations from the speech recognition system to the users. It is possible that these alternative transcriptions have different segmentations. For example, Sphinx can provide two possible outputs for a document, e.g.: the best recognition result contains the word "`sedikitnya`" in the range `4.02-4.3` seconds, whereas an alternative result contains the words "`sedih`" in the range `4.02-4.2` seconds and the word "`kita`" in the range `4.2-4.3` seconds. By using GrAF annotation we can deliver both segmentation results as well as providing an appropriate annotation for each segment as follows:

```
<graph>
    <edgeSet id="Speech Segmentation">
      <instant id="e1" from="4.02" to="4.3"/>
      <instant id="e2" from="4.02" to="4.2"/>
      <instant id="e3" from="4.2" to="4.3"/>
      ...
    </edgeSet>
    <edge id="t1" ref="e1">
      <fs type="token">
        <f name="word" sVal="sedikitnya"/>
      </fs>
    </edge>
    <edge id="t2" ref="e2">
      <fs type="token">
        <f name="word" sVal="sedih"/>
      </fs>
    </edge>
    <edge id="t2" ref="e2">
      <fs type="token">
        <f name="word" sVal="kita"/>
      </fs>
    </edge>
    ...
</graph>
```

This example is still a rough idea of how we represent the primary recognition result and its alternatives in cases of different segmentations found among the results. We are still experimenting with more suitable representations.

This web service has been implemented, and can currently be accessed from the following URL: **`http://langrid.cs.ui.ac.id/GRAFSpeechRecognizer/ws/recognize?file=<URL_to_media>`**.

To support the crowdsourcing system to be developed, we use our previously developed corpus repository (Manurung et al., 2010), which will be used to store all audio or video data along with its automatic or crowdsourced GrAF annotation.

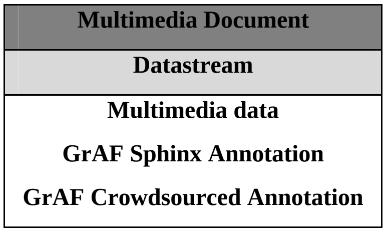


Figure 6. Fedora digital object representation

In this corpus repository, data and its annotations will be represented as a datastream in a Fedora Commons digital object that can be accessed using a persistent and unique URL. An illustration of this is shown in Figure 6. For example, the audio data can be accessed at: **`http://fws.cs.ui.ac.id/fedora/objects/Speech:1/datastreams/FILE/content`** and the automatic GrAF annotation can be accessed at: **`http://fws.cs.ui.ac.id/fedora/objects/Speech:1/datastreams/UserAnnotation-1/content`**.

6 Crowdsourcing audio transcriptions

As mentioned in Section 2, one way in which we hope to leverage this standards-compliant speech recognition web service is as a supporting tool for an interactive web application that enables users to correct the automatically produced speech transcription, which will likely still contain errors. Users will be able to play back the primary data, whether in audio or video form, and the transcription will be displayed synchronized to the media playback. They can then view and edit this information in a non-linear fashion. This application is currently under development, utilizing open standards such as HTML5 and Javascript to ensure maximum interoperability.

Several design issues arise, as follows:

1. User interface design. We are currently experimenting with various designs, e.g. displaying the transcriptions as a scrolling “ticker tape”, as a full length text field, or in static segments similar to how movie subtitles are displayed.

2. Crowdsourcing incentive scheme. A crucial aspect of successful crowdsourcing initiatives is the appropriate incentive scheme, i.e. providing motivation for users, which may be financial, sociological, or psychological in nature (Shaw et al., 2011). Our aim is to place the transcription task within a context that provides natural motivation for the user, e.g. in a learning management system (LMS), wherein students would benefit from studying and working with lecture transcriptions.

3. Utilizing user corrections. Once user corrections have been collected, we aim to feed them back into the speech recognition system by retraining the acoustic and language models. During this process, we aim to measure inter-annotator reliability to remove outliers.

7 Further Work and Summary

The development of GrAF-compliant Indonesian Speech Recognition Web Service is just the first step to built a robust Bahasa Indonesia speech recognition system. This service will be used to create an interactive website that can be used by the user to see the transcript of a video and also the user can give feedback to the incorrect transcription. Using segmentation from GrAF annotation, the transcription will be displayed adjusted to the video timeline.

We realize that our speech recognition system is still not perfect, therefore, in addition to providing the best recognition results, the GrAF-compliant Indonesian Speech Recognition Web Service will also provide some alternatives recognition result. The alternative recognition result will also displayed alongside the transciption result. By providing the alternative result, we hope the user willing to give feedback about the incorrect transciption. We will use the feedback from user to get a larger and valuable corpus to retrain the speech recognition system.

References

Sadar Baskoro and Mirna Adriani. 2008. “Developing an Indonesian Speech Recognition System”. Second MALINDO Workshop. Selangor, Malaysia.

Steve Cassidy. 2008. “A RESTful Interface to Annotations on the Web”, in Proceedings of the 2nd Linguistic Annotation Workshop (LAW II), LREC2008, Marrakech.

Masataka Goto and Jun Ogata. 2010. “PodCastle: A Spoken Document Retrieval Service Improved by User Contributions”, in Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation (PACLIC 24), pp.3-11.

Yoshihiko Hayashi, Thierry Declerck and Chiharu Narawa. 2010. “LAF/GrAF-grounded Representation of Dependency Structures”. LREC 2010, Malta.

Nancy Ide and Laurent Romary. 2003. “Outline of the International Standard Linguistic Annotation Framework,” in Proceedings of the ACL'03 Workshop on Linguistic Annotation: Getting the Model Right, Sapporo, pp. 1-5.

Nancy Ide and Laurent Romary. 2006. “Representing Linguistic Corpora and Their Annotations,” in Proceedings of LREC 2006, Genoa, Italy.

Nancy Ide and Keith Suderman. 2007. “GrAF: A Graph-based Format for Linguistic Annotations,” in Proceedings of the Linguistic Annotation Workshop, held in conjunction with ACL 2007, Prague, June 28-29, pp. 1-8.

Toru Ishida. 2005. “Language Grid: An Infrastructure for Intercultural Collaboration,” in Proceedings of the 2005 Symposium on Applications and the Internet (SAINT’06), vol., no., pp. c1- c1.

Myrna Laksman-Huntley and Mirna Adriani. 2009. “Developing Indonesian Pronunciation Dictionary”. The Third International MALINDO Workshop, Co-located Event ACL-IJCNLP 2009. Singapore.

Yunjia Li et al. 2009. “Synote: Enhancing Multimedia E-learning with Synchronised Annotation”, in Proceedings of the first ACM international workshop on Multimedia technologies for distance learning. Beijing.

Ruli Manurung, Bayu Distiawan, and Desmond Darma Putra. 2010. “Developing an Online Indonesian Corpora Repository”. in Proceedings of the 24th Pacific Asia Conference on Language, Information, and Computation (PACLIC 2010), pp.243-249, Sendai, Japan.

Aaron Shaw, John Horton, and Daniel Chen. 2011. “Designing incentives for inexpert human raters”. in Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work (CSCW '11), pp.275-284, Hangzhou, China.

Bayu Distiawan Trisedya and Ruli Manurung. 2010. “Extending the Language Grid for GrAF-based Linguistic Annotations”, in Proceedings of the International Conference on Advanced Computer Science and Information Systems (ICACSIS 2010). Bali.

Amalia Zahra, Sadar Baskoro and Mirna Adriani. 2009. “The Performance of Speech Recognition System for Bahasa Indonesia Using Various Speech Corpus”. Second Workshop on Technologies and Corpora for Asia-Pacific Speech Translation (TCAST 2009). Singapore.